

MACHINE LEARNING FOR TRAJECTORIES OF PARAMETRIC NONLINEAR DYNAMICAL SYSTEMS

Roland Pulch & Maha Youssef*

*Institute for Mathematics and Computer Science, University of Greifswald,
Walther-Rathenau-Str. 47, D-17489 Greifswald, Germany*

*Address all correspondence to: Roland Pulch, Institute for Mathematics and Computer Science, University of Greifswald, Walther-Rathenau-Str. 47, D-17489 Greifswald, Germany, E-mail: roland.pulch@uni-greifswald.de

Original Manuscript Submitted: 3/5/2020; Final Draft Received: 6/18/2020

We investigate parameter-dependent nonlinear dynamical systems consisting of ordinary differential equations or differential-algebraic equations. A single quantity of interest is observed, which depends on the solution of a system. Our aim is to determine efficient approximations of the trajectories belonging to the quantity of interest in the time domain. We arrange a set of samples including trajectories of this quantity. A proper orthogonal decomposition of this data yields a reduced basis. Consequently, the mapping from the parameter domain to the basis coefficients is approximated. We apply machine learning with artificial neural networks for this approximation, where the degrees of freedom are fitted to the data of the sample trajectories in a nonlinear optimization. Alternatively, we consider a polynomial approximation, which is identified by regression, for comparison. Furthermore, concepts of sensitivity analysis are examined to characterize the impact of an input parameter on the output of the exact mapping or the approximations from the neural networks. We present results of numerical computations for examples of nonlinear dynamical systems.

KEY WORDS: *nonlinear dynamical system, differential-algebraic equation, initial value problem, parametric model order reduction, proper orthogonal decomposition, machine learning, neural network, polynomial regression, sensitivity analysis*

1. INTRODUCTION

Mathematical modeling of real-world problems often yields dynamical systems in science and engineering. We consider initial value problems for nonlinear systems of ordinary differential equations (ODEs) or differential-algebraic equations (DAEs), which depend on physical parameters. A transient quantity of interest (QoI) is defined depending on the solution of a system. Many evaluations of the QoI are required for different realizations of the parameters in some tasks like optimization and uncertainty quantification; see Xiu (2010), for example. Often a time integration of the dynamical system is costly for realistic applications. Thus our aim is to determine efficient approximations of the trajectories associated with the parameter-dependent QoI. An evaluation of this approximation should be cheap, while good accuracy is still achieved for most of the relevant parameter values.

We determine the trajectories of the QoI for parameter samples in some bounded parameter domain. Proper orthogonal decomposition (POD) represents a method for projection-based

model order reduction (Antoulas, 2005; Kunisch and Volkwein, 2001). We use a similar POD approach to obtain a reduced basis for the trajectories. Any trajectory is approximated by a function in a low-dimensional space. The approximation is uniquely determined by the coefficients in the reduced basis. Hence we obtain a mapping between finite-dimensional spaces, where a parameter value is mapped to the basis coefficients.

Now the task is to determine an efficient approximation of the mapping between finite-dimensional spaces. This procedure can be seen as a kind of parametric model order reduction (pMOR) (Benner et al., 2015). Instead of solving the full-order model consisting of the dynamical system, a parameter-dependent reduced-order model is constructed. However, our reduced-order model is not a dynamical system anymore. On the one hand, we apply a concept of machine learning based on artificial neural networks (Du and Swamy, 2014; Goodfellow et al., 2017). On the other hand, we use a multivariate polynomial regression for comparison (Seber and Lee, 2003). In both approaches, an optimization process is performed to identify the degrees of freedom appropriately, where the data of the sample trajectories is included. This optimization is called training in the case of machine learning. Neural networks (NNs) imply a nonlinear optimization problem, whereas the polynomial fit requires just the solution of linear least squares problems. Both approaches represent data-driven methods.

Similar problems have also been tackled by NNs in previous works. The POD method was used for parametric stationary solutions of partial differential equations in Hesthaven and Ubbiali (2018) and Yu and Hesthaven (2019). Trajectories of solutions satisfying (non-parametric) autonomous systems of ODEs were reproduced by Qin et al. (2019).

Furthermore, we discuss a variance-based sensitivity analysis of the input–output behavior in the mapping between the finite-dimensional spaces. The total effect sensitivity indices yield a quantification of the impact of the individual parameters (Saltelli et al., 2008; Sobol and Kucherenko, 2009). Thus a ranking of the importance is feasible for the parameters. The variance-based sensitivity analysis can be performed for both the exact mapping and an approximation. Alternatively, we also investigate the weights in a trained NN to obtain information about the sensitivities with respect to the input parameters.

We present numerical results for two examples, which are nonlinear systems of DAEs modeling electric circuits. Both the machine learning approach and the polynomial regression are used to obtain the approximations. The errors of the methods are analyzed and compared. In addition, we illustrate the sensitivity analysis by the examples.

In this article, we introduce parametric nonlinear dynamical systems and the investigated problem in Section 2. The POD method yields the representation in the reduced basis, and the polynomial approximation is outlined. The variance-based sensitivity indices are formulated for our problem. In Section 3, we apply artificial NNs for the approximation. We define the weight-based sensitivity measures. The sources of errors are discussed for the entire numerical method. Finally, Section 4 demonstrates results of numerical computations for the two examples.

2. PROBLEM DEFINITION

We describe the problem in this section, which will be tackled by artificial NNs.

2.1 Nonlinear Dynamical Systems

We consider nonlinear dynamical systems in the form

$$\begin{aligned} M(p)\dot{x}(t, p) &= f(t, x(t, p), p) \\ y(t, p) &= g(x(t, p), p). \end{aligned} \quad (1)$$

The mass matrix M and/or the right-hand side f depend on physical parameters $p \in \Pi \subseteq \mathbb{R}^q$. Hence the state variables or inner variables $x : [t_0, t_{\text{end}}] \times \Pi \rightarrow \mathbb{R}^n$ depend on time as well as the parameters. If the mass matrix is non-singular, then we obtain a system of ODEs. In contrast, a singular mass matrix implies a system of DAEs. Initial value problems (IVPs) are specified by

$$x(t_0, p) = x_0(p), \quad (2)$$

with a predetermined function $x_0 : \Pi \rightarrow \mathbb{R}^n$. In the case of DAEs, the initial values have to be consistent. The consistency conditions represent systems of algebraic equations. Consistent initial values typically depend on the physical parameters of the system.

A QoI $y : [t_0, t_{\text{end}}] \times \Pi \rightarrow \mathbb{R}^{n_{\text{qoi}}}$ is defined by the function g depending on the solution x of the dynamical system [Eq. (1)]. We assume that a single QoI ($n_{\text{qoi}} = 1$) is under investigation. Often y depends linearly on the variables x . Sometimes y coincides with a single component of x .

We suppose that each parameter is located in a compact interval: $p_j \in [p_{j,\min}, p_{j,\max}]$ for $j = 1, \dots, q$. Consequently, the parameter domain is a multidimensional cuboid. Without loss of generality, we assume that the parameter domain is the unit hypercube $\mathcal{H}_q = [0, 1]^q$. In this standardization, the bijective mapping reads as

$$\Xi : \mathcal{H}_q \rightarrow \Pi, \quad p_j \mapsto p_{j,\min}(1 - p_j) + p_{j,\max} p_j \quad \text{for } j = 1, \dots, q,$$

where Π is the multidimensional cuboid incorporating the physical quantities.

The following strategy can be applied for boundary value problems (BVPs) of dynamical systems as well, because only the information of the trajectories of the QoI is included. It does not matter if the trajectories are computed by IVPs or BVPs. The techniques are data-driven.

2.2 Proper Orthogonal Decomposition

In Mifsud et al. (2016) POD was used for ensembles of solutions at different parameter values. We employ this idea for the transient problems [Eq. (1)]. A set of parameter samples

$$\mathcal{S} = \{p_1, \dots, p_k\} \subset \mathcal{H}_q \quad (3)$$

is generated. For example, random samples can be chosen in \mathcal{H}_q .

