Model Order Reduction for Real-Time Hybrid Simulation: Comparing Polynomial Chaos Expansion and Neural Network methods

N. Tsokanas[a], T. Simpson[a], R. Pastorino[b], E. Chatzi[a], B. Stojadinović[a]

[a] *ETH Zurich, IBK, D-BAUG, Stefano-Franscini Platz 5, 8093 Zurich, Switzerland.*
[b] *Siemens Industry Software NV, Interleuvenlaan 68, 3001 Leuven, Belgium.*

Abstract

Hybrid simulation is a method used to investigate the dynamic response of a system subjected to a realistic loading scenario by combining numerical and physical substructures. To ensure high fidelity of the simulation results, it is often necessary to conduct hybrid simulation in real-time. One of the challenges arising in real-time hybrid simulation originates from high-dimensional nonlinear numerical substructures and, in particular, from the computational cost linked to the computation of their dynamic responses with sufficient accuracy. It is often the case that the simulation time-step must be decreased to capture the dynamic behavior of numerical substructures, thus resulting in longer computation. When such computation takes longer than the actual simulation time, time delays are introduced and the simulation timescale becomes distorted. In such a case, the only viable solution for doing hybrid simulation in real-time is to reduce the order of such complex numerical substructures.

In this study, a model order reduction framework is proposed for real-time hybrid simulation, based on polynomial chaos expansion and feedforward neural networks. A parametric case study encompassing a virtual hybrid model is used to validate the framework. Selected numerical substructures are substituted with their respective reduced-order models. To determine the robustness of the framework, parameter sets are defined to cover the design space of interest. A comparison between the full- and reduced-order hybrid model response is delivered. The attained results demonstrate the performance of the proposed framework.

Keywords: real-time hybrid simulation, model order reduction, polynomial chaos expansion, feedforward neural networks, dynamic response.

1. Introduction

Hybrid simulation (HS) is a method used to investigate the dynamic response of a system subjected to a realistic loading scenario [1]. Across different engineering

disciplines, HS is also known as hardware-in-the-loop (HiL) [2] or system-in-the-loop. It is one of the methods used for Model-based Development [3]. The system-under-consideration consists of multiple individual sub-components, out of which one or more are selected to be tested physically, and are called the physical substructures (PS), while the remaining components, called the numerical substructures (NS), are simulated numerically. The coupled PS and NS constitute the so-called hybrid model. In more detail, the hybrid model dynamic response is evaluated online, based on a step-by-step numerical solution of the governing equations of motion of the hybrid model, combined with experimental responses obtained from the PS. The coupling of NS and PS is achieved by an arrangement of transfer systems, such as servo-controlled motors and actuators. These transfer systems are responsible for synchronizing the dynamic boundary conditions of the substructures at their interfaces throughout the entire HS. The advantage of HS lies in the fact that it can be used to examine the inner workings of a system's sub-component, beyond its linear regime, without testing the entire full-scale system. As a consequence, realistic results from the tested component are obtained at a much lower experimentation cost. The report by Schellenberg and Mahin [4] provides a comprehensive review of HS methods and algorithms.

Frequently, HS is performed on a distorted timescale, e.g. by slowing down the rate of PS testing to accommodate the power or speed of the transfer system in use. These HSs correspond to the so-called pseudodynamic tests [5]. However, in order to increase the fidelity and trust in HS outcomes, HS should be conducted as close to real-time as possible. This is especially important for loading-rate-sensitive substructures. Even though real-time hybrid simulation (RTHS) has many advantages, it comes with numerous challenges. The first challenge originates from the inherent dynamics of the transfer system exciting the PS, as it introduces time delays in the overall HS that alter the dynamic response of the tested hybrid model. This problem is addressed by designing control systems to adequately compensate for these delays. Several control strategies for RTHS have been proposed and are available in the literature. Some of the recently developed novel approaches can be found in [6, 7, 8, 9]. Nonetheless, challenges in RTHS do not arise only from the transfer system of the PS but also from the NS side, such as from the computational power needed to compute the NS responses. In order to capture the dynamic behavior of interest of a high-dimensional nonlinear numerical model, it is often necessary to use small time-steps for the numerical solver. However, the smaller the time-step of the simulation, the larger the computational power needed to compute it. In RTHS, when the required computation time becomes longer than the actual simulation time, time delays are introduced to the hybrid model risking not only to decrease the fidelity of the HS results but also to drive the overall HS into instability. Therefore, in such cases, the only viable solution is to reduce the order of the NS to be able to conduct the HS in real-time.

Several model order reduction (MOR) techniques can be found in the literature. Proper orthogonal decomposition (POD) [10, 11, 12], also known as principal component analysis (PCA), and its discrete version, namely the singular value decomposition (SVD), are mostly combined with the method of snapshots [13, 14] and have been extensively used for extraction of mode shapes from high-dimensional systems, among other techniques. A technique similar to POD, also using the method of snapshots but following a data-based instead of a model-based approach, is the dynamic mode

decomposition (DMD) [15]. In particular, the main difference between POD and DMD is that the former relies upon a time-averaged spatial correlation matrix while the latter represents the temporal dynamics with high-degree polynomials [16]. An advancement of POD was also proposed based on the use of separated representations, the so-called proper generalized decomposition (PGD) [17]. More recently, the component-mode synthesis (CMS) approach [18, 19] has been used for non classically damped linear systems [20] as well as in HS as a method to reduce the order of the comprising NS and PS [21]. Furthermore, a quadratic manifold [22] and a non-intrusive [23] approach were proposed for MOR of nonlinear structural systems. MOR techniques have also been used for Bayesian finite element model updating [24] and for reliability-based design problems [25].

In this study, a MOR framework originating from a data-driven regression is proposed to reduce the order of high-dimensional nonlinear NS in HS. Two different methodologies for MOR are addressed; a polynomial chaos expansion (PCE) and a feedforward neural network (FFNN). In both cases we treat the NS, whose order we want to reduce, as a *black box*, meaning that no prior knowledge of its full-order model (FOM) dynamics is required. Instead, we map the outputs of the respective NS to its inputs by simpler-to-evaluate functions than those of the original FOM. The goal of the MOR framework is to adequately capture the dynamic behavior of high-dimensional NS at a much lower computational cost and in shorter computation time, and thus enable real-time HS. A parametric case study encompassing a virtual hybrid model is employed to validate the proposed MOR framework. Selected NS are replaced with their respective FFNN- and PCE-based reduced-order models, while specific parameters of the hybrid model are varied to investigate the framework's robustness to different hybrid model configurations. The corresponding reduced-order hybrid model responses are compared to the ones from the FOM. Results demonstrate the effectiveness of the proposed MOR framework.

The paper is organized as follows. Section 2 introduces the MOR framework describing the PCE and FFNN. Section 3 presents the case study with the virtual hybrid model used to validate the framework. Section 4 discusses the obtained results and Section 5 presents the overall conclusions of this study.

