applied sciences

MDPI

Article

Application of Artificial Neural Networks in Hybrid Simulation

Waldemar Mucha

Department of Computational Mechanics and Engineering, Silesian University of Technology, 44-100 Gliwice, Poland; waldemar.mucha@polsl.pl; Tel.: +48-32-237-12-04

Received: 24 September 2019; Accepted: 22 October 2019; Published: 23 October 2019

Featured Application: Performing effective and efficient hybrid simulations of mechanical systems in real-time by implementing a model order reduction technique to the computational algorithm. The efficiency of the computations is understood as the ability to test systems of a higher number of degrees of freedom while maintaining high accuracy, without increasing the time step. Effective simulation is a simulation that allows acquiring correct results while maintaining the time regime.

Abstract: Hybrid simulation is a technique for testing mechanical systems. It applies to structures with elements hard or impossible to model numerically. These elements are tested experimentally by straining them by means of actuators, while the rest of the system is simulated numerically using a finite element method (FEM). Data is interchanged between experiment and simulation. The simulation is performed in real-time in order to accurately recreate the dynamic behavior in the experiment. FEM is very computationally demanding, and for systems with a great number of degrees of freedom (DOFs), real-time simulation with small-time steps (ensuring high accuracy) may require powerful computing hardware or may even be impossible. The author proposed to swap the finite element (FE) model with an artificial neural network (ANN) to significantly lower the computational cost of the real-time algorithm. The presented examples proved that the computational cost could be reduced by at least one number of magnitude while maintaining high accuracy of the simulation; however, obtaining appropriate ANN was not trivial and might require many attempts.

Keywords: hybrid simulation; model order reduction; finite element method; real-time; metamodeling; artificial neural network; artificial intelligence

1. Introduction

Hybrid simulation can be defined as a variant of hardware-in-the-loop simulation where the tested object is a deformable mechanical structure. The method consists of performing numerical analysis and experiment at the same time. It applies to mechanical systems that have elements difficult or impossible to model numerically. These elements are in a hybrid simulation of the physical subsystem, are connected to actuators and deformation sensors, and tested in the experiment. The rest of the system is called the analytical subsystem and is modeled numerically using the finite element method (FEM). The main idea is that the physical subsystem is deformed during the simulation, exactly as it would be deformed as a part of the whole system under load. In order to achieve that, maintaining the full dynamical behavior of the system, all the finite element (FE) computations and data interchange between both subsystems must be performed in real-time [1,2].

The algorithm of hybrid simulation is presented in Figure 1 [3]. The algorithm executes step-after-step. In every time step, i the displacement vector for the next time step $\mathbf{u}_{i+1}$ is computed using FEM and imposed in the physical subsystem. The displacement develops forces in the physical

subsystem, which are measured in the form of a vector $\mathbf{r}^{\mathbf{P}}{}_{i+1}$, where subsequent elements of the vector correspond to degrees of freedom (DOFs) of the discretized FE model [4,5].

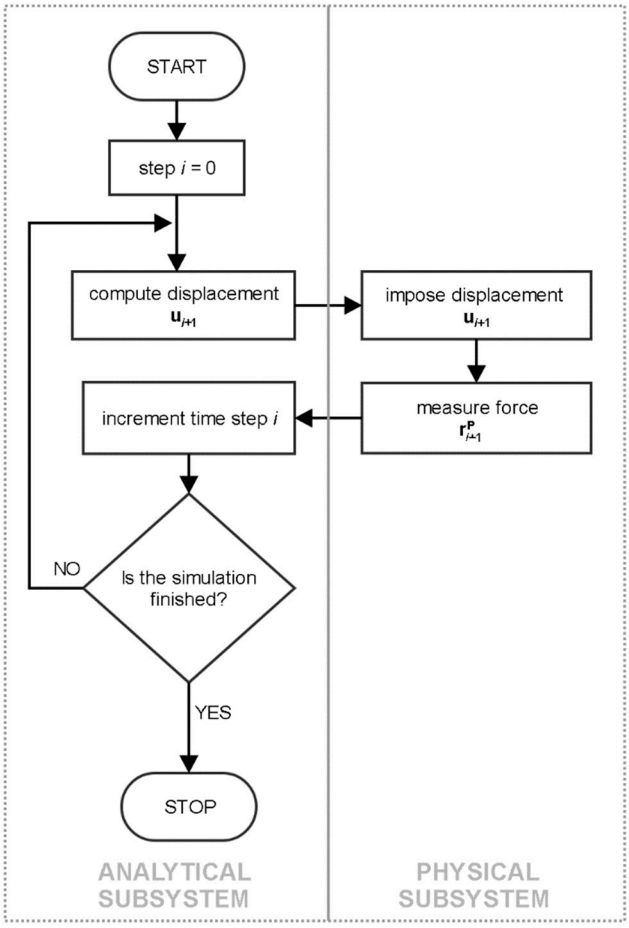


Figure 1. Hybrid simulation algorithm.

Many applications of hybrid simulation concern civil engineering and investigating the response of reinforced concrete or other structures to seismic excitations [6–8]. Mechanical systems containing nonlinear dumpers (e.g., magneto-rheological) are also typically investigated with the described method [9–11]. Hybrid simulation has also found broad applications in the automotive and aerospace industries [12–14].

Because FEM is a highly computationally demanding method, performing it in real-time may be difficult [15–17]. FE models of systems tested using hybrid simulations are rather simple, in most cases, built of the 1D beam, truss, or frame elements. Sources [6–9] have described modern (2013–2016) results of hybrid simulations in civil engineering. The mechanical systems presented in the papers are frames and beams containing highly nonlinear elements (e.g., magnetorheological dampers) that must be tested experimentally. The problem of utilizing very complex FE models (e.g., 3D structure with thousands of DOFs) is that (a) performing FE computations in real-time with time steps of milliseconds or fractions of milliseconds would be highly difficult or impossible and (b) realization of experiment would be difficult as the number of actuators connected to every node of the interface between physical and analytical subsystem must be equal to the number of DOFs of this node. Brodersen et al. [9] presented hybrid simulations of two structures with complex viscous dampers: wind turbine modeled with beam elements (eight DOFs) and shear frame with 10 DOFs.

In the hybrid simulation, there is a conflict between accuracy, number of DOFs, and time step size. The more DOFs the tested system is discretized to, the greater the accuracy of the model is obtained;

however, the greater the time step must also be set. The greater the time step is set, the lower the accuracy of the simulation is obtained (taking into account the dynamic behavior of the structure).

The current trend to solve the abovementioned problem, for systems that require discretization to relatively great numbers of DOFs, is to implement distributed computing, involving powerful hardware resources like computer clusters or supercomputers, where the communication between the laboratory with physical subsystem and computing resources is performed, utilizing peer-to-peer networks [18–21].

The main idea presented in the current study was opposite—the FE model had to be reduced while maintaining the highest possible accuracy in order not to involve very powerful computing resources for real-time simulations. Reducing the model is, however, not trivial, when accuracy must be maintained, and also usually requires a computational cost. Therefore, the following assumption was made: computational cost necessary for building the reduced model (offline—not in real-time) is of little importance as long as real-time computations during hybrid simulation can be performed effectively and efficiently. The efficiency of the computations is understood as the ability to test systems of a higher number of degrees of freedom while maintaining high accuracy, without increasing the time step. Effective simulation is a simulation that allows acquiring correct results while maintaining the time regime.

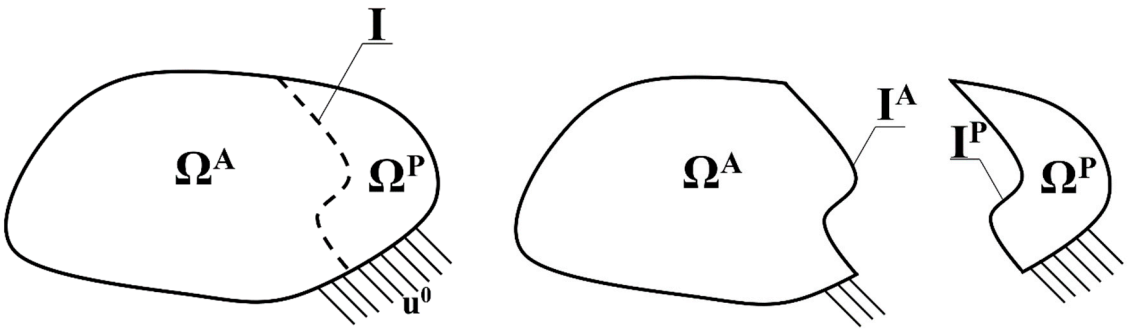
The author researched a method for model order reduction in hybrid simulation–mode superposition [22,23]. The method has proven the above assumptions, however, only for mechanical systems that meet certain requirements. The requirements are that the whole analytical subsystem is linear, the damping is proportional, and the physical subsystem is significantly smaller than the analytical. The method does not apply to other structures.