A discretization in time implies a grid with points t_1, \dots, t_m satisfying $t_0 \leq t_1 < t_2 < \dots < t_m \leq t_{\text{end}}$. The initial point t_0 may be included in the grid. Consequently, a time integration of the IVPs [Eqs. (1) and (2)] yields the values of the QoI in the time points. We assume that the errors of the time integration are negligible. Let $Y \in \mathbb{R}^{m \times k}$ be the matrix with entries $y_{ij} := y(t_i, p_j)$, which represent discrete observations at the parameter samples [Eq. (3)]. We perform a POD by the singular value decomposition

$$Y = USV^\top, \quad (4)$$

with a diagonal matrix $S \in \mathbb{R}^{m \times k}$ containing the singular values $\sigma_1, \sigma_2, \dots, \sigma_s$ in descending order with $s = \min\{m, k\}$. The orthogonal matrix $U \in \mathbb{R}^{m \times m}$ includes the associated basis

vectors u_1, \dots, u_m in its columns. Taking the r dominant singular values, we form the smaller matrix $\tilde{U} \in \mathbb{R}^{m \times r}$ with the columns u_1, \dots, u_r . Let $a = (a_1, \dots, a_r)^\top \in \mathbb{R}^r$ be coefficients. If $y(\cdot, p)$ is a trajectory of the QoI for any $p \in \mathcal{H}_q$, then its representation in the reduced basis reads as

$$w(p) := \begin{pmatrix} y(t_1, p) \\ \vdots \\ y(t_m, p) \end{pmatrix} \approx \begin{pmatrix} \tilde{y}(t_1, p) \\ \vdots \\ \tilde{y}(t_m, p) \end{pmatrix} := \sum_{\ell=1}^r a_\ell u_\ell = \tilde{U}a. \quad (5)$$

The best approximation of the coefficients a is obtained by the projection

$$\hat{a}(p) = \tilde{U}^\top w(p). \quad (6)$$

The accuracy of the POD is characterized by the requirement

$$\sum_{\ell=1}^r \sigma_\ell^2 > \delta \sum_{\ell=1}^s \sigma_\ell^2, \quad (7)$$

including a user-specified tolerance δ , say $\delta \geq 0.999$ (Benner et al., 2015, p. 502). The smallest rank r is chosen such that the condition [Eq. (7)] is satisfied.

If the coefficients are given, then the right-hand side of Eq. (5) implies an approximation of the transient QoI. The time integration produces the transient QoI in the left-hand side of Eq. (5) and thus the projection [Eq. (6)] yields the mapping

$$\Gamma : \mathcal{H}_q \rightarrow \mathbb{R}^r, \quad p \mapsto \hat{a}. \quad (8)$$

We want to approximate this nonlinear function between low-dimensional spaces.

2.3 Polynomial Regression

For comparison, we arrange a straightforward polynomial approximation of the mapping [Eq. (8)]. We apply the Legendre polynomials as basis functions (Stoer and Bulirsch, 2002, p. 177) because well-conditioned problems are expected in comparison to other bases like the monomial basis, for example. The polynomial approximation reads as $\tilde{a} : \mathcal{H}_q \rightarrow \mathbb{R}^r$ with

$$\tilde{a}(p) = \sum_{i=1}^s c_i \Phi_i(p), \quad (9)$$

including vectors $c_i = (\gamma_{i1}, \dots, \gamma_{ir})^\top \in \mathbb{R}^r$. The multivariate basis polynomials are the products of the (univariate) Legendre polynomials

$$\Phi_i(p) = L_{i_1}(p_1) L_{i_2}(p_2) \cdots L_{i_q}(p_q), \quad (10)$$

for $i = 1, \dots, s$ with $p = (p_1, p_2, \dots, p_q)^\top$. There is a one-to-one mapping from the integers i to the multiindices (i_1, \dots, i_q) . The traditional Legendre polynomials are linearly transformed from their domain of dependence $[-1, 1]$ to $[0, 1]$. The degree of $L_j : [0, 1] \rightarrow \mathbb{R}$ is exactly j . Hence the total degree of a multivariate polynomial [Eq. (10)] is $i_1 + \dots + i_q$. The number of basis polynomials up to a total degree d is (Xiu, 2010, p. 65),

$$s = \frac{(q+d)!}{q!d!}. \quad (11)$$

Now we employ the set [Eq. (3)] consisting of k parameter samples, which was also used for the computation of the reduced basis in the POD method. Let $s < k$. We arrange a Vandermonde matrix $V \in \mathbb{R}^{k \times s}$ and right-hand sides $b_\ell \in \mathbb{R}^k$ for $\ell = 1, \dots, r$ by

$$V = \begin{pmatrix} \Phi_1(p^{(1)}) & \Phi_2(p^{(1)}) & \dots & \Phi_s(p^{(1)}) \\ \Phi_1(p^{(2)}) & \Phi_2(p^{(2)}) & \dots & \Phi_s(p^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(p^{(k)}) & \Phi_2(p^{(k)}) & \dots & \Phi_s(p^{(k)}) \end{pmatrix} \quad \text{and} \quad b_\ell = \begin{pmatrix} \hat{a}_\ell(p^{(1)}) \\ \hat{a}_\ell(p^{(2)}) \\ \vdots \\ \hat{a}_\ell(p^{(k)}) \end{pmatrix}.$$

Now we solve the linear least squares problems

$$\min_{z_\ell \in \mathbb{R}^s} \|Vz_\ell - b_\ell\|_2, \quad (12)$$

including the Euclidean norm $\|\cdot\|_2$. Each solution z_ℓ yields the coefficients $\gamma_{1\ell}, \dots, \gamma_{s\ell}$ for $\ell = 1, \dots, r$. Therein, a QR -decomposition (Golub and van Loan, 1996) of the matrix V can be reused for each right-hand side. Since this decomposition dominates the computational effort, the dimension r of the reduced basis is not significant. Consequently, the approximation [Eq. (9)] is identified. This approach is also called (multivariate) polynomial regression as in Seber and Lee (2003).

The polynomial regression may suffer from the effect of overfitting in the case of higher-degree polynomials. A regularization like Tikhonov's method or (discrete) \mathcal{L}^2 -regularization can prevent overfitting (Wang, 2019). However, a similar approximation error often results by simply restricting to polynomials of lower degree.

Furthermore, we note that a polynomial approximation can be constructed using the trajectories of the samples in the discrete time points (without a reduced basis). Thus the approximation error of the POD method is avoided. The computation work does not become much larger than in our approach, since the matrix of the linear least squares problems is identical in all time points. Yet a separate polynomial occurs for each time point, which generates a large number of polynomials. In contrast, the number of polynomials is equal to the dimension of the reduced basis in our approach. Hence a more compact description of the problem is achieved.

2.4 Sensitivity Analysis

There are derivative-based sensitivity measures and variance-based sensitivity measures (Sobol and Kucherenko, 2009). We consider a variance-based approach. Let $\mathcal{H}_q = [0, 1]^q$ be the unit hypercube again. Given a function $f : \mathcal{H}_q \rightarrow \mathbb{R}$, we assume that $f \in \mathcal{L}^2(\mathcal{H}_q)$. The total variance of f reads as

$$V(f) = \int_{\mathcal{H}_q} f(p)^2 \, dp - \left(\int_{\mathcal{H}_q} f(p) \, dp \right)^2. \quad (13)$$

A sensitivity analysis is obsolete in the case of $V(f) = 0$, because f becomes a constant function. Thus we assume that $V(f) > 0$. Variance-based sensitivity measures often require the computation of partial variances. We define the partial variances using polynomial chaos expansions (PCEs); see Sudret (2008) or Pulch and Narayan (2019). The function f exhibits the PCE

$$f(p) = \sum_{i=1}^{\infty} \hat{f}_i \Phi_i(p), \quad (14)$$

including the multivariate Legendre polynomials [Eq. (10)]; see Xiu (2010). The coefficients read as

$$\hat{f}_i = \langle f, \Phi_i \rangle = \int_{\mathcal{H}_q} f(p) \Phi_i(p) \, dp, \quad (15)$$

using the inner product of the Hilbert space $\mathcal{L}^2(\mathcal{H}_q)$. The series [Eq. (14)] converges in the norm of $\mathcal{L}^2(\mathcal{H}_q)$. We define the index sets

$$I_j = \{i \in \mathbb{N} : \Phi_i \text{ is non-constant in } p_j\},$$

for $j = 1, \dots, q$. Now the partial variances read as

$$V_j(f) = \sum_{i \in I_j} |\hat{f}_i|^2 \quad \text{for } j = 1, \dots, q. \quad (16)$$

An alternative formula of the same partial variances is given in Sobol (2001).

The total effect sensitivity indices are defined by

$$S_j^T(f) = \frac{V_j(f)}{V(f)} \quad \text{for } j = 1, \dots, q, \quad (17)$$

using Eqs. (13) and (16). It follows that $0 \leq S_j^T \leq 1$ for each j . The sensitivity indices [Eq. (17)] quantify the impact of each parameter on the variability of the function f .