2. Polynomial chaos expansion and neural networks as model order reduction techniques

Considering an input vector $\boldsymbol{X}$ and a computational model $\boldsymbol{Y} = \mathcal{M}(\boldsymbol{X})$, data-driven regression algorithms formulate a map $\mathcal{M} : \boldsymbol{X} \mapsto \boldsymbol{Y}$ based on an obtained sample set of input points $\boldsymbol{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right\}^T$ and of the respective output values, i.e., model evaluations, $\boldsymbol{Y} = \left\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\right\}^T$. The set of $\boldsymbol{X}, \boldsymbol{Y}$ realizations corresponds to the so-called experimental design (ED). Regression techniques encompass linear regression, kernel methods, neural networks and graphical models among others. An overview of these techniques can be found in [26, 27]. As mentioned above, the proposed MOR framework of this study explores use of PCE and FFNN to this end, with these methods elaborated in what follows.

2.1. Polynomial chaos expansion

PCE is a well-known uncertainty quantification spectral method used to substitute the dynamics of an expensive-to-compute numerical model with an inexpensive-to-compute surrogate (a.k.a. metamodel), representing the outputs of the model by a polynomial function of its inputs [28, 29]. It is proven to be a powerful surrogate technique used in a wide variety of engineering contexts to replicate the dynamic response of complex high-dimensional models [30, 31, 32, 33]. Hence, it can be seen as a promising technique for MOR of NS in HS. Furthermore, to the best of the authors' knowledge, such use of PCE within the HS context is new.

In more detail, given a random input vector $\boldsymbol{X} \in \mathbb{R}^M$ with statistically independent components expressed by the joint probability density function (PDF) $f_{\boldsymbol{X}}$ and a finite variance computational model $Y = \mathcal{M}(\boldsymbol{X})$, with $Y \in \mathbb{R}$ such that $\mathbb{E}[Y^2] = \int_{\mathcal{D}_{\boldsymbol{X}}} \mathcal{M}^2(\boldsymbol{x}) f_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x} < \infty$, the PCE of $\mathcal{M}(\boldsymbol{X})$ follows:

$$Y = \mathcal{M}(\boldsymbol{X}) \approx \mathcal{M}^{PCE}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^M} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}). \tag{1}$$

The PCE function is built on the $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ multivariate orthonormal polynomial basis with respect to the input vector $f_{\boldsymbol{X}}$. The degree of the $\Psi_{\boldsymbol{\alpha}}$ polynomials components is identified by the $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M), \boldsymbol{\alpha} \in \mathbb{N}^M$ multi-index for each of the input variables, while $y_{\boldsymbol{\alpha}}$ corresponds to the polynomial coefficients.

2.1.1. Polynomial basis

The multivariate polynomials are constructed as a tensor product of their univariate orthonormal polynomials $\phi_k^{(i)}(x_i)$ such that $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) = \prod_{i=1}^{M} \phi_{\alpha_i}^{(i)}(x_i)$. The latter meet the $\left\langle \phi_j^{(i)}(x_i), \phi_k^{(i)}(x_i) \right\rangle = \int_{\mathcal{D}_{X_i}} \phi_j^{(i)}(x_i) \phi_k^{(i)}(x_i) f_{X_i}(x_i)\, dx_i = \delta_{jk}$ orthonormality criteria, where i corresponds to the input variable, j and k to the polynomial degree, $f_{X_i}(x_i)$ to the i^{th}-input marginal PDF and δ_{jk} to the Kronecker symbol. The selection of the univariate orthonormal polynomial families depends on the marginal PDF of each input variable to which they are orthogonal, e.g., if an input variable follows the uniform/ Gaussian distribution then the Legendre/ Hermite orthogonal polynomial family is used respectively for this specific input variable [29, 34].

2.1.2. Truncation schemes

Once the univariate polynomials families are selected for each input variable, the next step is the construction of the PCE following Eq. (1). However, because the terms of the sum in Eq. (1) are infinite, it is often truncated to a finite number of terms for practical reasons. Hence the truncated basis is defined as $\mathcal{A} \subset \mathbb{N}^M$ and the PCE of $\mathcal{M}(\boldsymbol{X})$ admits:

$$\mathcal{M}^{PCE}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}). \tag{2}$$

The performance of PCE is closely connected with the chosen truncation scheme. Underfitting or overfitting is possible when too many terms are discarded or introduced, respectively [32]. In the standard basis truncation scheme [29] the maximum degree

$p \in \mathbb{N}^+$ is defined, such that the degree of each polynomial is capped by this value. Therefore, the standard basis truncation scheme consist of $\binom{M+p}{p}$ elements and follows:

$$\mathcal{A}^{M,p} = \left\{ \boldsymbol{\alpha} \in \mathbb{N}^M : |\boldsymbol{\alpha}| \leq p \right\}, \tag{3}$$

where M corresponds to the number of input variables of $\mathcal{M}(\boldsymbol{X})$ and $|\boldsymbol{\alpha}| = \sum_{i=1}^{M} \alpha_i$ to the total degree of all polynomials in $\Psi_{\boldsymbol{\alpha}}$. Alternate truncation schemes have been developed namely, the maximum interaction and hyperbolic truncation [35], and are used based on the scope of each application. In this study the standard basis truncation scheme of Eq. (3) is employed.

2.1.3. PCE coefficient calculation

Once the truncation scheme is determined, the next step is the calculation of coefficients $y_{\boldsymbol{\alpha}}$ in Eq. (2). Various methods have been developed to do this [34]. In the proposed framework, the least-squares minimization technique [36] is used. This is an non-intrusive technique, meaning that the coefficients are obtained after post-processing the ED points. More specifically, Eq. (2) is reformed as:

$$\mathcal{M}^{PCE}(\boldsymbol{X}) = \sum_{j=0}^{P-1} y_j \Psi_j(\boldsymbol{X}) + \varepsilon_P = \boldsymbol{y}^T \boldsymbol{\Psi}(\boldsymbol{X}) + \varepsilon_P, \tag{4}$$

where $\boldsymbol{\Psi}(\boldsymbol{X}) = \{\Psi_0(\boldsymbol{X}), \ldots, \Psi_{P-1}(\boldsymbol{X})\}^T$ are the multivariate orthonormal polynomials, $\boldsymbol{y} = \{y_0, \ldots, y_{P-1}\}^T$ the PCE coefficients, ε_P the truncation error, and $P = \binom{M+p}{p}$. Then, the PCE coefficients are obtained by:

$$\hat{\boldsymbol{y}} = \arg\min \mathbb{E}\Big[\big(\boldsymbol{y}^T \boldsymbol{\Psi}(\boldsymbol{X}) - \mathcal{M}(\boldsymbol{X}) \big)^2 \Big]. \tag{5}$$

The solution of Eq. (5) is attained by Ordinary Least-Squares (OLS) and follows:

$$\hat{\boldsymbol{y}} = \left(\boldsymbol{A}^T \boldsymbol{A} \right)^{-1} \boldsymbol{A}^T \boldsymbol{Y}, \tag{6}$$

where the experimental matrix is $A_{ij} = \Psi_j(\boldsymbol{x}^{(i)})$ with $i = 1, \ldots, N$ and $j = 0, \ldots, P-1$.