In the present paper, a completely different approach was presented—the whole finite element model was replaced with an artificial neural network (ANN). ANN plays the role of a surrogate model that mimics the behavior of the FE model while being incomparably cheaper to evaluate. This approach does not have the abovementioned limitations of the mode superposition method. In [24], the authors have suggested the first time the idea of utilizing surrogate models in hybrid simulation; however, without any details of what form the surrogate model should have. Since then, after many tests, they came to the conclusion that ANNs are most suited in this role—are resistant to noise and distortion of signals, can represent complex issues, and detect significant connections between data. In the literature, there can be found other examples than a hybrid simulation of successful substituting FE models by ANNs [25–27].

Details of the proposed method are described in Section 2. Section 3 contains numerical and experimental examples. The results are discussed in Section 4.

2. Materials and Methods

The mechanical system $\boldsymbol{\Omega}$ was considered. On the part of the boundary of $\boldsymbol{\Omega}$, displacements $\mathbf{u^0}$ are known. On the rest of the boundary, acting forces are known. Under the influence of forces, the mechanical system deforms. The motion of system $\boldsymbol{\Omega}$ is defined by displacement $\mathbf{u}$ in time t: $\mathbf{u}(\mathbf{x},t)$, where $\mathbf{x} \in \boldsymbol{\Omega}$. In the hybrid simulation, the mechanical system is divided into analytical subsystem $\boldsymbol{\Omega^A}$ and physical subsystem $\boldsymbol{\Omega^P}$, as presented in Figure 2, where $\boldsymbol{\Omega^A} \cup \boldsymbol{\Omega^P} = \boldsymbol{\Omega}$. The division between both subsystems is defined by interface $\mathbf{I}$. Boundary conditions for interface boundaries of subsystems: $\mathbf{I^A}$ and $\mathbf{I^P}$ are defined by actuators and force sensors. On the boundary $\mathbf{I^A}$, forces are known (measured by the sensors), and on the boundary $\mathbf{I^P}$, displacements are known (imposed by the actuators) [2,3,13].



(**a**) (**b**)

Figure 2. Considered mechanical system: (**a**) idea of the interface between subsystems; (**b**) division for hybrid simulation.

The equation of motion of mechanical system $\mathbf{\Omega}$ is as follows:

$$(\mathbf{M^A} + \mathbf{M^P})\ddot{\mathbf{u}}(t) + (\mathbf{C^A} + \mathbf{C^P})\dot{\mathbf{u}}(t) + (\mathbf{K^A}(t) + \mathbf{K^P}(t))\mathbf{u}(t) = \mathbf{F}(t) \tag{1}$$

where **M**, **C**, and **K** are mass, damping, and stiffness matrices, respectively. Superscript **A** denotes an analytical subsystem and superscript **P** physical subsystem. If the system is not linear elastic, the stiffness matrix is time-dependent. **F** is the vector of excitation forces [3].

Because forces developed in the physical subsystem are measured, Equation (1) takes the form:

$$\mathbf{M^A}\ddot{\mathbf{u}}(t) + \mathbf{C^A}\dot{\mathbf{u}}(t) + \mathbf{K^A}(t)\mathbf{u}(t) + \mathbf{r^P} = \mathbf{F}(t) \tag{2}$$

where the vector of measured forces $\mathbf{r^P}$ corresponds to the sum of inertia, damping, and stiffness forces:

$$\mathbf{r^P} = \mathbf{M^P}\ddot{\mathbf{u}}(t) + \mathbf{C^P}\dot{\mathbf{u}}(t) + \mathbf{K^P}(t)\mathbf{u}(t). \tag{3}$$

Normally in hybrid simulation, Equation (2) is solved step-after-step using explicit or implicit integration schemes. These algorithms in detail can be found in [5].

The artificial neural network, to substitute the finite element model, to be built and trained, should approximate the following function f:

$$\mathbf{u}^{\mathbf{I}}_{i+1} = f\left(\mathbf{r}^{\mathbf{P}}_i, \mathbf{F}_{i+1}, \mathbf{u}^{\mathbf{I}}_i, \mathbf{u}^{\mathbf{I}}_{i-1}, \mathbf{u}^{\mathbf{I}}_{i-2}\right) \tag{4}$$

where $\mathbf{u^I}$ is the displacement vector of the interface **I**. The five inputs to the ANN are result of the fact that in integration schemes for hybrid simulation, the displacement vector of the whole system for time step $i + 1$ is computed from (a) measured forces in the physical subsystem in time step i, (b) excitation force vector for time step $i + 1$, (c) displacements of the whole system in three previous discrete states of time before i+1 (sufficient approximation of velocity and acceleration of the system), (d) system matrices (mass, damping, and stiffness), where the stiffness matrix of the physical subsystem is approximated (at least roughly).

As one can see from Equation (4), the ANN does not compute the displacements of the whole system. In order to improve computational efficiency, only displacements of the interface **I** are calculated. This information is sufficient to perform hybrid simulation, and after it, the complete results can be easily reproduced numerically, knowing vectors $\mathbf{F}(t)$ and $\mathbf{r^P}(t)$.

The ANN is built based on the offline FE model from which training data is generated as a set of samples. Each sample consists of a set of five input values and the correct output value (displacement of the interface for the next iteration). Training data comes from subsequent states of the system in dynamic step-by-step simulation. Therefore, the built ANN must be used only in hybrid simulations with the same time step as set in dynamic FE simulation.

If the time course of the excitation forces $\mathbf{F}(t)$ is exactly known *a priori*, the ANN can be trained based on FE simulations with the same force-time course. If $\mathbf{F}(t)$ is ambiguous, the ANN can be trained for different variations assuming that they are finite and of a relatively small number. In other cases, $\mathbf{F}(t)$ should be parametrized, and training data are generated for different sets of discrete values of parameters. For example, if $\mathbf{F}(t)$ only has one non-zero element $F_0(t)$ acting sinusoidally,

$$F_0(t) = A \cdot \sin(\omega \cdot t), \tag{5}$$

with amplitude in range $A \in [A_{\min}, A_{\max}]$ and circular frequency $\omega \in [\omega_{\min}, \omega_{\max}]$, $n \times m$ dynamic FE simulations should be performed for every combination of $A_i \in [A_{\min}, A_{\max}]$ and $\omega_j \in [\omega_{\min}, \omega_{\max}]$, where $i = 1,2, \ldots, n$ and $j = 1,2, \ldots, m$.

A similar approach is used for simulating input $\mathbf{r^P}(t)$. In principle, hybrid simulation is utilized for structures where the physical subsystem is difficult or impossible to model numerically. Therefore, dynamic FE simulations cannot imitate the behavior of $\mathbf{\Omega^P}$. Limit values of the nonlinear stiffness of the physical subsystem must be assumed: $\mathbf{K^P} \in [\mathbf{K^P_{min}}, \mathbf{K^P_{max}}]$. A set of s dynamic FE simulations should be performed for every $\mathbf{K^P}_k$, where $k = 1,2, \ldots, s$. For example, concerning Equation (5), $n \times m \times k$ simulations would be performed to generate the training data. ANNs have the ability to generalize; therefore, by training them using linear FE simulations, they will be able to perform nonlinear hybrid simulations.

There is no unambiguous rule determining what architecture of ANN should be chosen for a given model for hybrid simulation. The trial and error method should be used in which architecture of the smallest possible number of neurons (that meets accuracy requirements) should be chosen—minimization of computational cost and the risk of overfitting the network. The author's experience is that two-layer feed-forward networks with sigmoid hidden neurons and linear output neurons often work well if properly trained—they mimic the FE model behavior with high accuracy and are resistant to noise.

The most important problem with training the ANN using known algorithms (Levenberg–Marquardt [28,29], Bayesian Regularization [30,31], Scaled Conjugate Gradient [32,33]) is that even if the ANN is trained resulting with high correlation and low error between outputs and reference data (divided to training, validation, and testing data), the ANN does not show convergence when working in recurrence during hybrid simulation. When outputs from ANN become its inputs in subsequent time steps, the common problem is that the results become inconsistent with reality either by large error or lack of convergence when $\mathbf{u^I} \to \pm\infty$.

In order to solve this problem and obtain minimal architecture of the network, the author proposed to build and train the ANN according to the algorithm presented in Figure 3. The algorithm is dedicated to ANNs with one hidden layer, and extending it to networks with a greater number of hidden layers is trivial. Before starting the process, the minimum number of neurons in hidden layer N_{min} (in most cases $N_{min} = 1$) and a maximum number of attempts n to train the same architecture should be set. After each training of ANN with a known method, it should be tested offline in a set of numerical dynamic simulations in which the outputs from ANN are compared to reference data from FE simulations, and mean squared error (MSE) between time courses of $\mathbf{u^I}$ is compared. The author advised simulating the behavior of the physical subsystem with an arbitrary chosen nonlinear function that lies between the stiffness limits assumed in the ANN training data generation process. If the MSE is less or equal to maximum error MSE_{max}, the ANN is considered as correct, the algorithm should stop, and the ANN should be saved to be used in real-time hybrid simulations.