In numerical methods, we have to replace the PCE [Eq. (14)] by approximations like Eq. (9). First, the series is truncated to a finite sum. Second, the coefficients in Eq. (15) are approximated. Since the inner products represent multivariate integrals, quadrature methods or cubature methods can be used.

We consider the mapping of Eq. (8): $\Gamma : \mathcal{H}_q \rightarrow \mathbb{R}^r, p \mapsto \hat{a}(p)$. The above sensitivity analysis is applicable to each component of Γ separately. Thus the sensitivity indices are $S_j^T(\hat{a}_i)$ for $j = 1, \dots, q$ and $i = 1, \dots, r$, which form an array of rq quantities. However, the importance of the coefficients \hat{a}_i decreases for increasing i due to the decay of the singular values in the decomposition [Eq. (4)]. Alternatively, we observe the sensitivity measures

$$S_j^T \left(\sum_{i=1}^r \hat{a}_i^2 \right) \quad \text{for } j = 1, \dots, q, \quad (18)$$

which allows for a more compact discussion. These sensitivity indices characterize the impacts of the parameters on the (Euclidean) norm of the low-dimensional representation.

Furthermore, an approximation $\tilde{\Gamma}$ of the mapping [Eq. (8)] can be used to compute the sensitivity indices with a low computational effort, because the evaluations of $\tilde{\Gamma}$ are cheaper than the evaluations of Γ . The approximations are obtained from either the above polynomial approach or an artificial NN.

3. MACHINE LEARNING

We employ a strategy of machine learning to solve the problem introduced in Section 2.

3.1 Artificial Neural Networks

We apply an artificial NN (Du and Swamy, 2014; Genzel and Kutyniok, 2019) to represent the input–output relation of the mapping [Eq. (8)]. A similar approach was used for spatial solutions of partial differential equations in Yu and Hesthaven (2019).

Figure 1 illustrates the schematic of an NN with three hidden layers. In the general case, let $L + 1$ be the total number of layers and N_ℓ be the number of neurons in the ℓ th layer for $\ell = 0, 1, \dots, L$. Hence the number of hidden layers is $L - 1$. The values N_0 and N_L denote the numbers of input neurons and output neurons, respectively. The mathematical modeling of an NN consists of a chain of operators

$$\Psi = T_L \circ \rho \circ T_{L-1} \circ \rho \circ T_{L-2} \circ \dots \circ \rho \circ T_2 \circ \rho \circ T_1. \quad (19)$$

The operators $T_\ell : \mathbb{R}^{N_{\ell-1}} \rightarrow \mathbb{R}^{N_\ell}$ are affine-linear functions:

$$T_\ell(z) = A_\ell z + b_\ell,$$

with matrices $A_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and vectors $b_\ell \in \mathbb{R}^{N_\ell}$. The entries of A_ℓ and b_ℓ are called weights and biases, respectively. The operator ρ represents a nonlinear activation function $\rho : \mathbb{R} \rightarrow \mathbb{R}$; for example, the hyperbolic tangent sigmoid function

$$\rho(x) = \frac{2}{1 + e^{-2x}} - 1, \quad (20)$$

or the rectified linear unit (ReLU)

$$\rho(x) = \begin{cases} 0 & \text{for } x < 0, \\ x & \text{for } x \geq 0. \end{cases}$$

In Eq. (19), the function ρ is evaluated on a vector separately for each component. If the number of hidden layers is larger or equal to 3, then the model is called a deep NN (deep learning). Otherwise, the model represents a shallow NN.

In our application, there are q inputs given by a parameter tuple $p \in \mathcal{H}_q$. The r outputs are the coefficients a in Eq. (5) associated with the reduced basis. The degrees of freedom (DOFs) are the weights and biases $\Theta = (A_\ell, b_\ell)_{\ell=1}^L$ in the optimization problem. An appropriate choice is determined by a minimization of the distances $\Gamma(p_j) - \Psi(p_j)$ for realizations $p_j \in \mathcal{H}_q$ of the parameters. A norm or distance function, which quantifies these differences, is called a performance function in the context of NNs. Typical performance functions are the mean squared error or the mean absolute error.

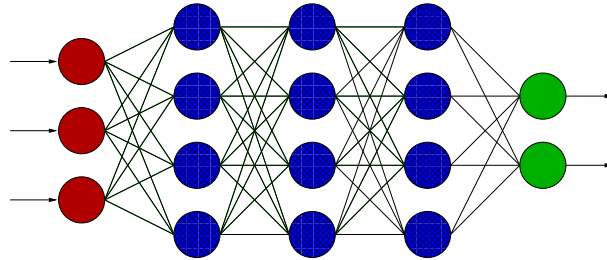


FIG. 1: Artificial neural network with input layer (left), hidden layers (center), and output layer (right)

In the determination of an NN, three sets of parameter samples

$$\mathcal{S}_{\text{train}} = \{p_1, \dots, p_k\} \subset \mathcal{H}_q, \quad (21)$$

$$\mathcal{S}_{\text{valid}} = \{q_1, \dots, q_{k'}\} \subset \mathcal{H}_q, \quad (22)$$

$$\mathcal{S}_{\text{test}} = \{r_1, \dots, r_{k''}\} \subset \mathcal{H}_q, \quad (23)$$

are arranged, which are pairwise disjoint. The training set [Eq. (21)] is used to identify the DOFs in an iterative optimization method. Thus the performance function always decreases monotone for the training set in the iteration. The validation set [Eq. (22)] yields additional data to prevent an overfitting. The iteration is stopped if the performance function increases for the validation set. The test set [Eq. (23)] is not used in the optimization method at all. This independent set allows for an estimation of the accuracy achieved by a trained NN. In our application, we employ the set of samples [Eq. (3)], used in the POD method, also as the training set [Eq. (21)].

Concerning the context of pMOR, a technique consists of an offline phase and an online phase. In our offline phase, the sample trajectories are computed and an NN is trained. The computation of the trajectories involves significant computation work, since the nonlinear dynamical systems have to be solved. In our online phase, a trained NN is evaluated for possibly many parameter values, which is cheap.

3.2 Errors of the Methods

The approximations $\Psi(p) = \tilde{a}(p)$ from Eq. (19) imply the approximate trajectories $\tilde{y}(t_i, p)$ for each realization p of the parameters in Eq. (5). The total error consists of three parts:

- (1) The numerical error of the time integration,
- (2) The approximation error with respect to the reduced basis from POD,
- (3) The approximation error of the NN.

We impose high accuracy requirements in the numerical time integration. Consequently, the error of part (1) becomes negligible. Although a tolerance $\delta \approx 1$ is applied in the condition [Eq. (7)], the error of part (2) may be relatively large for some parameter values if the associated trajectory is significantly different from the sample trajectories. Hence a decline of the error within part (2) also requires an increase in the number of samples in the POD. In the alternative approach of Section 2.3, just part (3) changes into the error of the polynomial approximation.

We estimate the error by a discrete \mathcal{L}^1 -norm in time. Given a parameter tuple $p \in \mathcal{H}_q$, this error reads as

$$E(p) = \frac{1}{t_{\text{end}} - t_0} \sum_{i=1}^{m-1} (t_{i+1} - t_i) |y(t_i, p) - \tilde{y}(t_i, p)|, \quad (24)$$

assuming $t_0 = t_1$ and $t_{\text{end}} = t_m$. Our reference values $y(t_i, p)$ will still include an error of a numerical time integration. However, this error is negligible due to the high accuracy of the time integration. In the case of sample sets [Eqs. (21)–(23)], we observe statistics of the errors [Eq. (24)] like the mean value and the sample variance, for example.

3.3 Sensitivity Analysis Using Weights

In the field of machine learning and NNs, there is a sensitivity analysis based on a layer-wise relevance propagation and associated relevance scores (Montavon et al., 2018). However, the relevance scores depend on the inputs of the NN; that is different input values imply different relevance scores. In contrast, the variance-based sensitivity indices illustrated in Section 2.4 represent global sensitivity measures, which are defined for the complete parameter domain. Thus we examine the variance-based sensitivity analysis for our problem.