2.2. Feedforward neural network

Neural networks are an extremely versatile group of methods used both for regression and classification problems. The most straightforward and widely used variety of neural networks are the fully connected FFNN. Such a FFNN consists of layers of "neurons", where the value of the neurons in each layer is calculated as a matrix multiplication between a weight matrix and the vector of neuron values at the previous layer, followed by the application of an element-wise nonlinear "activation" function. FFNNs have proven highly capable at approximating a broad range of functions and can be shown to be, in the limit of infinite neurons, universal approximators [37, 38]. As such, they are an obvious candidate for metamodelling, and have been widely demonstrated in many such contexts [39, 40, 41]. Much of the popularity of FFNNs is due to their scalability to very large and high dimensional datasets, and, especially in the scope

of deep learning methods, power for dealing with highly nonlinear relations [42, 43]. This also makes the FFNNs a very attractive option for metamodelling in a hybrid testing context, in which they have been demonstrated in limited number of cases to date [44, 45].

2.2.1. Network Architecture

Much of the power of neural networks comes from the ease with which their architectures can be adapted to deal with systems of different sizes and complexity [46]. The basic architecture of a 3 layer FFNN is demonstrated in Figure 1. Such a network consists of the input layer, where the neurons take the values of the input vector $\boldsymbol{X}$. The number of neurons in this layer is naturally equal to the dimensionality of the input vector. The final layer of the network is the output layer, at which the neuron values are the elements of the output vector $\boldsymbol{Y}$. The number of neurons in the output layer is equal to the dimensionality of the output of the network. Between these two layers lies a hidden layer which can have an arbitrary number of neurons, the selection of which greatly affects the performance of the network. The hidden layer neurons are characterized by the activation values, which are represented in the vector $\boldsymbol{A}$. It is possible and often desirable to use multiple hidden layers between the input and the output layers, creating a so-called deep neural network. Deep neural networks are even more powerful for representing highly complex functions [47], however, only a single hidden layer is used in this work since such a FFNN is deemed adequate.

The single-hidden-layer FFNN diagram in Figure 1 is mathematically described in Eqs. (7) and (8). Eq. (7) represents the calculation of the activation values in the hidden layer of the FFNN, the input vector $\boldsymbol{X}$ is matrix multiplied by the weight matrix $\boldsymbol{W^1}$ and the bias vector $\boldsymbol{b^1}$ is added. The activation vector $\boldsymbol{A}$ is then calculated by the element-wise application of the activation function g^1. In Eq. (8), the output vector of the FFNN is calculated in a similar fashion, via matrix multiplication of the activation vector $\boldsymbol{A}$ with a second weight matrix $\boldsymbol{W^2}$, to which a second bias vector $\boldsymbol{b^2}$ is added before the element-wise application of a second activation function g^2. In these equations, both weight matrices $\boldsymbol{W^1}$ and $\boldsymbol{W^2}$, and both bias vectors, $\boldsymbol{b^1}$ and $\boldsymbol{b^2}$, are populated by parameters which are learned during the training of the FFNN. The activation functions g^1 and g^2, on the other hand, are hyperparameters and must be chosen by the network designer. Common candidates for these activation functions include the $tanh$, $RELU$ and $linear$ activation functions, each with different associated properties [48].

$$\boldsymbol{A} = g^1(\boldsymbol{W^1}\boldsymbol{X} + \boldsymbol{b^1}) \tag{7}$$

$$\boldsymbol{Y} = g^2(\boldsymbol{W^2}\boldsymbol{A} + \boldsymbol{b^2}) \tag{8}$$

2.2.2. Network Training

When using a FFNN to create a metamodel, the first step involves generating a dataset with which to train, validate and test the network. Such a dataset consists of input and output vector pairs (points) for the system being metamodelled. For training a neural network, this dataset is then usually further broken down into training, validation and testing datasets. The training dataset contains those input-output pairs used for optimizing the learned parameters of the system in the weight matrices and

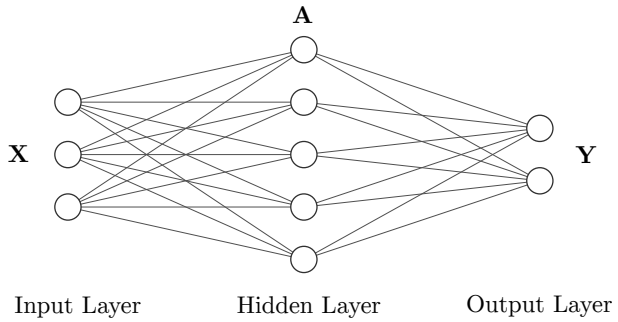


Figure 1. Architecture of a 3 layer FFNN

bias vectors. The validation dataset is a set of held-back input-output pairs which are used to compare FFNNs created with differing hyperparameters, such as hidden layer size or activation functions. This yields FFNNs with better performance on the training dataset. The use of a separate validation dataset informs on the ability of the trained FFNN to generalize, i.g., demonstrate whether the FFNN performance is maintained for input-output pairs outside of the training dataset, as well as indicate whether the FFNN is "overfitting" to the training points [46]. The testing dataset finally, comprises those points on which the final performance of the trained network of the chosen architecture is tested.

In this work, the neural network was created and trained using the Matlab deep learning toolbox [49]. Since, in the context of metamodelling, the problem is one of regression, i.e., predicting a continuous output variable, the loss function of the network to be minimized was selected as the mean squared error value, as demonstrated in Eq. (9). In this equation, the loss value J is calculated as the mean value of the squared error between the true output value Y_i for the training point i and the predicted output value from the network $\hat{Y}_i$ for all N training data points. The network weights were trained using the Bayesian regularization algorithm. The latter is more resistant to overfitting and tends to train networks with greater generalization [50]. A validation dataset was used to select the final architecture of the network as that which gave the best validation performance. The final network architecture selected for this problem was a 3 layer network with a single hidden layer. The activation functions for the hidden and output layers are $tanh$ and $linear$ functions, respectively.

$$J = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{9}$$

3. Case study

Given that the focus of this work is a MOR framework for RTHS, the case study employs a virtual hybrid model to conduct virtual RTHSs (vRTHSs). All of the components of the virtual hybrid model are implemented numerically. Thus, virtual PSs (vPSs) are used in place of physical specimens. In this regard, without loss of

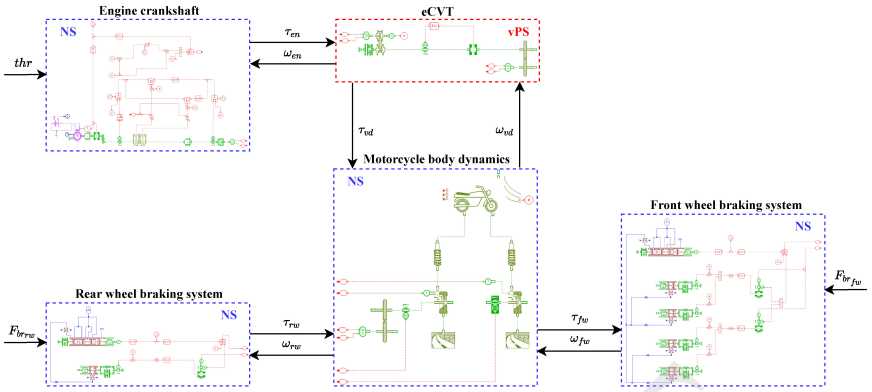


Figure 2. Hybrid model block diagram.

generality and for the sake of clarity, transfer systems that would be required between NS and PS in a physical HS are omitted from this case study.