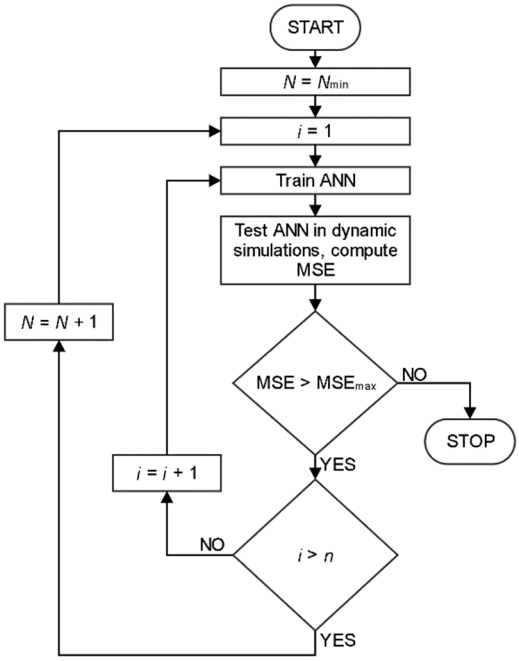


Figure 3. Algorithm for building, training, and testing artificial neural network (ANN) for hybrid simulation.

3. Results

Two numerical and one experimental example were discussed. The provided examples verified and demonstrated the effectiveness of the proposed methodology and algorithms. In the numerical examples, an experiment on the physical subsystem was simulated using different nonlinear functions. All the computations were performed on a PC, and the model order reduction measure was the comparison of (offline) simulation time between the FE model and created ANN. MSEs were also presented. A real-time hybrid simulation with a non-simulated experiment was considered as the last, practical example.

3.1. Numerical Verification of Implementing ANN in Hybrid Simulation on the Example of Beam Supported by Nonlinear Spring

A system of a beam (square pipe with outer dimension $B = 48$ mm and inner dimension $b = 45$ mm, of length $2l = 1$ m), fixed at point A and supported by a nonlinear spring at point C, presented in Figure 4, was considered as the first numerical example. The beam was loaded perpendicularly, in the middle of its length, by time-dependent force $\boldsymbol{F}$. The beam was made of aluminum alloy (Young modulus 69 GPa and density 2720 kg/m^3). The beam was considered as the analytical subsystem and the spring as the physical subsystem in hybrid simulation.

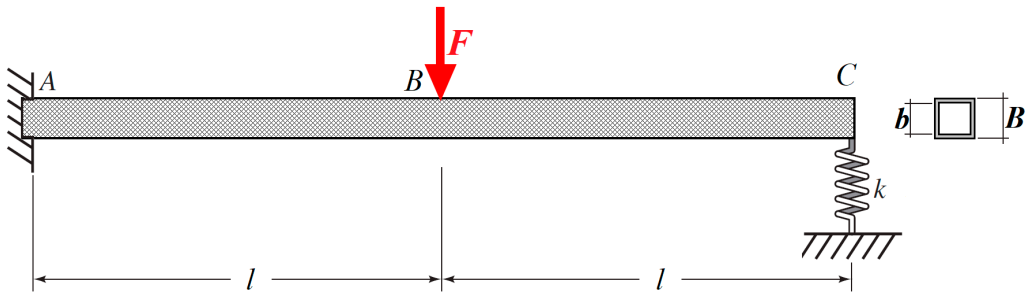


Figure 4. Numerical example 1—mechanical system.

In order to generate the reference data to train ANN, the beam was modeled using 10 beam finite elements of equal size. The spring was modeled as a one-rod finite element. Therefore, the system got 20 DOFs (10 transitional and 10 rotational). Rayleigh damping model was adapted, where the damping matrix is proportional to stiffness and mass matrices,

$$\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}, \tag{6}$$

with coefficients $\alpha = 2$ and $\beta = 0.4$. The time step of the simulation was set to 0.1 ms.

The excitation force $\boldsymbol{F}$ has the following time-course:

$$\boldsymbol{F}(t) = -A[\cos(2\pi f \cdot t) - 1]. \tag{7}$$

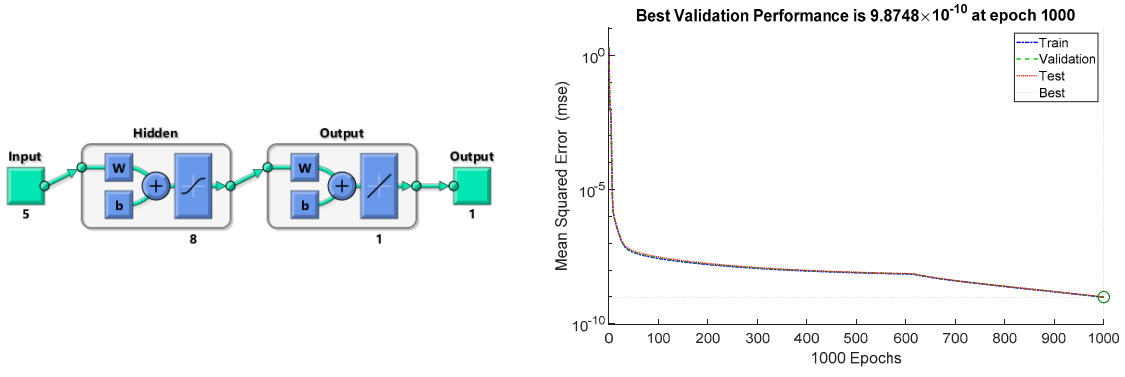
with parameters: amplitude A and frequency f. The following assumptions were made: amplitude cannot exceed 250 N and frequency 100 Hz, the nonlinear stiffness of the spring is in the range from 30 to 180 N/mm.

Fifty FE simulations were performed for every combination of parameters A = 25 N, 50 N, 75 N, ... , 250 N and k = 30 N/mm, 45 N/mm, 60 N/mm, ... , 150 N/mm. Every analysis was performed with swept-frequency f from 0 to 100 Hz. A total of 449,820 data samples were generated.

There was only one nonzero entry in the vectors $\mathbf{u^I}$ and $\mathbf{r^P}$ (corresponding to vertical displacement of the spring). The exciting force vector $\mathbf{F}$ was represented only by force applied at point B. Therefore, the ANN got five inputs and one output.

Two-layer ANN was trained, with a sigmoid activation function in the hidden layer and linear activation function in the output layer. A total of 70% of the reference data were used as training data, 15% as validation data, and 15% as test data. To train the ANN, Levenberg–Marquardt algorithm was utilized [34] with the random initial solution (normalized using the method of Nguyen and Widrow [35]).

When building the ANN according to the algorithm presented in Figure 3, the following parameters were set: $N_{min} = 1$, $n = 20$, $\mathrm{MSE_{max}} = 10^{-3}$ mm^2 (sum of errors of all four simulations presented below). The ANN got eight neurons in the hidden layer. Its architecture and a plot of training record error values against the number of training epochs are presented in Figure 5. The time of execution of the algorithm presented in Figure 3 was about 2.5 h.



(**a**) (**b**)

Figure 5. ANN obtained in example 1: (**a**) architecture, (**b**) performance of training.

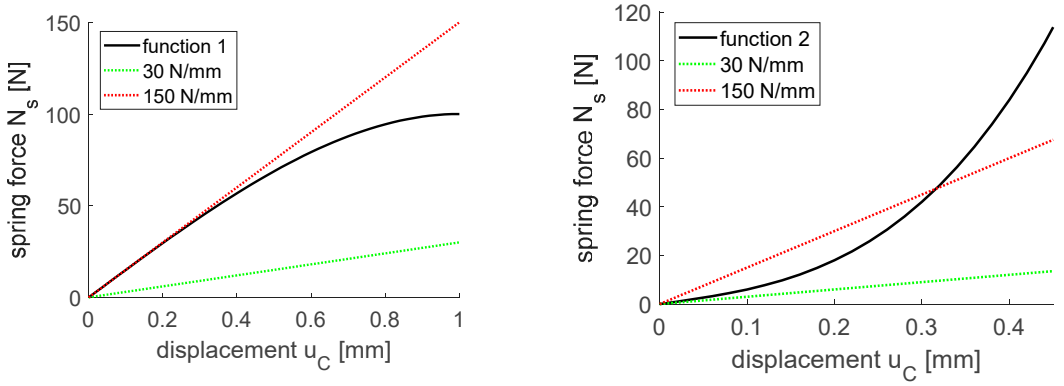
To verify the obtained ANN, the nonlinear elasticity of the spring was described by the value of normal force generated in the spring $\boldsymbol{N_s}$ (N) in the function of vertical displacement of point C u_c (mm). Two alternative functions were taken into consideration:

$$\boldsymbol{N}_{s,1} = 150u_C - 50(u_C)^3 \tag{8}$$

and

$$N_{s,2} = 50u_C + 1000(u_C)^3. \quad (9)$$

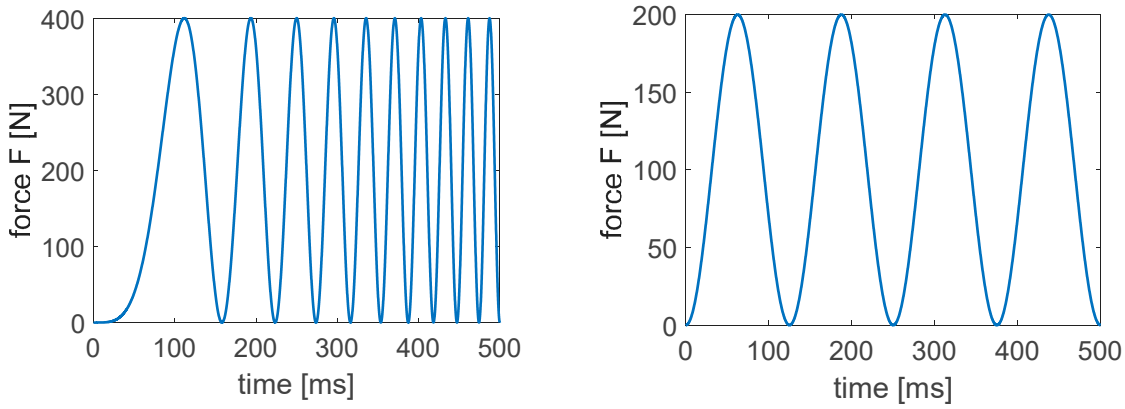
Substantiation about the selection of functions (8) and (9) is presented in Figure 6 with the lowest and highest spring stiffness k used to generate the highlighted reference data. Nonlinear odd functions in the range of k on which neural network was trained (or near it, as ANNs have the ability to generalize) were chosen.