If an NN represents a good approximation of the mapping [Eq. (8)] meaning that $\Gamma(p) \approx \Psi(p)$ for all p , then the total effect sensitivity indices also agree for the mappings Γ and Ψ . We investigate the magnitudes of the weights in a trained NN to obtain an alternative sensitivity analysis. In the NN model [Eq. (19)], the first operator T_1 describes the mapping from the input layer to the first hidden layer. It holds that $T_1(z) = A_1 z + b_1$ with matrix $A_1 \in \mathbb{R}^{N_1 \times q}$ and vector $b_1 \in \mathbb{R}^{N_1}$. Let w_{ij} for $i = 1, \dots, N_1$ and $j = 1, \dots, q$ be the weights in A_1 . We define sensitivity measures by the Euclidean norm of the set of weights associated to the j th input:

$$S_j^W = \sum_{i=1}^{N_1} w_{ij}^2 \quad \text{for } j = 1, \dots, q. \quad (25)$$

The square of the Euclidean norm is used for a comparison to the variance-based sensitivity analysis, because the partial variances [Eq. (16)] are sums of squares. We do not consider the weights involved in the subsequent hidden layers, because such a weight cannot be assigned to a specific input any more.

In general, the number of neurons in a hidden layer is often chosen larger than the number of input neurons. Thus it holds that $N_1 > q$. Numerical computations of test examples show that a sensitivity coefficient [Eq. (25)] may not be small, even though the influence of the associated parameter on the outputs is insignificant. It follows that the first hidden layer gets input from a insignificant parameter, which is averaged out or canceled out in the subsequent layers.

We propose an approach to avoid this behavior. Assume that an NN is sufficiently accurate with $L - 1$ hidden layers of sizes N_1, \dots, N_{L-1} . We extend this network to L hidden layers with sizes $\hat{N}_1, \dots, \hat{N}_L$ using $\hat{N}_1 = N_0$, $\hat{N}_\ell = N_{\ell-1}$ for $\ell = 2, \dots, L$. The activation function between the input layer and the first hidden layer is chosen purely linear ($\rho(x) = x$ for all x); that is the identity operator. This extended NN reads as

$$\hat{\Psi} = \hat{T}_{L+1} \circ \rho \circ \hat{T}_L \circ \rho \circ \hat{T}_{L-1} \circ \dots \circ \rho \circ \hat{T}_3 \circ \rho \circ \hat{T}_2 \circ \hat{T}_1. \quad (26)$$

The approximation quality of the extended NN is at least as good as that in the original NN, because choosing \hat{T}_1 as the identity and $\hat{T}_\ell = T_{\ell-1}$ for $\ell = 2, \dots, L + 1$ implies $\Psi = \hat{\Psi}$. However, the input information is not spread around a larger number of neurons in the first hidden layer. Hence this approach enforces a compact propagation of the information from the inputs. Now the sensitivity measures [Eq. (25)] are investigated for the extended NN [Eq. (26)], where it holds that $\hat{N}_1 = N_0 = q$.

The concept of the sensitivity measures [Eq. (25)] is heuristic. We will investigate the computed sensitivity indicators for the examples in Section 4. In particular, a comparison between Eqs. (17) and (25) is presented.

4. NUMERICAL RESULTS

We investigate two examples of nonlinear dynamical systems. All computations were performed on a FUJITSU Esprimo P920 Intel(R) Core(TM) i5-4570 CPU with 3.20 GHz (4 cores) and the Microsoft Windows 10 operating system. The software package MATLAB (version 9.7.0.1190202/R2019b) produced the numerical results. The NNs were trained using its deep learning toolbox.

4.1 Example: Transistor Amplifier

We consider the electric circuit of a transistor amplifier shown in Fig. 2. This circuit includes three capacitances, six resistances, and a bipolar transistor. In Hairer and Wanner (1996), a mathematical model is given, which consists of five DAEs for five unknown node voltages u_1, \dots, u_5 ($n = 5$). The mass matrix and the right-hand side of Eq. (1) read as

$$M = \begin{pmatrix} -C_1 & C_1 & 0 & 0 & 0 \\ C_1 & -C_1 & 0 & 0 & 0 \\ 0 & 0 & -C_2 & 0 & 0 \\ 0 & 0 & 0 & -C_3 & C_3 \\ 0 & 0 & 0 & C_3 & -C_3 \end{pmatrix}$$

$$f = \begin{pmatrix} \frac{u_1}{R_0} \\ u_2 \left(\frac{1}{R_1} + \frac{1}{R_2} \right) + (1 - \gamma)w(u_2 - u_3) \\ \frac{u_3}{R_3} - w(u_2 - u_3) \\ \frac{u_4}{R_4} + \gamma w(u_2 - u_3) \\ \frac{u_5}{R_5} \end{pmatrix} + \begin{pmatrix} -\frac{u_{in}}{R_0} \\ -\frac{u_{op}}{R_2} \\ 0 \\ -\frac{u_{op}}{R_4} \\ 0 \end{pmatrix}.$$

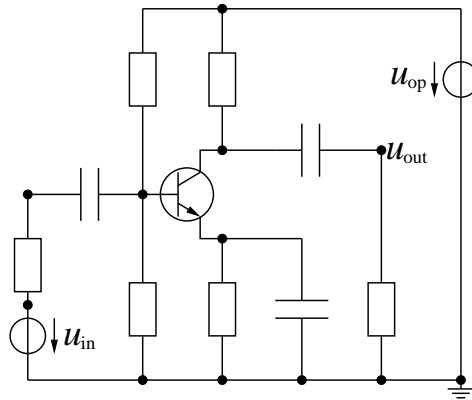


FIG. 2: Electric circuit of a transistor amplifier

The current-voltage relation of the bipolar transistor is described by the nonlinear function

$$w(u) = \alpha \left[\exp\left(\frac{u}{\beta}\right) - 1 \right], \quad (27)$$

with constants $\alpha = 10^{-6}$, $\beta = 0.026$, and $\gamma = 0.99$. We use nominal parameter values as given in Hairer and Wanner (1996): capacitances $C_1 = 10^{-6}$, $C_2 = 2 \cdot 10^{-6}$, and $C_3 = 3 \cdot 10^{-6}$; resistances $R_0 = 1000$, $R_1 = \dots = R_5 = 9000$; and operating voltage $u_{\text{op}} = 6$. The differential index of the system is one. We supply a harmonic oscillation

$$u_{\text{in}}(t) = A \sin\left(\frac{2\pi}{T}t\right), \quad (28)$$

with period $T = 0.01$ and amplitude $A = 0.4$ as input voltage. The output voltage $u_{\text{out}} = u_5$ represents the QoI.

We consider parameter variations in capacitances, resistances, and operating voltage. Variability of the parameters within the transistor model is not examined. Thus the dimension of the parameter domain is $q = 10$. Parameter variations of this example were also investigated for another purpose in Pulch (2019). A variation of 20% around the above nominal value is set for each parameter, which forms the multidimensional cuboid $\Pi \subset \mathbb{R}^{10}$.

Concerning the numerical solution of IVPs, we use the function `ode15s` in MATLAB, which is a multistep method based on the numerical differentiation formula (NDF; Shampine and Reichelt, 1997). We specify the same initial condition as a starting value for all parameters and the method determines consistent initial values [Eq. (2)] depending on the parameters. The time integrations are performed in the interval $[t_0, t_{\text{end}}] = [0, 0.03]$ with local error control using relative tolerance $\varepsilon_r = 10^{-6}$ and absolute tolerance $\varepsilon_a = 10^{-8}$.

We produce the sets [Eqs. (21)–(23)] with $k = k' = k'' = 1000$ samples using pseudo random numbers in \mathcal{H}_{10} . The QoI is obtained in $m = 500$ equidistant points in the time interval $[t_0, t_{\text{end}}]$ including t_0 and t_{end} , where the accuracy of the output agrees to the predetermined tolerances. Figure 3 illustrates both the trajectory for the mean values of the parameters and several trajectories for different parameter samples.

In the POD method, we apply the training set [Eq. (21)] only. The computed singular values are shown in Fig. 4. We observe a fast decay of the singular values. The reduced dimension $r = 9$ is the smallest number satisfying the accuracy requirement [Eq. (7)] for the threshold $\delta = 0.999$.