The virtual hybrid model used in this case study represents a prototype motorcycle. This model consists of one vPS, the electronically-controlled continuously variable transmission (eCVT) of the motorcycle, and four NS: i) the engine dynamics model, ii) the motorcycle body dynamics model including the tires and the road, iii) the rear wheel braking systems, and iv) the front wheel braking system. The interconnections between NS and the vPS are illustrated in Figure 2. All substructures of the hybrid model are developed in the Simcenter Amesim software and are briefly introduced below. For a more detailed description of the aforementioned models, as well as the underlying equations governing each substructure, the reader is encouraged to consult [51].

In particular, the eCVT vPS consist of a multi-input-multi-output (MIMO) model with two sets of inputs/outputs. The first set is connected to the engine dynamics model NS and the second to the motorcycle body dynamics NS. The latter connection corresponds to the transmission output shaft of the motorcycle. The engine dynamics model NS aims to simulate the behavior of the motorcycle combustion engine. It is a multi-input-single-output (MISO) model, the inputs being the angular velocity of the engine crankshaft ω_{en} and the throttle level thr, and the output being the torque of the engine crankshaft τ_{en}. The parameters of this model are the engine moment of inertia and coefficient of viscous friction. The motorcycle body NS addresses the inner body dynamics of the motorcycle along with the dynamics of the suspension and the tires as well as the road profile and the environmental driving conditions. The parameters of this model include the motorcycle mass, as well as damping and stiffness of the front and rear tires and suspensions. The motorcycle body NS is a MIMO model with 3 sets of inputs/outputs. The first set is connected to the eCVT vPS with the input and the output being the torque τ_{vd} and the angular velocity ω_{vd} of the transmission output shaft, respectively. The second and third sets are connected to the rear and front wheel braking system NS, respectively. Both braking systems are represented as MISO models. The inputs of the rear wheel braking system NS are the angular velocity of the rear wheel ω_{rw} and the applied force in the brake pedal $F_{br_{rw}}$, while the output is the

generated braking torque in the rear wheel τ_{rw}. Similarly, the inputs for the front wheel braking system NS are the angular velocity of the front wheel ω_{fw} and the applied force on the front brake lever $F_{br_{fw}}$, while the output is the braking torque in the front wheel τ_{fw}.

3.1. Motorcycle testing scenario

The performance of the motorcycle prototype is examined by testing its virtual hybrid model under predefined driving, road and wind scenarios. These tests are, in fact, virtual real-time hybrid simulations. A Simcenter real-time simulation platform is used for data exchange between substructures and for running the vRTHSs.

The case study involves a 45 sec long motorcycle driving scenario on a road with a defined profile and in defined wind conditions. The driving scenario is described by the variation of the throttle thr and the braking forces $F_{br_{rw}} = F_{br_{fw}}$ (front and rear wheel braking forces are assumed to be equal) described by Eq. (10) and Eq. (11), respectively, and shown in Figure 3. Note that the braking forces are measured in Newton and the maximum applied throttle is 0.5, corresponding to a half-open throttle. The road profile, i.e. height variation of the road, is defined by the sinusoidal function $h(x)$ in Eq. (12), where x denotes the current position of the motorcycle in meters. The ambient wind velocity is assumed to be zero.

Each case study motorcycle driving test involves "driving" the motorcycle virtual hybrid model following the predefined driving scenario along the given road in the given wind conditions. The position of the motorcycle is computed by solving its equation of motion. In this case study, the RK4 (fourth-order Runge–Kutta) method with a fixed time-step of 0.1 msec was used to numerically solve this equation of motion. In addition to the position of the motorcycle, a number of indicative performance parameters are computed and monitored. These include global response parameters, such as the motorcycle velocity v, as well as the input and output parameters of the NSs, i.e. the angular velocities and torques shown in Figure 2.

$$thr(t) = \begin{cases} 0.125t & , 5 \leq t < 9 \\ 0.5 & , 9 \leq t \leq 13 \\ -0.125t & , 13 < t \leq 17 \\ 0 & , \text{elsewhere} \end{cases} \tag{10}$$

$$F_{br_{rw}}(t) = \begin{cases} 10t & , 20 \leq t < 25 \\ 50 & , 25 \leq t \leq 32 \\ -10t & , 32 < t \leq 37 \\ 0 & , \text{elsewhere} \end{cases} \tag{11}$$

$$h(x) = \begin{cases} 0 & , 0 \leq x \leq 2 \\ 0.02\Big(\cos\left(\frac{2\pi x}{3}\right) - 1\Big) & , \text{elsewhere} \end{cases} \tag{12}$$

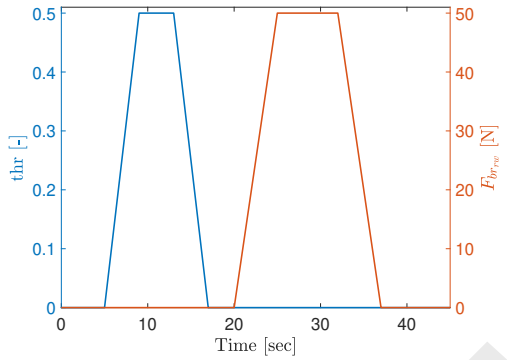


Figure 3. Case study driving scenario.

3.2. Reduced-order numerical substructures

To demonstrate the proposed MOR framework, two of the four NS in the virtual hybrid model of the motorcycle, specifically the rear and front wheel braking systems, were replaced with their respective reduced-order models, applying the PCE and FFNN techniques addressed in Section 2. The braking systems were selected because they include the brake hydraulic systems and friction pads. Adequately emulating the dynamic behavior of such systems using numerical models requires small integration time-steps, making them ideal candidates for MOR. The reduced-order models of the two braking system NSs will be referred to as reduced-order NS (RO-NS) hereafter.

To train the front and rear brake system PCE and FFNN RO-NSs, two sets of 200 points were generated for the input variables listed in Table 1, assuming the inputs are statistically independent, using the Latin hypercube sampling (LHS) method [52]. For each of the 200 generated points, full-order models (FOMs) of the front and rear braking systems were used to (accurately but expensively) compute the resulting front and rear brake torques τ_{rw} and τ_{fw}, respectively. The obtained experimental designs (EDs) for the front and the rear braking systems are:

$$\begin{aligned} \boldsymbol{X_{fw}} &= \left\{F_{br_{fw}}, \omega_{fw}\right\}^T \\ \boldsymbol{Y_{fw}} &= \left\{\tau_{fw}\right\} \end{aligned} \tag{13}$$

$$\begin{aligned} \boldsymbol{X_{rw}} &= \left\{F_{br_{rw}}, \omega_{rw}\right\}^T \\ \boldsymbol{Y_{rw}} &= \left\{\tau_{rw}\right\} \end{aligned} \tag{14}$$

Two types of RO-NSs were developed for each braking system, one featuring a single PCE and the other being a FFNN. The same ED was used to train, validate and test the PCE and the FFNN RO-NSs for each braking system, as discussed in Section 2.