(**a**) (**b**)

Figure 6. Nonlinear functions used to simulate the physical subsystem: (**a**) function (8), (**b**) function (9).

Two alternative time courses of force $\boldsymbol{F}$ were taken into consideration and are presented in Figure 7.



(**a**) (**b**)

Figure 7. Time course of force $\boldsymbol{F}$: (**a**) load case 1, (**b**) load case 2.

Time courses of vertical displacement of point C obtained from ANN and FE model of hybrid simulation (considered as exact) are presented in Figure 8, and mean-squared errors are summarized in Table 1.

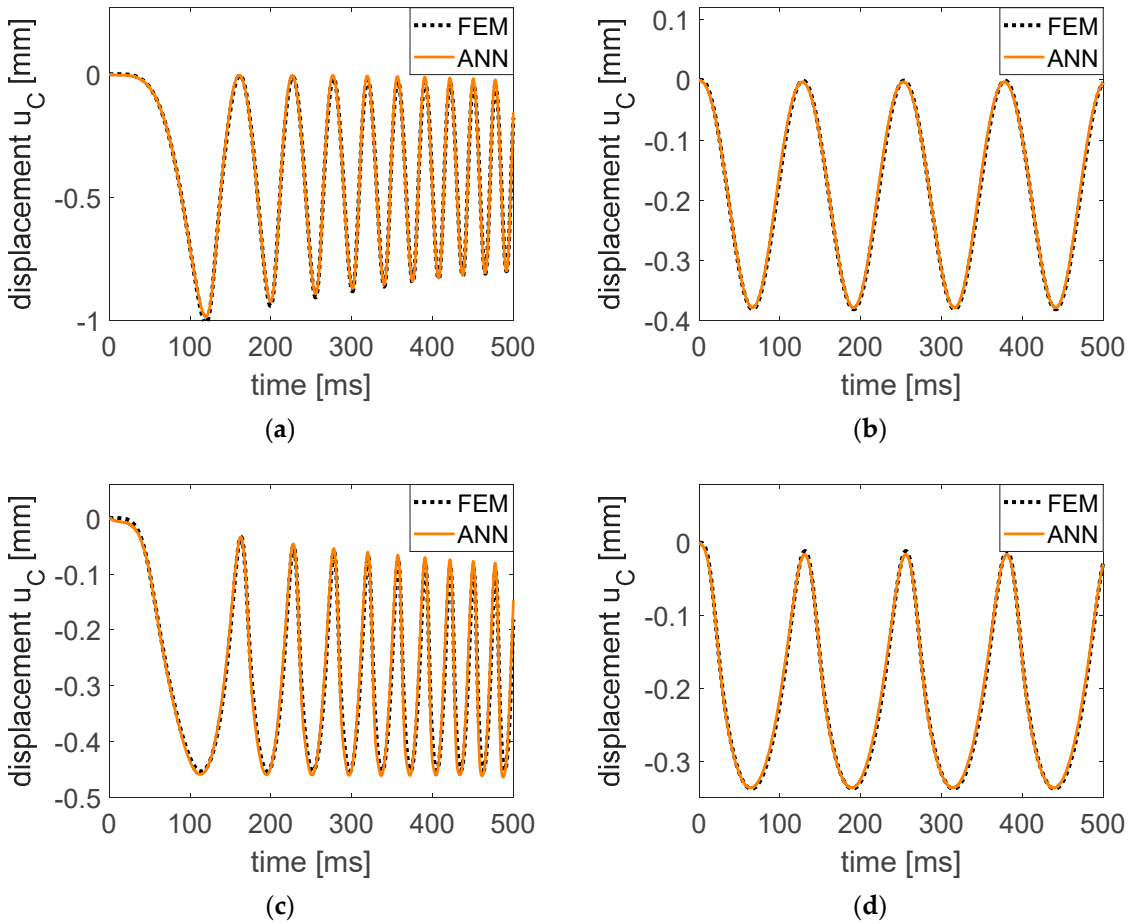


Figure 8. Time course of vertical displacement of point C: (**a**) load case 1 and function (8), (**b**) load case 2 and function (8), (**c**) load case 1 and function (9), (**d**) load case 2 and function (9). FEM, finite element method.

Table 1. Mean-squared errors of vertical displacement of point C (mm^2).

Function of Spring Stiffness	Load Case 1	Load Case 2
Equation (8)	1.75×10^{-4}	2.93×10^{-5}
Equation (9)	2.72×10^{-4}	2.13×10^{-5}

Table 2 contains a comparison of simulation time for FE analysis (Newmark scheme) and ANN. Simulations were performed in MATLAB software on a PC (Intel Core i7-8700 3.20 GHz, 8 GB RAM). The table shows the average of 10 measurements. Although the calculations were not performed in real-time as in actual hybrid simulation, the simulation times presented in Table 2 are also a measure of the model reduction.

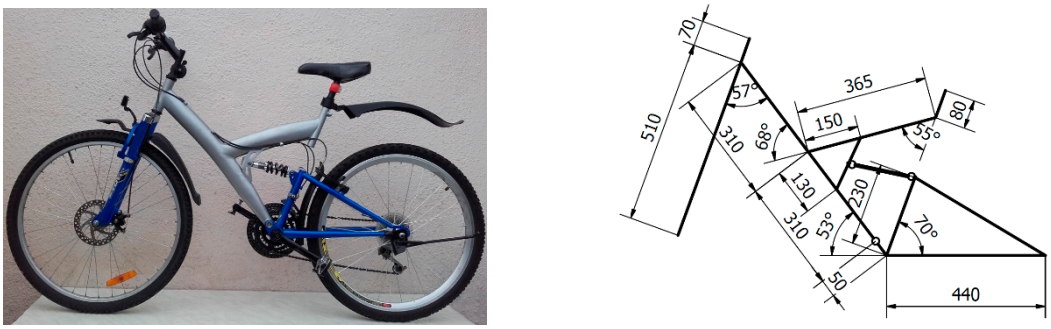
Table 2. Simulation times comparison—example 1.

Function of Spring Stiffness	Load Case 1	Load Case 2
Equation (8)	FEM: 0.54 s ANN: 0.26 s	FEM: 0.55 s ANN: 0.25 s
Equation (9)	FEM: 0.56 s ANN: 0.25 s	FEM: 0.54 s ANN: 0.25 s

FEM—Finite Element Method, ANN—Artificial Neural Network.

3.2. Numerical Verification of Implementing ANN in Hybrid Simulation on the Example of Bicycle Frame with Nonlinear Shock Absorber

A steel (Young modulus 200 GPa, density 7800 kg/m^3) mountain bicycle frame was considered as the second numerical example and is presented in Figure 9. The frame was equipped with nonlinear shock absorber, consisting of the physical subsystem in hybrid simulation. The rest of the mechanical system was linear elastic, consisting of the analytical subsystem.



(**a**) (**b**)

Figure 9. Numerical example 2—mechanical system: (**a**) photograph; (**b**) CAD model (reprinted from [23], with the permission of AIP Publishing).

The finite element model consisted of 27 finite elements, 73 DOFs, and is presented in Figure 10. The element 17 represented the shock absorber and was of the rod type. The rest of the elements were of the plane frame type. The node linking elements 18 and 19 were a ball joint; therefore, they had four DOFs: vertical translation, horizontal translation, left-side rotation, and right-side rotation. Force ***F1*** represented the load from the handlebar, and force ***F2*** represented the load from the saddle. Cross-section parameters of subsequent elements are given in Table 3. The Rayleigh damping (6) coefficients were $\alpha = 2$ and $\beta = 0.1$. The time step was 1 ms.