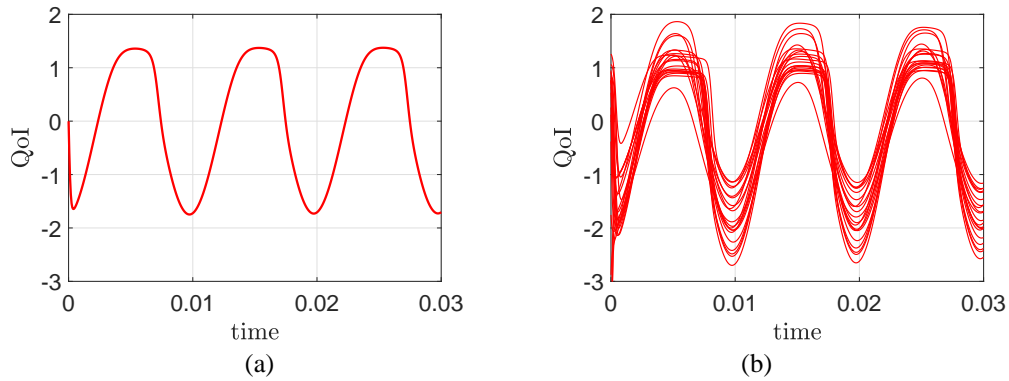


FIG. 3: Trajectory of QoI for mean value of parameters (a) and 20 trajectories of QoI for different parameter samples (b) produced by transistor amplifier circuit

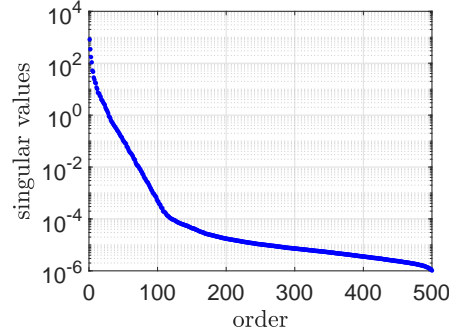


FIG. 4: Singular values from POD for matrix including the samples of the QoI in transistor amplifier example

We train two NNs: a network with two hidden layers including 30 neurons in each layer and a network with three hidden layers including 20 neurons. The activation function used is the hyperbolic tangent relation [Eq. (20)]. The performance function is the mean squared error. The Levenberg-Marquardt method (Du and Swamy, 2014, p. 130) executes the nonlinear optimization. Figure 5 depicts the performance of the training in both NNs. The training is terminated after a maximum number of 1000 iterations in each case, because the stopping criterion based on the validation set is not satisfied yet. Imposing a maximum iteration number represents a kind of regularization in the context of minimization. We observe that the achieved mean squared errors are similar in both NNs. Figure 6 illustrates some samples for the trajectories of the QoI, where the first NN yields the approximations.

Furthermore, we employ the polynomial approximation from Section 2.3 for comparison. Let s_d be the number of basis polynomials up to total degree d depending on 10 variables. We discuss the cases $d = 2, 3, 4$. It follows that $s_2 = 66$, $s_3 = 286$, and $s_4 = 1001$ due to Eq. (11). We use only the training samples [Eq. (21)] in the least squares problem. Hence the validation set becomes just an additional test set. In the case of $d = 4$, the number s_4 of DOFs is larger than the number $k = 1000$ of training samples. Thus we extend the training set by just one sample once. It follows that the polynomial regression changes into a polynomial interpolation in the case of $d = 4$.

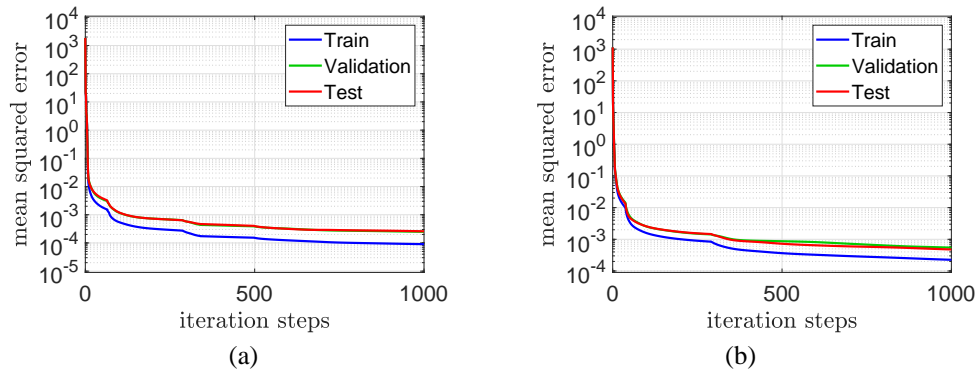


FIG. 5: Performance in training of NNs with two hidden layers (a) and three hidden layers (b) in transistor amplifier example

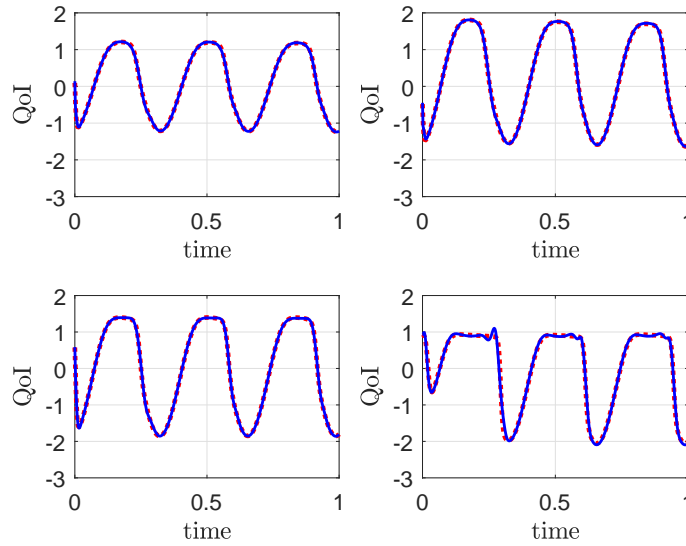


FIG. 6: Trajectories of QoI for four different parameter samples: solution of IVPs (dotted line) and approximation from NN (solid line) in transistor amplifier example (time interval $[0, 0.03]$ is standardized to $[0, 1]$)

Table 1 demonstrates the statistics of the errors [Eq. (24)] based on the discrete \mathcal{L}^1 -norms. The accuracy of the two NNs coincides. The errors of the polynomial regression with degree 2 are worse. Yet the accuracy of the polynomial approximation improves for increasing total degree in the case of the training set. The polynomial fit of degree 4 produces the same errors as the NNs for the training set, whereas the error of the polynomial approximation is much larger for the validation set as well as the test set. This typical phenomenon is an oversampling with respect to the training samples. In the training routines of the NNs, the consideration of the validation set would stop the training if an overfitting is detected. However, this stopping criterion does not occur in the fitting of our two NNs, because the training terminates after the maximum number of iteration steps. Thus an important observation is that the training of NNs omits overfitting without termination in this example. Moreover, the polynomial interpolation ($d = 4$) features no approximation error in the training set. Hence this mean value is dominated by the approximation error of the reduced basis from the POD approach, which is the error part (2)

TABLE 1: Statistics of errors in approximations by NNs and polynomials in transistor amplifier example

		NN 2 layers	NN 3 layers	Polynomial degree 2	Polynomial degree 3	Polynomial degree 4
Mean	Training set	0.0223	0.0223	0.0443	0.0302	0.0223
	Validation set	0.0223	0.0224	0.0465	0.0364	0.6611
	Test set	0.0223	0.0224	0.0462	0.0367	0.6719
Standart deviation	Training set	0.0112	0.0112	0.0217	0.0140	0.0112
	Validation set	0.0099	0.0099	0.0195	0.0165	0.5204
	Test set	0.0109	0.0108	0.0225	0.0194	0.5021

in Section 3.2. We conclude that the approximation errors of the NNs are also negligible, since their mean errors of all sets nearly coincide with the mean error of the polynomial approach for $d = 4$ in the training set.

Since the validation set is not used in the polynomial regression, we perform an additional numerical experiment. The polynomials are fitted to the data of the union of training set and validation set (2000 samples). Table 2 shows the statistics of errors. The results are similar to the previous polynomial approach. The effect of overfitting is reduced in the case of degree 4. However, the overfitting is still present and thus the errors are worse for the test set in comparison to the NN models. A polynomial approximation of degree 5 would include 3003 basis polynomials, which is a larger number than the total sample size.

We comment on the computing times. The polynomial regression with data size 1000/1001 required in seconds: 0.05 for degree 2, 0.21 for degree 3, and 0.86 for degree 4. Thus the polynomial approximation is cheap. In contrast, the training of the NN with two layers ran about 41 minutes. There is some potential to reduce the computation work in the training. On the one hand, the number of iterations can be reduced. On the other hand, there are much cheaper iteration techniques in comparison to the Levenberg-Marquardt method. However, the alternative techniques yield worse accuracy in this example.