3.3. Virtual hybrid models

Four different virtual hybrid models of the motorcycle were constructed in the case study, differing in the selection of braking system NSs. The base-line virtual hybrid

Input	Distribution	Mean value	Standard deviation	Units
$F_{br_{fw}}$	Uniform	25	14.43	N
ω_{fw}	Uniform	245	141.45	rpm
$F_{br_{rw}}$	Uniform	25	14.43	N
ω_{rw}	Uniform	235	135.68	rpm

Table 1. Marginal PDFs and their characteristics per input variable of the front and rear brake system RO-NSs. Used to construct the EDs of Eq. (14) and Eq. (13).

model features the FOM NSs of both the front and the rear braking systems. The remaining three virtual hybrid models have RO-NSs. In the PCE virtual hybrid model, both braking systems are represented using PCE RO-NSs. Analogously, the FFNN virtual hybrid model uses FFNN RO-NSs. Finally, the mixed virtual hybrid model of the motorcycle uses the FFNN RO-NS for the front brakes and the PCE RO-NSs for the rear brakes.

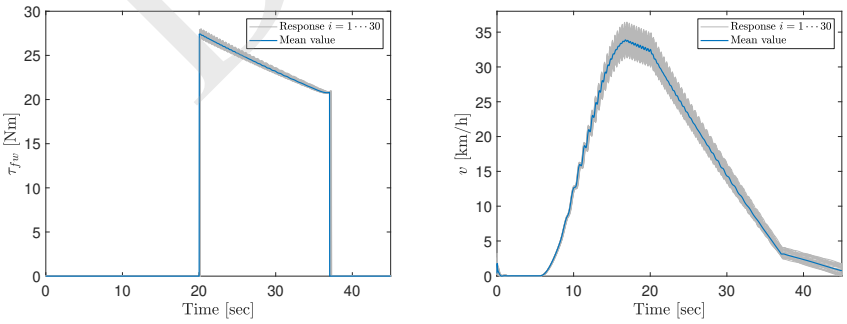
The performance of the thus generated motorcycle virtual hybrid models was tested statistically. To this end, the 11 parameters of the motorcycle model, listed in Table 2, were assigned uniform distributions with parameters identified from [53, 54, 55] to represent realistic variations of these mechanical motorcycle properties. A total of 30 different samples of the 11 motorcycle parameters were generated using the LHS method, essentially resulting in 30 different motorcycle prototypes. These prototypes were modeled using four virtual hybrid models described above. Using the driving scenario described above, a total of 120 motorcycle model response time histories were computed in this case study.

The variability of motorcycle prototype response in the selected driving scenario, computed using the (accurate but expensive) FOM virtual hybrid model, is demonstrated in Figures 4. The 30 time histories of two indicative virtual hybrid model response variables, the front wheel brake torque τ_{fw} and the motorcycle velocity $v = \dot{x}$, are compared to their mean value. Evidently, the 30 generated motorcycle prototypes adequately represent the underlying probability space defined by the parameters of Table 2. All results presented hereafter correspond to the mean values of 30 virtual hybrid model response simulations.

Additionally, the time history responses of Figures 4 can be intuitively interpreted. Figure 4a depicts the time history of the front wheel brake torque τ_{fw}, generated by the employed driving scenario. As can be appreciated from Figure 3 and Eq. (11), the motorcycle brakes are activated between the 20$^{\text{th}}$ and 37$^{\text{th}}$ second of the simulation and therefore the braking torque is non-zero only inside this time interval. Figure 4b illustrates the evolution of the motorcycle's velocity response v using the aforementioned driving scenario. According to Figure 3 and Eq. (10), for the first five seconds of the simulation, no throttle is applied to the motorcycle and hence its velocity is zero. Between the 5$^{\text{th}}$ and 17$^{\text{th}}$ second of the simulation, the driving scenario involves stepping on the throttle and therefore the motorcycle starts to accelerate. As a result, its velocity is increasing. Following Eq. (11), as mentioned before, after the 20$^{\text{th}}$ second, the driving scenario involves pressing on the brakes and thus the motorcycle's velocity begins to decrease and it is brought to almost a full stop in the 45 seconds of the simulation.

Parameter	Description	Distribution	Mean value	Standard deviation	Units
K_{rt}	Vertical stiffness rear tire	Uniform	58570	11714	N/m
Z_{rt}	Vertical damping rear tire	Uniform	11650	3495	Ns/m
K_{ft}	Vertical stiffness front tire	Uniform	25000	5000	N/m
Z_{ft}	Vertical damping front tire	Uniform	2134	640.2	Ns/m
K_{rs}	Stiffness rear suspension	Uniform	125000	25000	N/m
Z_{rs}	Damping rear suspension	Uniform	10000	3000	Ns/m
K_{fs}	Stiffness front suspension	Uniform	19000	3800	N/m
Z_{fs}	Damping front suspension	Uniform	1250	375	Ns/m
M	Motorcycle mass	Uniform	300	6	Kg
J	Engine moment of inertia	Uniform	0.0115	0.0023	Kgm^2
μ	Engine coefficient of viscous friction	Uniform	0.001	0.00005	Nm/(rev/min)

Table 2. Varying parameters of the hybrid model and their characteristics.



(a) Front wheel braking torque.

(b) Motorcycle velocity.

Figure 4. Thirty FOM virtual hybrid model responses time histories and their respective mean values for the front wheel braking torque (a) and the motorcycle velocity (b).

4. Results and discussion

The performance of the motorcycle prototype was evaluated in tests involving a 45 sec long driving scenario. Real-time hybrid simulations lasted approximately 42 sec when the FOM virtual hybrid model was used, while they took approximately 19 sec, 20 sec and 21 sec for the FFNN, PCE and mixed (FFNN&PCE) virtual hybrid models, respectively. Thus, by reducing the order of two braking system NSs, the computation time was halved. This is notable because the computation time of the FOM virtual hybrid model is close to the duration of the driving scenario: it is conceivable that if the driving scenario was more complex, the FOM virtual hybrid model would not complete its calculations in real-time, risking time delays and distortion of the HS timescale. Therefore, reducing the order of the critical NSs using the techniques discussed in Section 2 is justified.

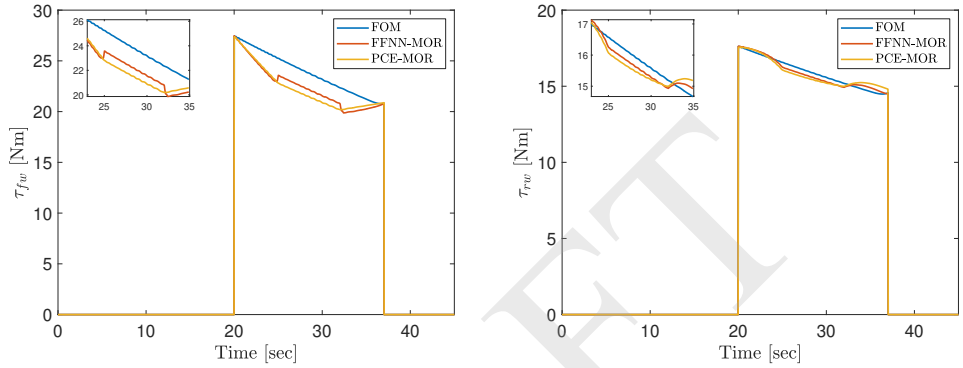
To further assess the performance of the MOR framework, the normalized root mean square error (NRMSE) is used to compare the selected hybrid model dynamic response parameters. Note that the compared responses were computed from the four sets of 30 driving scenario simulations using four virtual hybrid models of the motorcycle prototype described above.