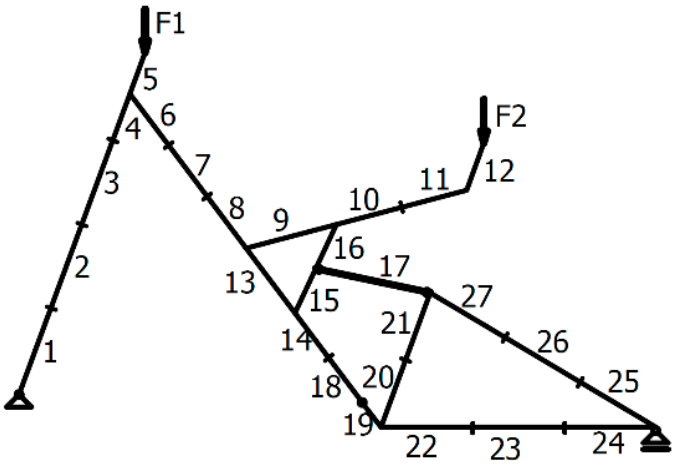


Figure 10. Finite element model of bicycle frame (reprinted from [23], with the permission of AIP Publishing).

Table 3. Cross-section parameters of the bicycle frame (reprinted from [23], with the permission of AIP Publishing).

Element Number	Cross-Section Area (mm^2)	Moment of Inertia (mm^4)
1–3	287	33,507
4–5	158	22,190
6–8, 13	221	92,112
9–11	231	104,925
12	130	12,287
14	210	75,663
15–16	210	36,463
18	172	35,695
19	540	91,125
20–21	134	13,674
22–27	174	7508

It was assumed that in hybrid simulation, the time courses of the values of **F1** and **F2** were described as follows:

$$\boldsymbol{F2}(t) = -A[\cos(2\pi f \cdot t) - 1] \tag{10}$$

and

$$\boldsymbol{F1}(t) = 0.4 \cdot \boldsymbol{F2}(t) \tag{11}$$

where A = 400 N, f = 3 Hz, and t denotes time.

In order to generate the reference data, it was assumed that the nonlinear stiffness of the shock absorber could vary in a range from 60 to 200 N/mm. Fifteen finite element analyses were performed, where linear stiffness of the shock absorber was adapted and changed every 10 N/mm. The time of a single dynamic simulation was 3 s. A total of 44,970 reference data samples were generated.

It was assumed that in hybrid simulation, the shock absorber was mounted to four actuators, imposing displacements along the directions of the global coordinate system, as presented in Figure 11. In the presented case, there were four nonzero entries in the vectors $\mathbf{u^I}$ and $\mathbf{r^P}$ (corresponding to horizontal and vertical displacements and forces of points A and B—Figure 12). The exciting force vector **F** could be represented only by force **F2** as **F1** was always proportional. Therefore, the ANN got 17 inputs and four outputs.

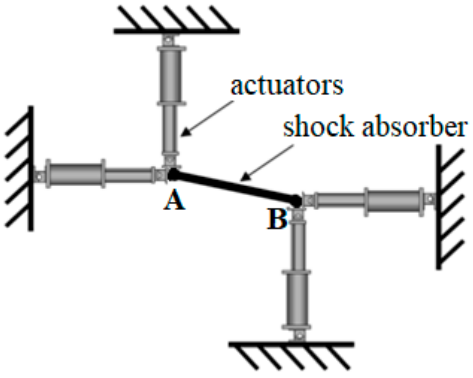
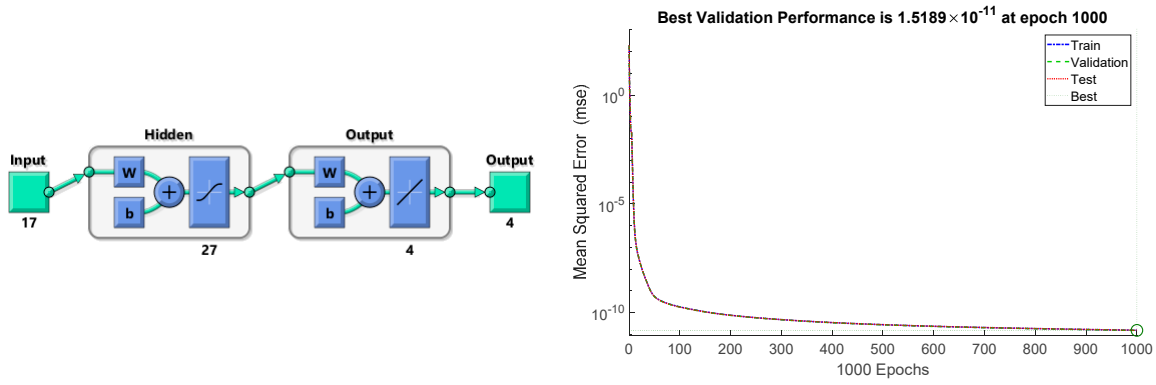


Figure 11. Method of mounting actuators to the shock absorber.



(**a**) (**b**)

Figure 12. ANN obtained in example 2: (**a**) architecture, (**b**) performance of training.

Two-layer ANN was trained, with a sigmoid activation function in the hidden layer and linear activation function in the output layer. A total of 70% of the reference data were used as training data, 15% as validation data, and 15% as test data. To train the ANN, Levenberg–Marquardt algorithm was utilized [34] with the random initial solution (normalized using the method of Nguyen and Widrow [35]).

When building the ANN according to the algorithm presented in Figure 3, the following parameters were set: $N_{min} = 1$, $n = 20$, $\mathrm{MSE}_{\max} = 0.5$ mm^2 (sum of average error of all four outputs from two simulations presented below). The ANN got 27 neurons in the hidden layer. Its architecture and a plot of training record error values against the number of training epochs are presented in Figure 12. The time of execution of the algorithm presented in Figure 3 was over 3 days.

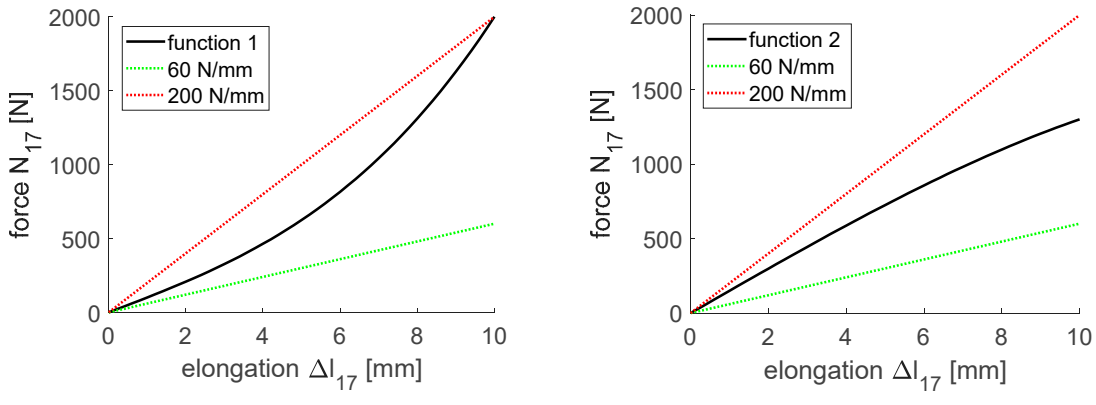
To verify the obtained ANN, the nonlinear elasticity of the shock absorber was described by the value of normal force generated in the element 17 $\boldsymbol{N}_{17}$ (N) in the function of its elongation $\boldsymbol{\Delta l}_{17}$ (mm). Two alternative functions were taken into consideration:

$$\boldsymbol{N}_{17,1} = 100\Delta l_{17} + (\Delta l_{17})^3 \tag{12}$$

and

$$\boldsymbol{N}_{17,2} = 150\Delta l_{17} - 0.2(\Delta l_{17})^3. \tag{13}$$

Substantiation about the selection of functions (12) and (13) is presented in Figure 13 with the lowest and highest stiffness of the shock absorber used to generate the highlighted reference data. Nonlinear odd functions in the presented range of stiffness were chosen.



(**a**) (**b**)

Figure 13. ANN obtained in example 2: (**a**) architecture, (**b**) performance of training.

Time courses of displacements of points A and B obtained from ANN and FE model of hybrid simulation (considered as exact) are presented in Figures 14 and 15, and mean-squared errors are summarized in Tables 4 and 5.