We compute the total effect sensitivity indices [Eq. (18)] for the varying physical parameters. The approximations \tilde{a} are evaluated on the grid of the Stroud-5 cubature (Stroud, 1971), which is exact for polynomials up to total degree 5. These evaluations yield approximate sensitivities [Eq. (18)] using a non-intrusive method as given in Pulch et al. (2015). Figure 7(a) shows the

TABLE 2: Statistics of errors in approximations by polynomials with joined set (union of training set and validation set) for fitting in transistor amplifier example

		Polynomial degree 2	Polynomial degree 3	Polynomial degree 4
Mean	Joined set	0.0443	0.0306	0.0249
	Test set	0.0450	0.0334	0.0350
Standart deviation	Joined set	0.0207	0.0142	0.0105
	Test set	0.0223	0.0180	0.0208

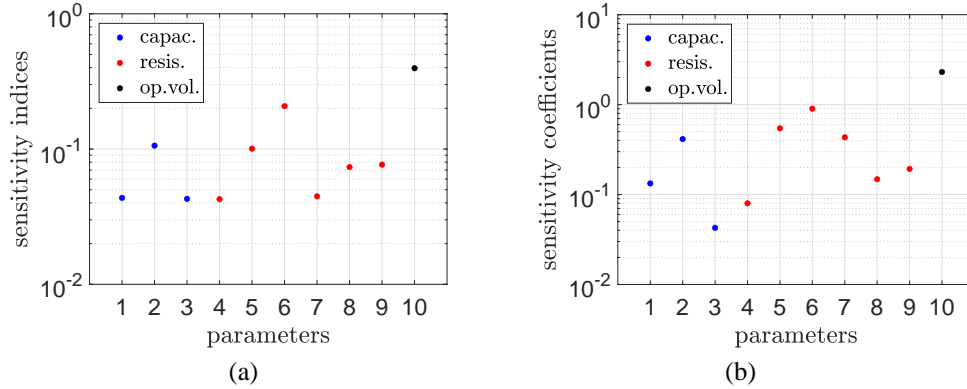


FIG. 7: Sensitivities for different physical parameters (capacitances 1-3, resistances 4-9, operating voltage 10) in transistor amplifier example: (a) total effect sensitivity indices [Eq. (18)] and (b) sensitivity coefficients [Eq. (25)] obtained by NN in semilogarithmic scale

sensitivity indices [Eq. (18)]. We emphasize that these values are computed directly from the mapping [Eq. (8)] without an approximation by NNs or polynomial regression. Alternatively, we consider the trained NN with two hidden layers to obtain the sensitivity coefficients [Eq. (25)] depicted in Fig. 7(b). We recognize a good agreement for the relative positions of the sensitivity indicators in the two concepts. In particular, the ranking of the parameters mostly coincides. The operating voltage exhibits the largest influence. Furthermore, we train an extended NN [Eq. (26)] with three hidden layers of sizes $\hat{N}_1 = 10$ and $\hat{N}_2 = \hat{N}_3 = 30$. Its sensitivity coefficients [Eq. (25)] are shown in Fig. 8. The results are similar to the previous NN.

Finally, we reproduce the total effect sensitivity indices based on the approximate mappings. In Eq. (18), the exact coefficients \hat{a}_i are substituted by the approximations \tilde{a}_i . On the one hand, the trained NN with two layers is used, where the NN model is evaluated at the nodes of the Stroud-5 quadrature. On the other hand, the polynomial regression of degree 2, which fits to 1000 samples of the training set, yields the approximation [Eq. (9)], and the coefficients c_i are directly inserted in Eq. (16) to obtain approximations of the partial variances. Figure 9 illustrates the resulting sensitivity indices. We observe that the outcomes of both methods agree roughly.

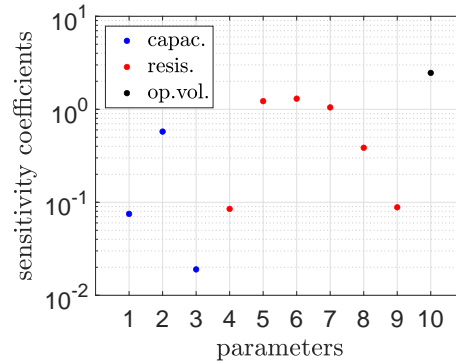


FIG. 8: Sensitivity coefficients [Eq. (25)] for different physical parameters (capacitances 1-3, resistances 4-9, operating voltage 10) using an extended NN in transistor amplifier example

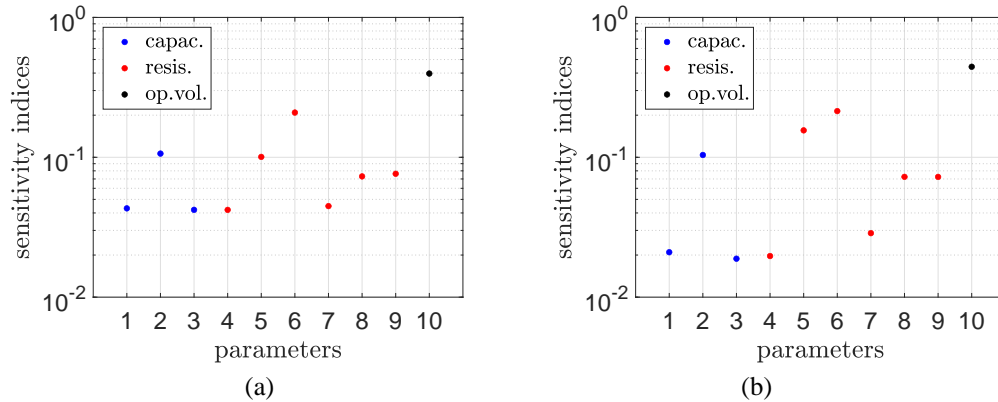


FIG. 9: Total effect sensitivity indices [Eq. (18)] for different physical parameters (capacitances 1-3, resistances 4-9, operating voltage 10) computed using NN (a) and polynomial approximation (b) both in semilogarithmic scale for transistor amplifier

The NN replicates the results in Fig. 7(a). The polynomial regression produces slightly different values for small sensitivity values. This behavior reflects that the approximation of the NN is more accurate.

4.2 Example: Schmitt Trigger

The electric circuit of a Schmitt trigger is illustrated in Fig. 10. There are five resistances, a capacitance, and two bipolar transistors. This circuit acts as an analog-digital converter. A mathematical model is presented in Kampowsky et al. (1992) that consists of five DAEs for five unknown node voltages. The mass matrix and the right-hand side of the system [Eq. (1)] are

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & C & 0 & -C & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -C & 0 & C & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad f = \begin{pmatrix} -\frac{u_1}{R_1} - (1 - \gamma)w(u_1 - u_3) \\ -\frac{u_2}{R_2} - \frac{u_2 - u_4}{R_4} - \gamma w(u_1 - u_3) \\ -w(u_1 - u_3) + \frac{u_3}{R_3} - w(u_4 - u_3) \\ -\frac{u_4 - u_2}{R_4} - (1 - \gamma)w(u_4 - u_3) \\ -\frac{u_5}{R_5} - \gamma w(u_4 - u_3) \end{pmatrix} + \begin{pmatrix} \frac{u_{in}}{R_1} \\ \frac{u_{op}}{R_2} \\ 0 \\ 0 \\ \frac{u_{op}}{R_5} \end{pmatrix}.$$

The current-voltage relation of the bipolar transistors is given by Eq. (27) again using the same physical parameters. The differential index of the system is one. We employ a harmonic oscillation [Eq. (28)] with period T and amplitude $A = 5$ as input voltage u_{in} . The QoI is the output voltage $u_{out} = u_5$.

We arrange a parameter variation in the capacitance, the five resistances, the operating voltage, and the period of the input oscillation. The mean values of the parameters read as capacitance $C = 4 \cdot 10^{-11}$; resistances $R_1 = 200$, $R_2 = 1600$, $R_3 = 100$, $R_4 = 3200$, and $R_5 = 1600$; operating voltage $u_{op} = 0.2$; and period $T = 0.002$. Ranges of 20% around these mean values are used for each parameter, except for the period varying just 5%. Thus the parameter domain is a cuboid $\Pi \subset \mathbb{R}^8$.