First, the RO-NSs of the front and rear braking systems are compared to the full-order model (FOM) prototype. Recall that the same EDs (Eqs. (13) and (14)), comprising 200 input-output data points generated by the FOM of the front and rear braking systems, were used to train the RO-NSs. To assess the RO-NSs performance, the quantities that characterize the braking systems, namely the rear τ_{rw} and front wheel braking torque τ_{fw}, are compared to the same outputs of the FOM NS. Table 3 displays the NRMSE between FOM NS and RO-NSs for τ_{rw} and τ_{fw}. The NRMSE is small, less than 3% for the real wheel torque and less than 7% for the front wheel, without a clear advantage of the FFNN or the PCE RO-NSs. The NRMSE error difference likely originates with the motorcycle dynamics. Figures 5a-b present the mean time history response of the braking torques obtained from the FOM, FFNN and PCE virtual hybrid models. The comparison indicates that the responses from the reduced-order virtual hybrid models are similar and only slightly different from the response of the FOM virtual hybrid model.

Second, the response of the four virtual hybrid models is compared by examining the NRMSEs and the time histories of the indicative model response quantities, namely the angular velocities and torques that couple the substructures of the hybrid models (ω_{rw}, ω_{fw}, ω_{en}, ω_{vd}, τ_{en} and τ_{vd}) and the motorcycle velocity v. The variables that couple the substructures of the hybrid models were investigated since deviations from their nominal values, would risk the HS outcomes and could drive the HS loop into instability. In addition, for non-virtual RTHS that include real test benches and PS, large deviations could damage the corresponding physical specimen and laboratory equipment. On the contrary, the motorcycle velocity v is chosen to be examined as it corresponds to a global response quantity that informs about the overall dynamic behavior of the motorcycle during the driving scenario. Table 4 provides an overview of the NRMSE and Figures 6, 7 show the response quantity time histories for the four virtual hybrid models. Evidently, the difference between the virtual hybrid models is small: the response time history graphs virtually overlap, while the NRMSEs are all

Rear and front wheel braking torque	NRMSE FOM - FFNN	NRMSE FOM - PCE
τ_{rw}	2.75	2.04
τ_{fw}	6.1	6.54

Table 3. NRMSE of rear and front wheel braking torque between FOM NS and RO-NSs. The values are in percent and correspond to the mean values of the 30 HS evaluations.



(a) Front wheel braking torque.

(b) Rear wheel braking torque.

Figure 5. Front and rear wheel braking torque for the cases of FOM, FFNN-based and PCE-based MOR. Mean values of the 30 HS evaluations.

smaller than 2%. Therefore, both the PCE and the FFNN MOR techniques described in Section 2 produce valid and efficient reduced-order models for RTHS. It is also notable that a combination of RO-NSs developed using different MOR approaches also works, giving a much-needed ability to combine reduced-order models in practice.

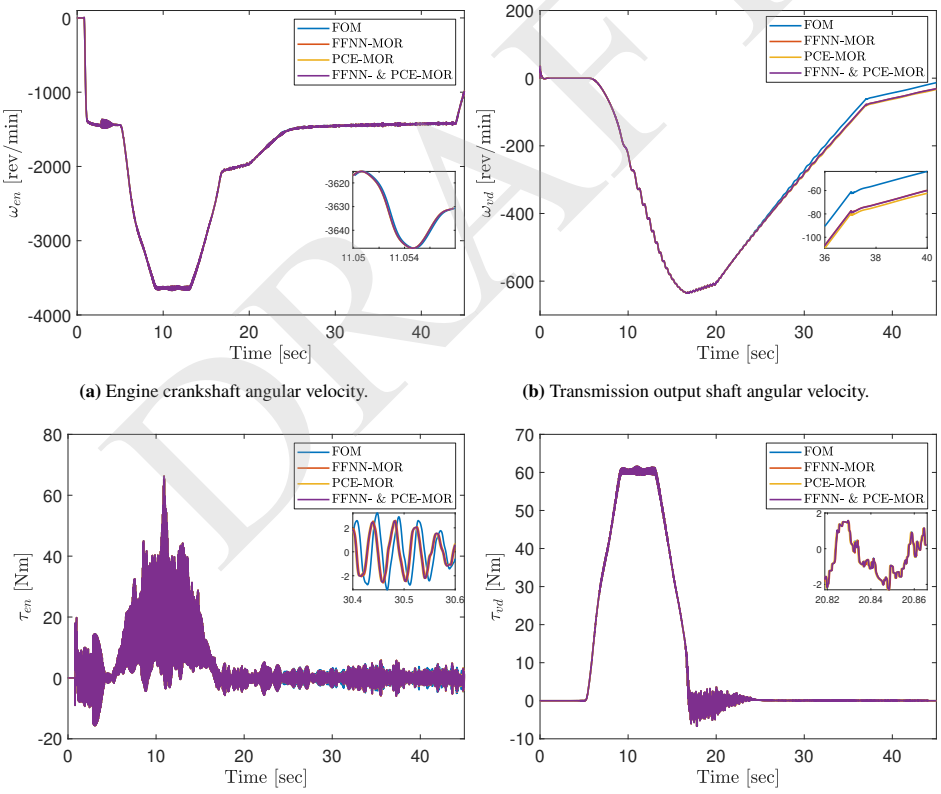
Finally, given that the performance of PCE-based and FFNN-based reduced-order models is practically the same, model designers need to make their modeling choice based on additional consideration. High-dimensional models, with long input and output vectors, are more suitable for FFNN-based MOR, since PCEs may require utilization of sparse schemes and thus incur additional errors. On the other hand, PCE-based MOR offers a distinct advantage in that the PCE coefficients can be used to compute the statistics of the relevant model response quantities, as well as assess their sensitivity to model input quantities via the Sobol' indices approach, at no additional cost [56, 57]. The latter is not a feature of neural networks. Notably, the software for developing both FFNN and PCE reduced-order models exists in the form of Matlab toolboxes [49, 33], making practical implementation of these to MOR approaches relatively easy.

5. Conclusions

In this study, several model order reduction techniques are bench-marked for real-time hybrid simulation. The use of model order reduction techniques in hybrid simu-

Hybrid model responses	NRMSE FOM - FFNN	NRMSE FOM - PCE	NRMSE FOM - FFNN&PCE
ω_{rw}	1.39	1.64	1.43
ω_{fw}	1.38	1.63	1.42
ω_{en}	0.13	0.17	0.16
ω_{vd}	1.39	1.64	1.43
τ_{en}	1.43	1.85	1.68
τ_{vd}	0.09	0.09	0.07
v	1.47	1.74	1.51

Table 4. NRMSE for the mean values of the selected virtual hybrid model response quantities. The values are in percent.