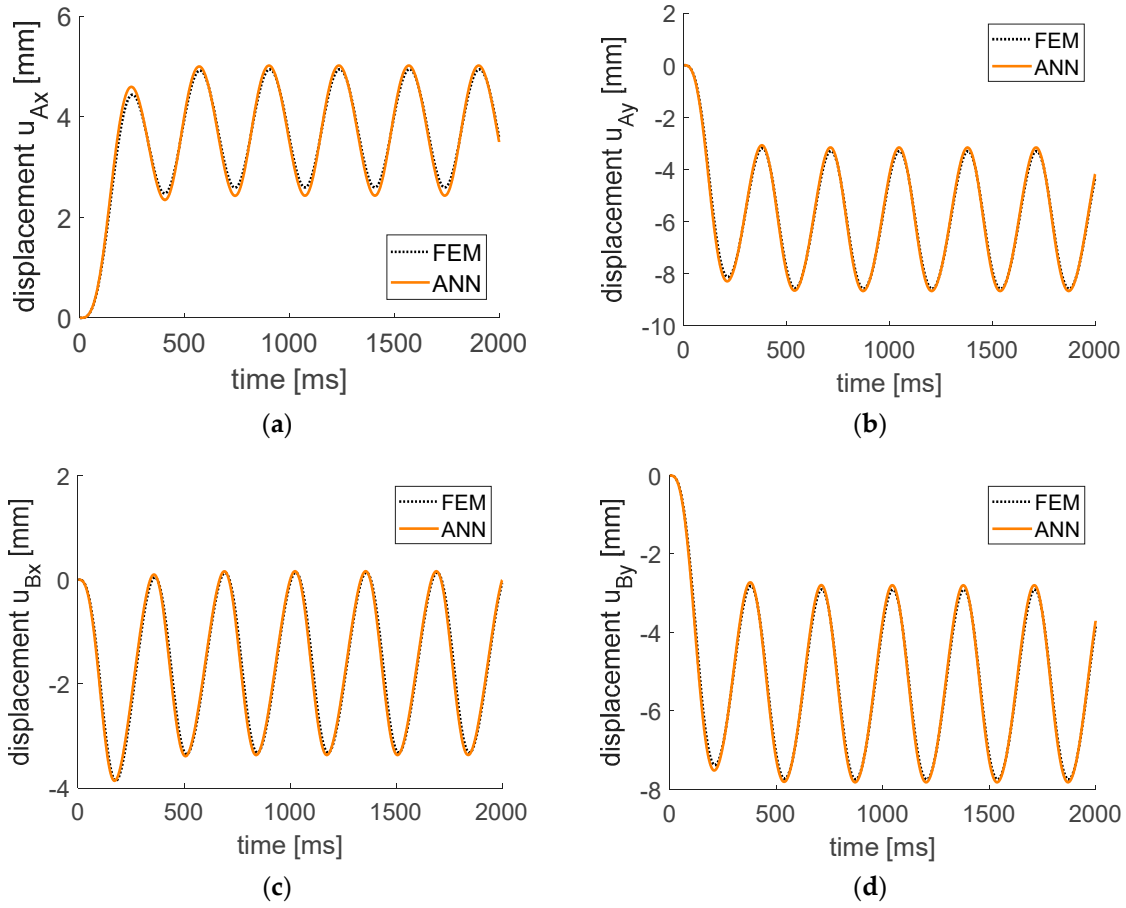


Figure 14. Time course of displacements for shock absorber function (12): (**a**) horizontal displacement of point A, (**b**) vertical displacement of point A, (**c**) horizontal displacement of point B, (**d**) vertical displacement of point B.

Table 4. Mean-squared errors of displacements for shock absorber function (12) (mm^2).

	Point A	**Point B**
Horizontal	0.0101	0.0145
Vertical	0.0144	0.0115

Table 5. Mean-squared errors of displacements for shock absorber function (13) (mm^2).

	Point A	**Point B**
Horizontal	0.0134	0.0250
Vertical	0.0225	0.0225

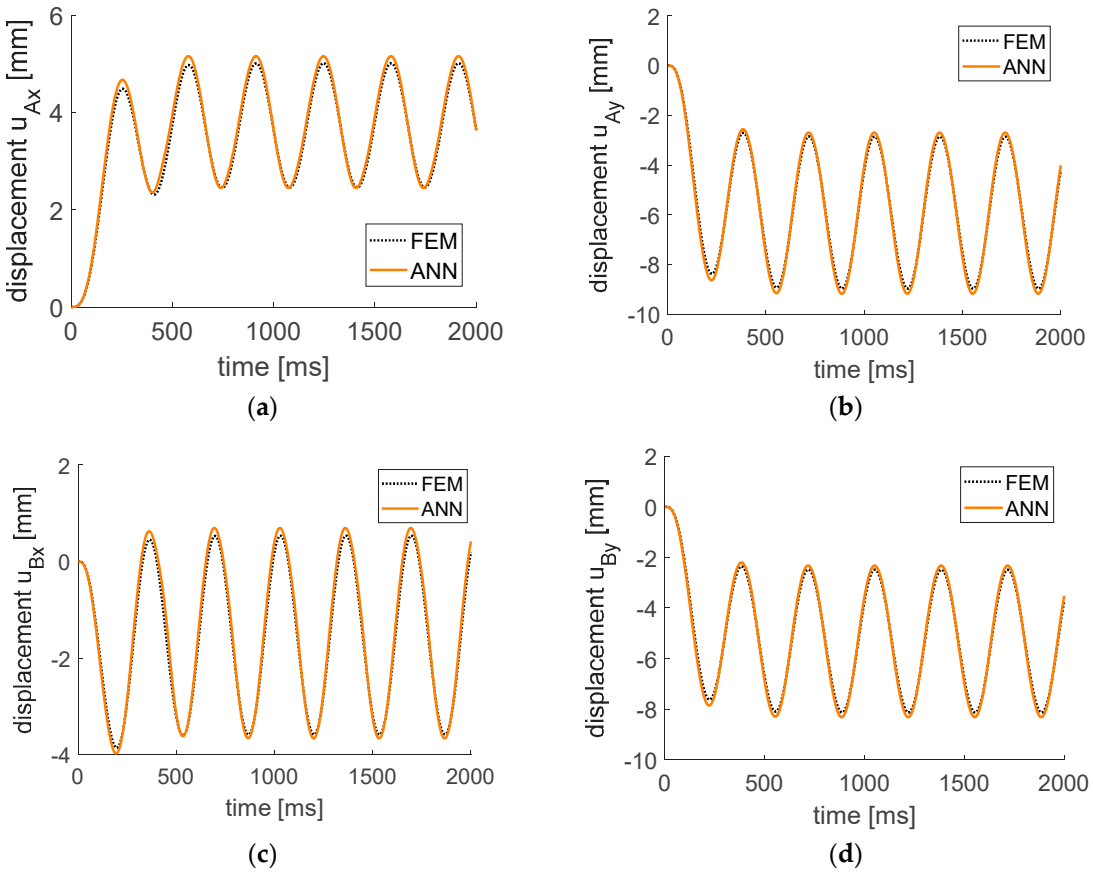


Figure 15. Time course of displacements for shock absorber function (13): (**a**) horizontal displacement of point A, (**b**) vertical displacement of point A, (**c**) horizontal displacement of point B, (**d**) vertical displacement of point B.

Similarly, as in the previous example, PC simulation times were measured. The average time from 20 simulations (10 for each shock absorber function) was 0.238 s for ANN and 1.296 s for FE analysis (Newmark scheme).

3.3. Experimental Verification of Implementing ANN in Hybrid Simulation on the Example of Bicycle Frame with Nonlinear Shock Absorber

Real-time hybrid simulation of the mechanical system presented in the previous example (3.2) was performed using ANNs. All the computations during hybrid simulation were executed on microcontroller National Instruments myRIO-1900 with the real-time operating system. The experiment was performed using dynamic materials testing machine Instron ElectroPuls E10000, which was controlled in a closed-loop by the microcontroller. Considering the capabilities of the testing machine, only axial elongation of the shock absorber was applied in the experiment, and its normal forces were measured. The inertia forces perpendicular to the axis of the shock absorber were transferred to the analytical subsystem. Figure 16 presents a photograph taken during the experiment.

It was assumed that in hybrid simulation, the time courses of the values of **F1** and **F2** were described as follows:

$$\boldsymbol{F2}(t) = \begin{cases} \frac{A_1}{t_1} t, & \text{for } t < t_1 \\ A_1 - A_2[\cos(2\pi f(t - t_1)) - 1], & \text{for } t \geq t_1 \end{cases} \tag{14}$$

and

$$\boldsymbol{F1}(t) = 0.4 \cdot \boldsymbol{F2}(t) \tag{15}$$

where A_1 = 500 N, t_1 = 2.5 s, A_2 = 200 N, $f \in [0, 4\text{ Hz}]$, and t denotes time.

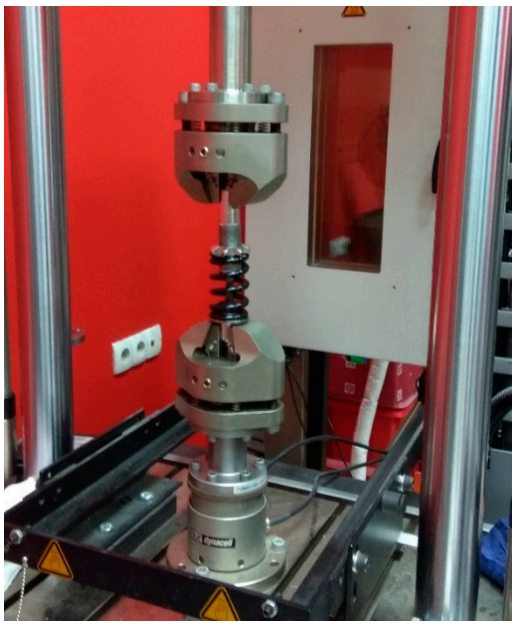


Figure 16. Photograph of shock absorber mounted in the testing machine.