In Eqs. (21)–(23), we incorporate $k = k' = k'' = 500$ samples using a pseudo random number generator. The number of equidistant time points is $m = 1000$ now. Again the NDF schemes yield the numerical solutions of the IVPs within the time interval $[t_0, t_{end}] = [0, 0.006]$. Starting

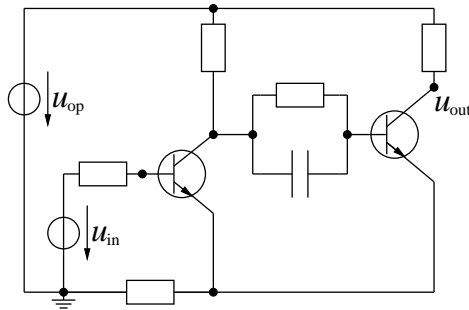


FIG. 10: Electric circuit of a Schmitt trigger

values for the initial values are always zero, whereas a consistent initial value [Eq. (2)] is determined for each parameter tuple by the method. The local error control uses relative tolerance $\varepsilon_r = 10^{-3}$ and absolute tolerance $\varepsilon_a = 10^{-5}$. Figure 11(a) depicts the trajectory associated with the mean value of the parameters. Although the solution of the DAE system is continuous, the output voltage exhibits fast transitions, which behave like jumps. The sinusoidal input signal [Eq. (28)] is transformed into a digital output signal. Figure 11(b) illustrates the trajectories for several parameter samples. The variation of the period shifts the locations of the jumps.

We perform the POD approach for the training samples. Figure 12 displays the singular values of the decomposition [Eq. (4)]. The decay of the singular values is slower in comparison to the previous example shown in Fig. 4, since the trajectories vary to a higher extent. We set the reduced dimension to $r = 11$, which is the smallest number satisfying Eq. (7) with the threshold $\delta = 0.99$.

We train two NNs of the same sizes as in the example of Section 4.1. A conjugate-gradient method (Du and Swamy, 2014, p. 136) solves the nonlinear optimization problem iteratively. On the one hand, a single iteration step is much cheaper in comparison to the Levenberg-Marquardt method. On the other hand, more iteration steps are required in total. Yet the total computation work is significantly lower now. Figure 13 shows the training procedure of the two NNs. The iteration is terminated at the 5189th step and the 10,258th step, respectively, since the performance function does not decrease any more on the validation set in both cases. In the training, the computing times were 22.2 seconds and 43.8 seconds, respectively.

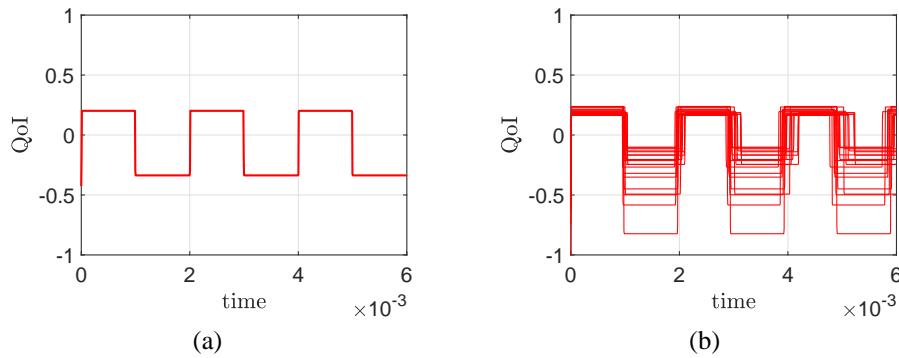


FIG. 11: Trajectory of QoI for mean value of parameters (a) and twenty trajectories of QoI for different parameter samples (b) produced by Schmitt trigger circuit

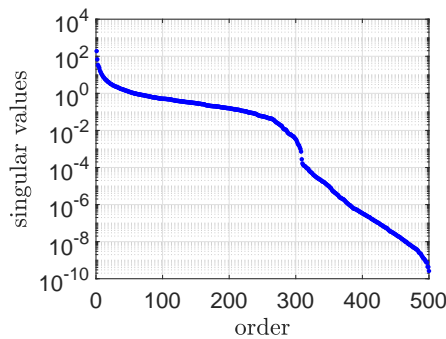


FIG. 12: Singular values from POD for matrix including the samples of the QoI in Schmitt trigger example

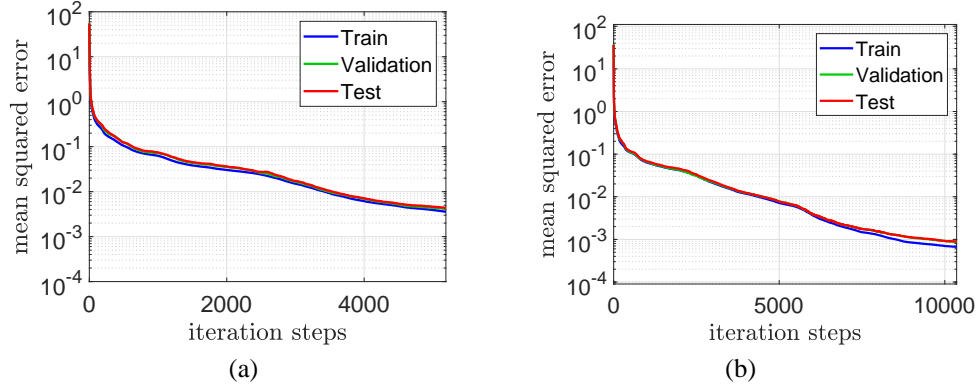


FIG. 13: Performance in training of NNs with two hidden layers (a) and three hidden layers (b) in Schmitt trigger example

We use the first NN to approximate the trajectories of the QoI. Figure 14 demonstrates the approximations together with the original trajectories of the time integration. We observe a good agreement of the amplitudes and a good localization of the jumps. However, incorrect oscillations occur close to the jumps, which are caused by the reduced basis in the POD approximation.

We check the NNs against the polynomial regression from Section 2.3. The approximations of total degree $d = 2, 3, 4$ are computed, where the number of basis polynomials is $s_2 = 45$, $s_3 = 165$, and $s_4 = 495$, respectively. Hence we solve linear least squares problems using the training set [Eq. (21)]. The case of $d = 4$ nearly coincides with a polynomial interpolation of the training set due to $s_4 \approx k$. Table 3 contains the statistics of the discrete \mathcal{L}^1 -errors [Eq. (24)]. We observe the same behavior as in the example of Section 4.1. Again the NNs are better than a straightforward polynomial approximation.

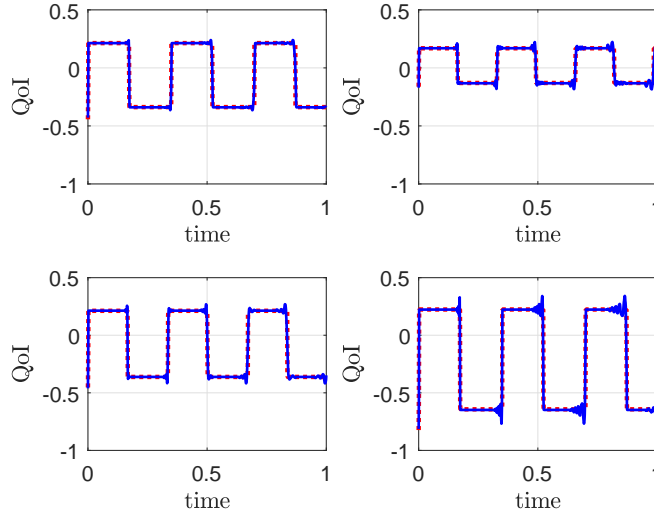


FIG. 14: Trajectories of QoI for four different parameter samples: solution of IVPs (dotted line) and approximation from NN (solid line) in Schmitt trigger example (time interval $[0, 0.006]$ is standardized to $[0, 1]$)

TABLE 3: Statistics of errors in approximations by NNs and polynomials in Schmitt trigger example

		NN 2 layers	NN 3 layers	Polynomial degree 2	Polynomial degree 3	Polynomial degree 4
Mean	Training set	0.0089	0.0089	0.0246	0.0180	0.0089
	Validation set	0.0092	0.0090	0.0269	0.0254	0.1896
	Test set	0.0093	0.0092	0.0276	0.0263	0.1792
Standart deviation	Training set	0.0029	0.0030	0.0087	0.0060	0.0030
	Validation set	0.0034	0.0033	0.0093	0.0105	0.1463
	Test set	0.0034	0.0035	0.0095	0.0107	0.1190

Again, we also fit the polynomials to the data of the union of training set and validation set (1000 samples). Table 4 depicts the statistical errors. An overfitting occurs again and thus the approximation of the NNs is more accurate. A polynomial approximation of degree 5 would include 1287 basis polynomials, where the number of degrees of freedom is larger than the size of the joined sample set.