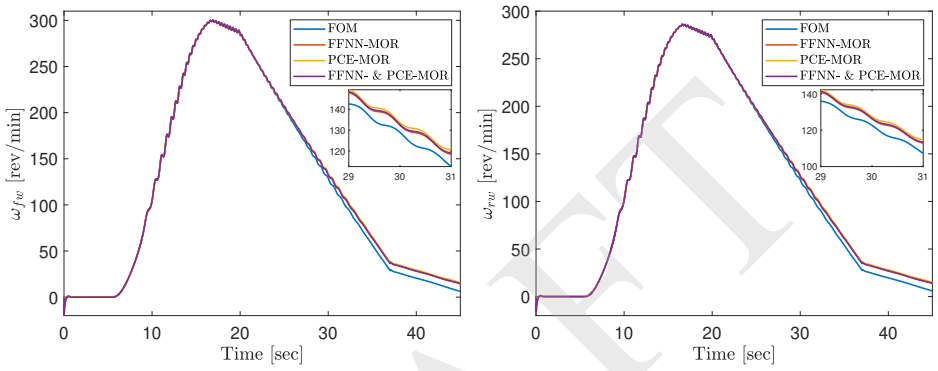
(a) Engine crankshaft angular velocity.

(b) Transmission output shaft angular velocity.

(c) Engine crankshaft torque.

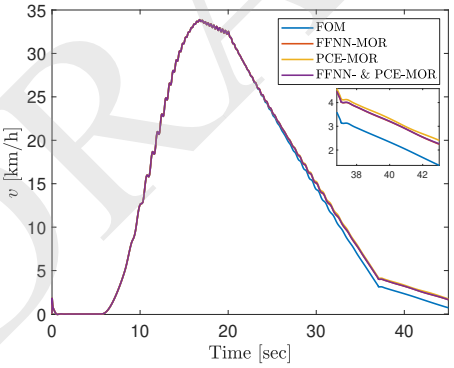
(d) Transmission output shaft torque.

Figure 6. Selected hybrid model responses evaluated in each testing scenario. Mean values of the 30 HS evaluations. Zoomed axes are added to highlight the difference between the shown response time histories.



(a) Front wheel angular velocity.

(b) Rear wheel angular velocity.



(c) Motorcycle velocity.

Figure 7. Selected hybrid model responses evaluated in each testing scenario. Mean values of the 30 HS evaluations. Zoomed axes are added to highlight the difference between the shown response time histories.

lation is especially important for hybrid models encompassing high-dimensional nonlinear numerical substructures, risking distorting the timescale of hybrid simulation by introducing time delays due to the extended computational power needed. The selected model order reduction techniques are based on data-driven regression techniques and particularly on polynomial chaos expansion and feedforward neural networks. A parametric case study consisting of a virtual hybrid model is used to validate the framework. Feedforward neural networks and polynomial chaos expansion surrogates are trained to replicate the dynamic response of selected numerical substructures. Substitution of their respective full-order components forms the reduced-order hybrid model. Specific parameters of the hybrid model are varied and multiple hybrid simulation evaluations are conducted to determine the robustness of the model order reduction framework. Comparisons with the full-order hybrid model responses are made to assess whether the reduced-order hybrid models can accurately replicate the initial dynamic responses while the corresponding errors are proven to be negligibly small. Results confirm the validity, benefits and performance of the proposed model order reduction framework. In particular, for the case study presented in this work, model order reduction techniques allowed for a $\approx 52\%$ reduction of the computational cost, for a maximum of 6.54% loss on absolute accuracy versus full-order model. Model order reduction techniques are therefore capable of producing hybrid models for more reliable real-time hybrid simulations without a significant loss of accuracy. The use of model order reduction techniques pushes the boundaries of hybrid simulation further by enabling the use of more complex and computationally involved numerical substructures. Future work will focus on quantifying how much model order reduction techniques contribute to improving real-time hybrid simulation with physical substructures.

6. Data Availability Statement

All data and code that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764547. The sole responsibility of this publication lies with the author(s). The European Union is not responsible for any use that may be made of the information contained herein.

References

[1] C. E. Silva, D. Gomez, A. Maghareh, S. J. Dyke, B. F. Spencer, Benchmark control problem for real-time hybrid simulation, Mechanical Systems and Signal Processing 135 (2020) 106381.

[2] W. Ren, M. Steurer, T. L. Baldwin, Improve the Stability and the Accuracy of Power Hardware-in-the-Loop Simulation by Selecting Appropriate Interface Algorithms, IEEE Transactions on Industry Applications 44 (4) (2008) 1286–1294, conference Name: IEEE Transactions on Industry Applications.

[3] F. L. M. dos Santos, R. Pastorino, B. Peeters, C. Faria, W. Desmet, L. C. Sandoval Góes, H. Van Der Auweraer, Model Based System Testing: Bringing Testing and Simulation Close Together, in: A. Wicks, C. Niezrecki (Eds.), Structural Health Monitoring, Damage Detection & Mechatronics, Volume 7, Conference Proceedings of the Society for Experimental Mechanics Series, Springer International Publishing, Cham, 2016, pp. 91–97.

[4] A. H. Schellenberg, S. A. Mahin, G. L. Fenves, Advanced Implementation of Hybrid Simulation, Tech. Rep. PEER 2009/104, Pacific Earthquake Engineering Research Center, University of California, Berkeley (2009).

[5] C. R. Thewalt, S. A. Mahin, Hybrid solution techniques for generalized pseudodynamic testing, Tech. Rep. UCB/EERC-87/09, Earthquake Engineering Research Center (1987).

[6] N. Tsokanas, D. Wagg, B. Stojadinović, Robust Model Predictive Control for Dynamics Compensation in Real-Time Hybrid Simulation, Frontiers in Built Environment 6 (2020).

[7] T. Simpson, V. K. Dertimanis, E. N. Chatzi, Towards Data-Driven Real-Time Hybrid Simulation: Adaptive Modeling of Control Plants, Frontiers in Built Environment 6 (2020).

[8] A. Maghareh, S. J. Dyke, C. E. Silva, A Self-tuning Robust Control System for nonlinear real-time hybrid simulation, Earthquake Engineering & Structural Dynamics 49 (7) (2020) 695–715.

[9] X. Ning, Z. Wang, C. Wang, B. Wu, Adaptive Feedforward and Feedback Compensation Method for Real-time Hybrid Simulation Based on a Discrete Physical Testing System Model, Journal of Earthquake Engineering (2020).

[10] A. Chatterjee, An introduction to the proper orthogonal decomposition, 2000.

[11] K. Kunisch, S. Volkwein, Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics, SIAM Journal on Numerical Analysis 40 (2) (2002) 492–515.

[12] W. H. A. Schilders, H. A. van der Vorst, J. Rommes (Eds.), Model Order Reduction: Theory, Research Aspects and Applications, Vol. 13 of Mathematics in Industry, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[13] S. Herzog, The large scale structure in the near-wall region of turbulent pipe flow, Ph.D. thesis, Diss, Cornell University, Ithaca, NY (1986).

[14] J. P. Bonnet, D. R. Cole, J. Delville, M. N. Glauser, L. S. Ukeiley, Stochastic estimation and proper orthogonal decomposition: Complementary techniques for identifying structure, Experiments in Fluids 17 (5) (1994) 307–314.

[15] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of Fluid Mechanics 656 (2010) 5–28.