In order to generate the reference data, it was assumed that the nonlinear stiffness of the shock absorber could vary in a range from 110 to 150 N/mm. Nine finite element analyses were performed, where linear stiffness of the shock absorber was adapted and changed every 5 N/mm. The time of a single dynamic simulation was 5 s. The frequency f changed linearly in each simulation from 0 to 4 Hz (Figure 15a). The time step of the simulations was 0.5 ms. A total of 89,973 reference data samples were generated.

In the hybrid simulation, the shock absorber was mounted to one linear actuator. Therefore, there was only one nonzero entry in vectors $\mathbf{u^I}$ and $\mathbf{r^P}$. The exciting force vector **F** could be represented only by force **F2** as **F1** was always proportional. Therefore, the ANN got five inputs and one output.

ANN with similar architecture, as in the previous example, was trained with the same training algorithm and parameters. When calculating the MSE (according to algorithm presented in Figure 3), a sum of errors from two simulations was taken into consideration: assuming swept-frequency from 0 to 4 Hz, and constant frequency of 2.5 Hz (while a nonlinear function was simulating the elastic behavior of the shock absorber), both presented in Figure 15. ANN with two neurons in the hidden layer was obtained. The training algorithm was executed the second time, and ANN with 20 neurons in the hidden layer was obtained. The times of execution of the algorithm presented in Figure 3 were about 10 min and over a dozen hours, respectively.

Real-time hybrid simulations, utilizing both ANNs, were performed for both load cases presented in Figure 17. Time courses of elongation of the shock absorber in the performed experiments are presented in Figure 18.

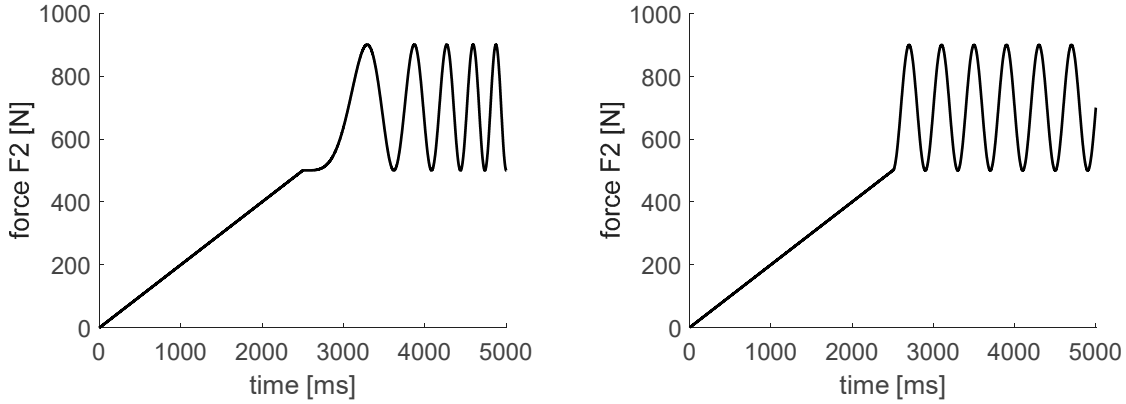


Figure 17. Time course of force F2: (**a**) load case 1, (**b**) load case 2.

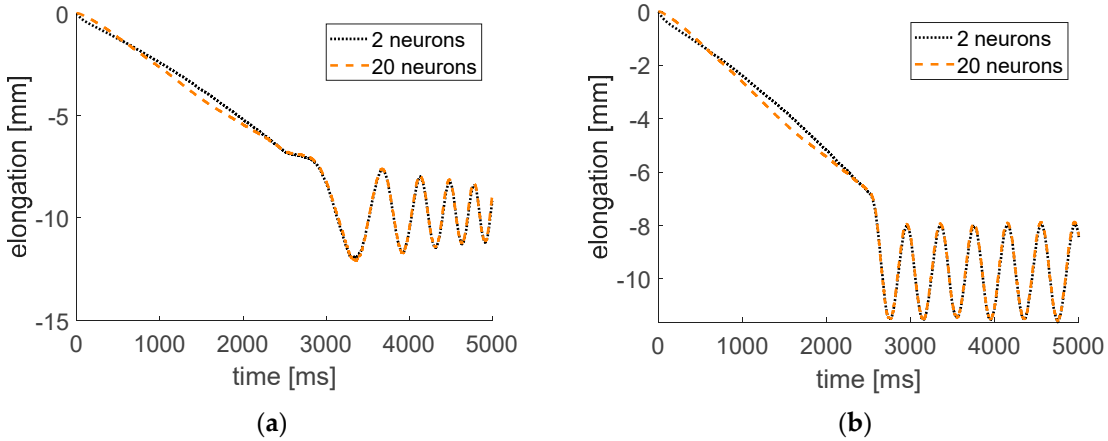


Figure 18. Elongation of the shock absorber: (**a**) load case 1, (**b**) load case 2.

4. Discussion

The provided two numerical examples and one experimental example proved that ANNs could be utilized in the hybrid simulation to substitute the finite element model, to speed-up the computations.

In the first numerical example, two-layer ANN with only eight hidden neurons could be utilized, providing the high accuracy of the results. About 50% of computational time reduction was achieved in a PC simulation. If taking this as a measure of model order reduction, one could expect that in real-time computations on given hardware, the time step of hybrid simulation could be reduced by half by utilizing ANN. The force $\boldsymbol{F}$ had two not constant parameters—amplitude A and frequency f. By performing a set of simulations to generate the reference data (with discrete values of amplitude and linearly swept-frequency), the ANN could generalize it and provide accurate results for any values of f and A (within the training range), constant or not.

In the second numerical example, a much more complex mechanical system was presented, containing 73 DOFs, while four of them make up the interface **I**. Therefore, the ANN got more inputs and outputs than in the first example. The ANN search algorithm resulted in 27 neurons in the hidden layer. The accuracy of the results obtained by ANN compared to the accurate FE model is satisfying, taking into account the reduction of computational time by five times.

The third example (experimental) proved that ANNs could be utilized in real-time hybrid simulations. The ANN search algorithm was executed twice resulting in 2 and 20 hidden neurons, testifying strongly to non-deterministic (random initial weights, random division of reference data to training, test, and validation) character of the algorithm (it may also indicate that $n = 20$ was too low in this case). Implementing the ANNs allowed to reduce the time step of the simulation to 0.5 ms, which would be impossible without model order reduction with the utilized microcontroller (the same

hardware was used by author in [3] to perform different hybrid simulation of system with 17 DOFs, and the lowest time step that could be achieved was 5 ms). Therefore, it is hard to measure the accuracy of the obtained results from the reduced models (the nonlinearity of the shock absorber cannot be simulated accurately). However, by comparing the results from two significantly different ANNs (presented in Figure 16), it can be said that they are very similar, which can lead to a conclusion about their accuracy. Furthermore, both ANNs demonstrated high accuracy in testing simulations during the execution of the searching algorithm. The ANN with 20 hidden neurons appeared to be more accurate, taking into account more linear regression of the elongation (Figure 16) in the first phase of loading the mechanical system.

All the examples proved the important assumption that by generating reference data from a proper set of linear simulations, the ANN was able to generalize it and provide accurate results in hybrid simulations with the nonlinear physical subsystem.

The conducted research is a starting point for future research on the development and applications of artificial intelligence techniques in hybrid simulations. The obtained results and conclusions drawn from them indicate the possibilities of future research regarding testing different architectures of neural networks and deep learning techniques in order to reduce the model order in hybrid simulations.

5. Conclusions

Reducing the model order in hybrid simulations is crucial because FEM is a very computationally demanding method, and performing it in real-time with small time steps can be difficult. For hybrid simulations with dynamic loading, the smaller the time step is, the more accurate results can be obtained. Therefore, by reducing the model order, the accuracy of hybrid simulation may increase.

Provided examples proved that the real-time computations in the hybrid simulation could be significantly reduced while maintaining high accuracy by substituting the FE model with an ANN. The presented algorithm for searching the ANN allowed finding the smallest possible network if the parameters of the algorithm were appropriately chosen.

The whole point of performing hybrid simulation was that the nonlinearities of the physical subsystem could not be accurately simulated. As proven, by performing a set of linear FE simulations to generate the reference data for training, the ANN was able to generalize it and give accurate results in hybrid simulations with high nonlinearities in the physical subsystem.

The presented algorithm was highly nondeterministic and could not guarantee a positive result each time, or it could give a positive result after large calculation costs. However, often, it is worth conducting several dozen hours of calculations offline in order to reduce the time step by 1 ms in real-time. After performing a hybrid simulation with ANN and obtaining all the data from it, the exact behavior of the whole mechanical system could be easily recreated using FEM.

Currently, to perform fully dynamic real-time hybrid simulations of systems, several dozen of DOFs distributed computing is introduced, involving powerful hardware resources like computer clusters or supercomputers with communication using P2P. The method presented in the paper allowed performing the same hybrid simulations using much smaller hardware resources or perform hybrid simulations with much smaller time steps on the same hardware resources.