Finally, we compute the total effect sensitivity indices [Eq. (18)] of the exact mapping as well as the sensitivity coefficients [Eq. (25)] of the first NN shown in Fig. 15. The variance-based concept identifies the operating voltage as most important, whereas the period dominates in the weight-based approach. Although the varying period changes the positions of the jumps, the (Euclidean) norm of the basis coefficients in Eq. (18) remains nearly the same. The total effect sensitivity indices of the first two parameters are tiny. A more detailed investigation confirms that these two parameters hardly influence the QoI, while they have some impact on other variables of the solution. However, the sensitivity coefficients [Eq. (25)] from the NN are not small for the two parameters. Thus we construct an extended NN [Eq. (26)] with three hidden layers of sizes $\hat{N}_1 = 8$ and $\hat{N}_2 = \hat{N}_3 = 30$. Figure 16 displays the computed sensitivity coefficients [Eq. (25)]. Now the first two parameters are correctly detected as insignificant variables with respect to the observed QoI.

5. CONCLUSIONS

We approximated trajectories of a QoI, which is the output of a nonlinear dynamical system. Artificial NNs were fitted to data obtained by a POD. The numerical computations of test examples demonstrate that neural networks with relatively low numbers of hidden layers are already sufficiently accurate. Moreover, the NNs are superior in comparison to a multivariate polynomial approximation by regression. In addition, a variance-based sensitivity analysis of the input–output

TABLE 4: Statistics of errors in approximations by polynomials with joined set (union of training set and validation set) for fitting in Schmitt trigger example

		Polynomial degree 2	Polynomial degree 3	Polynomial degree 4
Mean	Joined set	0.0250	0.0192	0.0140
	Test set	0.0268	0.0228	0.0258
Standart deviation	Joined set	0.0087	0.0067	0.0046
	Test set	0.0091	0.0084	0.0124

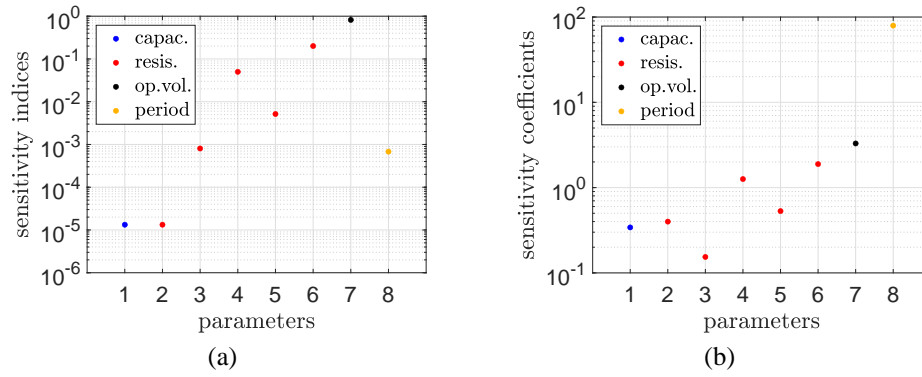


FIG. 15: Sensitivities for different physical parameters (capacitance 1, resistances 2-6, operating voltage 7, period 8) in Schmitt trigger example: (a) total effect sensitivity indices [Eq. (18)] and (b) sensitivity coefficients [Eq. (25)] obtained by NN in semilogarithmic scale

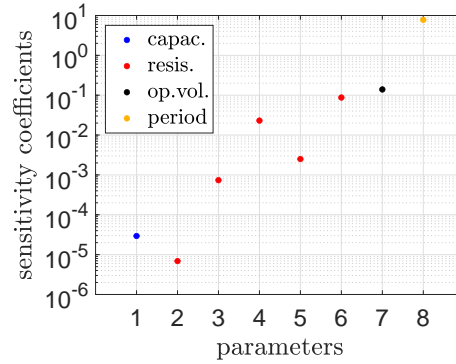


FIG. 16: Sensitivity coefficients [Eq. (25)] for different physical parameters (capacitance 1, resistances 2-6, operating voltage 7, period 8) using an extended NN in Schmitt trigger example

mapping was considered. We introduced an alternative concept based on the weights in a trained NN. The variance-based technique already becomes cheap when an NN is applied to approximate the mapping. Alternatively, the sensitivity concept using the weights does not significantly save computational effort but it provides some insight into the input–output relation of an NN.

REFERENCES

- Antoulas, A.C., *Approximation of Large-Scale Dynamical Systems*, Philadelphia: SIAM, 2005.
- Benner, P., Gugercin, S., and Willcox, K., A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems, *SIAM Review*, vol. **57**, no. 4, pp. 483–531, 2015.
- Du, K. and Swamy, M., *Neural Networks and Statistical Learning*, Berlin: Springer, 2014.
- Genzel, M. and Kutyniok, G., Artificial Neural Networks, *GAMM Rundbrief*, vol. **2**, pp. 12–18, 2019.
- Golub, G. and van Loan, C., *Matrix Computations*, Baltimore: Johns Hopkins University Press, 1996.
- Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, Cambridge, MA: MIT Press, 2017.
- Hairer, E. and Wanner, G., *Solving Ordinary Differential Equations. Vol. 2: Stiff and Differential-Algebraic Equations*, Berlin: Springer, 1996.

- Hesthaven, J. and Ubbiali, S., Non-Intrusive Reduced Order Modeling of Nonlinear Problems Using Neural Networks, *J. Comput. Phys.*, vol. **363**, pp. 55–78, 2018.
- Kampowsky, W., Rentrop, P., and Schmidt, W., Classification and Numerical Solution of Electric Circuits, *Surv. Math. Ind.*, vol. **2**, pp. 23–65, 1992.
- Kunisch, K. and Volkwein, S., Galerkin Proper Orthogonal Decomposition Methods for Parabolic Problems, *Numer. Math.*, vol. **90**, pp. 117–148, 2001.
- Mifsud, M., MacManus, D., and Shaw, S., A Variable-Fidelity Aerodynamic Model Using Proper Orthogonal Decomposition, *J. Numer. Meth. Fluids*, vol. **82**, pp. 646–663, 2016.
- Montavon, G., Samek, W., and Muller, K., Methods for Interpreting and Understanding Deep Neural Networks, *Digital Signal Process.*, vol. **73**, pp. 1–15, 2018.
- Pulch, R., Model Order Reduction for Random Nonlinear Dynamical Systems and Low-Dimensional Representations for Their Quantities of Interest, *Math. Comput. Simulat.*, vol. **166**, pp. 76–92, 2019.
- Pulch, R. and Narayan, A., Sensitivity Analysis of Random Linear Dynamical Systems Using Quadratic Outputs, *J. Comput. Appl. Math.*, 2019. DOI: 10.1016/j.cam.2019.112491
- Pulch, R., ter Maten, J., and Augustin, F., Sensitivity Analysis and Model Order Reduction for Random Linear Dynamical Systems, *Math. Comput. Simulat.*, vol. **111**, pp. 80–95, 2015.
- Qin, T., Wu, K., and Xiu, D., Data Driven Governing Equations Approximation Using Deep Neural Networks, *J. Comput. Phys.*, vol. **395**, pp. 620–635, 2019.
- Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S., *Global Sensitivity Analysis*, John Wiley and Sons, Ltd., 2008.
- Seber, G. and Lee, A., *Linear Regression Analysis*, Hoboken, NJ: John Wiley and Sons, Inc., 2003.
- Sobol, I., Global Sensitivity Indices for Nonlinear Mathematical Models and Their Monte Carlo Estimates, *Math. Comput. Simulat.*, vol. **55**, pp. 271–280, 2001.
- Sobol, I. and Kucherenko, S., Derivative based Global Sensitivity Measures and Their Link with Global Sensitivity Indices, *Math. Comput. Simulat.*, vol. **79**, pp. 3009–3017, 2009.
- Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis*, Berlin: Springer, 2002.
- Stroud, A., *Approximate Calculation of Multiple Integrals*, Upper Saddle River, NJ: Prentice-Hall, 1971.
- Sudret, B., Global Sensitivity Analysis Using Polynomial Chaos Expansions, *Reliability Eng. Syst. Safety*, vol. **93**, no. 7, pp. 964–979, 2008.
- Wang, X., The Effect of Regularization Coefficient on Polynomial Regression, *J. Phys.: Conf. Ser.*, vol. **1213**, p. 042054, 2019.
- Xiu, D., *Numerical Methods for Stochastic Computations: A Spectral Method Approach*, Princeton, NJ: Princeton University Press, 2010.
- Yu, J. and Hesthaven, J., Flowfield Reconstruction Method Using Artificial Neural Network, *AIAA J.*, vol. **57**, 2019.