[16] P. J. Schmid, K. E. Meyer, O. Pust, Dynamic Mode Decomposition and Proper Orthogonal Decomposition of flow in a lid-driven cylindrical cavity, in: 8th International Symposium on Particle Image Velocimetry, Melbourne, Victoria, Australia, 2009.

[17] F. Chinesta, P. Ladeveze, E. Cueto, A Short Review on Model Order Reduction Based on Proper Generalized Decomposition, Archives of Computational Methods in Engineering 18 (4) (2011) 395.

[18] R. H. MacNeal, A hybrid method of component mode synthesis, Computers & Structures 1 (4) (1971) 581–601.

[19] R. R. Craig, A. Kurdila, R. R. Craig, Fundamentals of structural dynamics, 2nd Edition, John Wiley, Hoboken, N.J, 2006.

[20] F. M. Gruber, D. J. Rixen, Dual Craig-Bampton component mode synthesis method for model order reduction of nonclassically damped linear systems, Mechanical Systems and Signal Processing 111 (2018) 678–698.

[21] G. Miraglia, M. Petrovic, G. Abbiati, N. Mojsilovic, B. Stojadinovic, A model-order reduction framework for hybrid simulation based on component-mode synthesis, Earthquake Engineering & Structural Dynamics 49 (8) (2020) 737–753.

[22] S. Jain, P. Tiso, J. B. Rutzmoser, D. J. Rixen, A quadratic manifold for model order reduction of nonlinear structural dynamics, Computers & Structures 188 (2017) 80–94.

[23] M. Karamooz Mahdiabadi, P. Tiso, A. Brandt, D. J. Rixen, A non-intrusive model-order reduction of geometrically nonlinear structural dynamics using modal derivatives, Mechanical Systems and Signal Processing 147 (2021) 107126.

[24] H. A. Jensen, E. Millas, D. Kusanovic, C. Papadimitriou, Model-reduction techniques for Bayesian finite element model updating using dynamic response data, Computer Methods in Applied Mechanics and Engineering 279 (2014) 301–324.

[25] H. A. Jensen, A. Muñoz, C. Papadimitriou, E. Millas, Model-reduction techniques for reliability-based design problems of complex structural systems, Reliability Engineering & System Safety 149 (2016) 204–217.

[26] C. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer-Verlag, New York, 2006.

[27] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, Data Mining: Practical Machine Learning Tools and Techniques, 4th Edition, Morgan Kaufmann, Amsterdam, 2016.

[28] R. Ghanem, P. D. Spanos, Polynomial Chaos in Stochastic Finite Elements, Journal of Applied Mechanics 57 (1) (1990) 197–202.

[29] D. Xiu, G. E. Karniadakis, The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations, SIAM Journal on Scientific Computing 24 (2) (2002) 619–644.

[30] G. Abbiati, S. Marelli, N. Tsokanas, B. Sudret, B. Stojadinović, A global sensitivity analysis framework for hybrid simulation, Mechanical Systems and Signal Processing 146 (2021).

[31] M. D. Spiridonakos, E. N. Chatzi, Metamodeling of dynamic nonlinear structural systems through polynomial chaos NARX models, Computers & Structures 157 (2015) 99–113.

[32] E. Torre, S. Marelli, P. Embrechts, B. Sudret, Data-driven polynomial chaos expansion for machine learning regression, Journal of Computational Physics 388 (2019) 601–623.

[33] S. Marelli, B. Sudret, UQLab: A Framework for Uncertainty Quantification in Matlab, in: Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom, 2014, pp. 2554–2563.

[34] S. Marelli, B. Sudret, UQLab user manual – Polynomial chaos expansions, Tech. Rep. # UQLab-V1.3-104, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland (2019).

[35] G. Blatman, B. Sudret, Adaptive sparse polynomial chaos expansion based on least angle regression, Journal of Computational Physics 230 (6) (2011) 2345–2367.

[36] M. Berveiller, B. Sudret, M. Lemaire, Stochastic finite element: a non intrusive approach by regression, European Journal of Computational Mechanics 15 (1-3) (2006) 81–92.

[37] K. Hornik, M. Stinchcombe, W. Halbert, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.

[38] M. Stinchcombe, H. White, Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights, in: 1990 IJCNN International Joint Conference on Neural Networks, Vol. 3, 1990, pp. 7–16.

[39] R. van der Merwe, T. K. Leen, Z. Lu, S. Frolov, A. M. Baptista, Fast neural network surrogates for very high dimensional physics-based models in computational oceanography, Neural Networks 20 (4) (2007) 462–478.

[40] B. Cao, M. Obel, S. Freitag, P. Mark, G. Meschke, Artificial neural network surrogate modelling for real-time predictions and control of building damage during mechanised tunnelling, Advances in Engineering Software 149 (2020) 102869.

[41] T. Wu, K. Ahsan, Modeling hysteretic nonlinear behavior of bridge aerodynamics via cellular automata nested neural network, Journal of Wind Engineering and Industrial Aerodynamics 99 (4) (2011) 378–388.

[42] R. K. Tripathy, I. Bilionis, Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification, Journal of Computational Physics 375 (2018) 565–588.

[43] R. Snaiki, T. Wu, Knowledge-enhanced deep learning for simulation of tropical cyclone boundary-layer winds, Journal of Wind Engineering and Industrial Aerodynamics 194 (2019).

[44] W. Mucha, Application of artificial neural networks in hybrid simulation, Applied Sciences 9 (21) (2019).

[45] E. E. Bas, M. A. Moustafa, Real-Time Hybrid Simulation with Deep Learning Computational Substructures: System Validation Using Linear Specimens, Machine Learning and Knowledge Extraction 2 (4) (2020) 469–489.

[46] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[47] Y. Bengio, Y. Lecun, Scaling learning algorithms towards AI, MIT Press, 2007.

[48] Y. Wang, Y. Li, Y. Song, X. Rong, The influence of the activation function in a convolution neural network model of facial expression recognition, Applied Sciences 10 (5) (2020).

[49] Matlab optimization toolbox, the MathWorks, Natick, MA, USA (2019).

[50] F. Burden, D. Winkler, Bayesian Regularization of Neural Networks, Humana Press, Totowa, NJ, 2009, pp. 23–42.

[51] S. M. Pinheiro, Motorcycle modeling for eCVT-in-the-loop real-time hybrid testing, Master's thesis, University of Porto, Porto, Portugal (2020).

[52] M. D. Mckay, R. J. Beckman, W. J. Conover, A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code, Technometrics 42 (1) (2000) 55–61.

[53] T. Kimishima, T. Nakamura, T. Suzuki, The Effects on Motorcycle Behavior of the Moment of Inertia of the Crankshaft, SAE Transactions 106 (1997) 1993–2003.

[54] M. Tanelli, Modelling, simulation and control of two-wheeled vehicles, John Wiley & Sons, 2014.

[55] R. Sharp, S. Evangelou, D. Limebeer, Advances in the Modelling of Motorcycle Dynamics, Multibody System Dynamics 12 (3) (2004) 251–283.

[56] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliability Engineering & System Safety 93 (7) (2008) 964–979.

[57] L. L. Gratiet, S. Marelli, B. Sudret, Metamodel-based sensitivity analysis: Polynomial chaos expansions and Gaussian processes, arXiv:1606.04273 (2015) 1–37.