Funding: The research was partially funded from financial resources from the statutory subsidy of the Faculty of Mechanical Engineering, Silesian University of Technology, in 2019.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Nakashima, M.; McCormick, J.; Wang, T. Hybrid simulation: A historical perspective. In *Hybrid Simulation: Theory, Implementation and Applications*; BALKEMA: London, UK, 2008; pp. 3–14.
2. Drazin, P.L.; Govindjee, S. Hybrid simulation theory for a classical nonlinear dynamical system. *J. Sound Vib.* **2017**, *392*, 240–259. [CrossRef]

3. Mucha, W. Real-time hybrid simulation using materials testing machine and FEM. In *Advances in Mechanics: Theoretical, Computational and Interdisciplinary Issues, Proceedings of the 3rd Polish Congress of Mechanics (PCM) and 21st International Conference on Computer Methods in Mechanics (CMM)-PCM-CMM-2015, Gdańsk, Poland, 8–11 September 2015*; CRC Press/Balkema: Leiden, The Netherlands, 2016; pp. 419–422.
4. Bursi, O.S. Computational techniques for simulation of monolithic and heterogeneous structural dynamic systems. In *Modern Testing Techniques for Structural Systems: Dynamics and Control*; Bursi, O.S., Wagg, D., Eds.; Springer Vienna: Vienna, Austria, 2008; 96p, ISBN 978-3-211-09445-7.
5. Shing, M. Integration schemes for real-time hybrid testing. In *Hybrid Simulation: Theory, Implementation and Applications*; BALKEMA: London, UK, 2008; pp. 25–34.
6. Mahmoud, H.; Elnashai, A. Hybrid simulation of semi-rigid partial-strength steel frames. In *Structures Congress 2013: Bridging Your Passion with Your Profession*; American Society of Civil Engineers: Pittsburgh, PA, USA, 2013; pp. 2410–2420.
7. Murray, J.A.; Sasani, M. Seismic shear-axial failure of reinforced concrete columns vs. system level structural collapse. *Eng. Fail. Anal.* **2013**, *32*, 382–401. [CrossRef]
8. Ramos, M.D.C.; Mosqueda, G.; Hashemi, M.J. Large-scale hybrid simulation of a steel moment frame building structure through collapse. *J. Struct. Eng.* **2016**, *142*. [CrossRef]
9. Brodersen, M.L.; Ou, G.; Høgsberg, J.; Dyke, S. Analysis of hybrid viscous damper by real time hybrid simulations. *Eng. Struct.* **2016**, *126*, 675–688. [CrossRef]
10. Christenson, R.; Lin, Y.Z. Real-time hybrid simulation of a seismically excited structure with large-scale Magneto-Rheological fluid dampers. In *Hybrid Simulation: Theory, Implementation and Applications*; BALKEMA: London, UK, 2008; pp. 169–180.
11. Zapateiro, M.; Karimi, H.R.; Luo, N.; Spencer, B.F., Jr. Real-time hybrid testing of semiactive control strategies for vibration reduction in a structure with MR damper. *Struct. Control Health Monit.* **2009**, *17*, 427–451. [CrossRef]
12. Van der Auweraer, H.; Vecchio, A.; Peeters, B.; Dom, S.; Mas, P. Hybrid testing in aerospace and ground vehicle development. In *Hybrid Simulation: Theory, Implementation and Applications*; BALKEMA: London, UK, 2008; pp. 203–214.
13. Ayari, L. Hybrid testing & simulation—The next step in verification of mechanical requirements in the aerospace industry. In *Hybrid Simulation: Theory, Implementation and Applications*; BALKEMA: London, UK, 2008; pp. 215–224.
14. Gagliano, C.; Martin, A.; Cox, J.; Clavin, K.; Gérard, F.; Michiels, K. *A Hybrid Full Vehicle Model for Structure Borne Road Noise Prediction*; SAE International: Warrendale, PA, USA, 2005.
15. Kuś, W.; Mucha, W. Real time computations using FEM-review of applications and possibilities. In Proceedings of the Mechanika 2014. Proceedings of the 19th International Conference, Kaunas, Lithuania, 24–25 April 2014; Technologija: Kaunas, Lithuania, 2014; pp. 153–156.
16. Röck, S.; Pritschow, G. Real-time capable Finite Element Models with closed-loop control: A method for Hardware-in-the-Loop simulation of flexible systems. *Prod. Eng.* **2007**, *1*, 37–43. [CrossRef]
17. Joldes, G.R.; Wittek, A.; Miller, K. Real-time nonlinear finite element computations on GPU—Application to neurosurgical simulation. *Comput. Methods Appl. Mech. Eng.* **2010**, *199*, 3305–3314. [CrossRef] [PubMed]
18. Peng, P.; Hiroshi, T.; Tao, W.; Masayoshi, N.; Makoto, O.; Mosalam, K.M. Development of peer-to-peer (P2P) internet online hybrid test system. *Earthq. Eng. Struct. Dyn.* **2006**, *35*, 867–890.
19. Wang, T.; McCormick, J.; Yoshitake, N.; Pan, P.; Murata, Y.; Nakashima, M. Collapse simulation of a four-story steel moment frame by a distributed online hybrid test. *Earthq. Eng. Struct. Dyn.* **2008**, *37*, 955–974. [CrossRef]
20. Wang, T.; Yoshitake, N.; Pan, P.; Lee, T.-H.; Nakashima, M. Numerical characteristics of peer-to-peer (P2P) Internet online hybrid test system and its application to seismic simulation of SRC structure. *Earthq. Eng. Struct. Dyn.* **2008**, *37*, 265–282. [CrossRef]
21. Kwon, O.-S.; Nakata, N.; Elnashai, A.; Spencer, B. A framework for multi-site distributed simulation and application to complex structural systems. *J. Earthq. Eng.* **2005**, *9*, 741–753. [CrossRef]
22. Mucha, W.; Kuś, W. Application of mode superposition to hybrid simulation using real time finite element method. *Mechanika* **2017**, *23*, 673–677. [CrossRef]

23. Mucha, W.; Kuś, W. Mountain bicycle frame testing as an example of practical implementation of hybrid simulation using RTFEM. In *Proceedings of the 22nd International Conference on Computer Methods in Mechanics (CMM2017), Lublin, Poland, 13–16 September 2017*; AIP Conference Proceedings 1922; AIP Publishing: Melville, SK, Canada, 2018; p. 140002.
24. Mucha, W.; Kuś, W. Metamodeling as a model order reduction technique in hybrid simulation using RTFEM. In *Proceedings of the 14th International Conference Mechatronic Systems and Materials, Zakopane, Poland, 4–6 June 2018*; AIP Conference Proceedings 2029; AIP Publishing: Melville, SK, Canada, 2018; p. 020045.
25. Song, Y.; Yu, Z. Springback prediction in T-section beam bending process using neural networks and finite element method. *Arch. Civ. Mech. Eng.* **2013**, *13*, 229–241. [CrossRef]
26. KılıÇ, N.; Ekici, B.; Hartomacıoğlu, S. Determination of penetration depth at high velocity impact using finite element method and artificial neural network tools. *Def. Technol.* **2015**, *11*, 110–122. [CrossRef]
27. Artero-Guerrero, J.A.; Pernas-Sánchez, J.; Martín-Montal, J.; Varas, D.; López-Puente, J. The influence of laminate stacking sequence on ballistic limit using a combined Experimental/FEM/Artificial Neural Networks (ANN) methodology. *Compos. Struct.* **2018**, *183*, 299–308. [CrossRef]
28. Lv, C.; Xing, Y.; Zhang, J.; Na, X.; Li, Y.; Liu, T.; Cao, D.; Wang, F.-Y. Levenberg-marquardt backpropagation training of multilayer neural networks for state estimation of a safety critical cyber-physical system. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3436–3446. [CrossRef]
29. Du, Y.-C.; Stephanus, A. Levenberg-marquardt neural network algorithm for degree of arteriovenous fistula stenosis classification using a dual optical photoplethysmography sensor. *Sensors* **2018**, *18*, 2322. [CrossRef] [PubMed]
30. Burden, F.; Winkler, D. Bayesian regularization of neural networks. In *Artificial Neural Networks: Methods and Applications*; Livingstone, D.J., Ed.; Humana Press: Totowa, NJ, USA, 2009; pp. 23–42. ISBN 978-1-60327-101-1.
31. Sun, Z.; Chen, Y.; Li, X.; Qin, X.; Wang, H. A Bayesian regularized artificial neural network for adaptive optics forecasting. *Opt. Commun.* **2017**, *382*, 519–527. [CrossRef]
32. Chel, H.; Majumder, A.; Nandi, D. Scaled conjugate gradient algorithm in neural network based approach for handwritten text recognition. In *Proceedings of the Trends in Computer Science, Engineering and Information Technology, Tirunelveli, India, 23–25 September 2011*; Nagamalai, D., Renault, E., Dhanuskodi, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 196–210.
33. Møller, M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **1993**, *6*, 525–533. [CrossRef]
34. Hagan, M.T.; Menhaj, M.B. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. Neural Netw.* **1994**, *5*, 989–993. [CrossRef] [PubMed]
35. Nguyen, D.; Widrow, B. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks, San Diego, CA, USA, 17–21 June 1990; Volume 3, pp. 21–26.