aci avances en ciencias e ingenierías

Artículo/Article
Sección/Section C

# Potential of neural networks for maximum displacement predictions in railway beams on frictionally damped foundations

**Miguel Abambres[1, 2*], Rita Corrêa[3], António Pinto da Costa[4], Fernando Simões[4]**

[1] R&D, Abambres' Lab, 1600-275 Lisboa, Portugal
[2] Escola de Tecnologias e Engenharia, Instituto Superior de Educação e Ciências (ISEC), 1750-142 Lisboa, Portugal
[3] GRID International, Consulting Engineers, 1050-125 Lisboa, Portugal
[4] CERIS and Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal
* Autor para correspondencia / Corresponding author, e-mail: abambres@netcabo.pt

# Potencial de las Redes Neuronales para Predicción de Desplazamientos Máximos de Vigas Ferroviarias Sobre Fundaciones Amortiguadas por Fricción

## Resumen

Por las simulaciones por elementos finitos (EF) para el análisis dinámico de vigas ferroviarias sobre fundaciones amortiguadas por fricción requirieren (i) mucho tiempo, bien como (ii) conocimiento y software avanzados que ultrapasan los recursos disponibles en empresas de diseño en Ingeniería Civil, este articulo enseña el potencial de las Redes Neuronales Artificiales (ANN en inglés) para predecir de forma efectiva los desplazamientos máximos y la velocidad critica en vigas ferroviarias sobre acción de fuerzas móviles. Cuatro modelos basados en ANN son propuestos, uno por cada rango de velocidades ([50, 175] U [250, 300] m/s; ]175, 250[ m/s) y por cada tipo de desplazamiento (para arriba o para bajo). Cada modelo es función de dos variables independientes, el parámetro friccional y la velocidad de la fuerza. Entre todos los modelos y los 663 datos utilizados, ha sido obtenido un error máximo de 5.4 % cuando comparadas las soluciones basadas en ANN y EF. Mientras la última tiene asociado un tiempo promedio de computación, por cada punto/dato, de miles de segundos, la anterior ni un milisegundo necesita. Este estudio fue un paso importante hacia el desarrollo de métodos analíticos basados en ANN que sean más versátiles (i.e., incluyendo más tipos de variables independientes) para el mismo tipo de problema.

**Palabras clave:** Redes Neuronales Artificiales; Vigas Ferroviarias; Fundaciones Amortiguadas por Fricción; Fuerzas Móviles; Desplazamientos Máximos; Velocidad Critica

## Abstract

Since the use of finite element (FE) simulations for the dynamic analysis of railway beams on frictionally damped foundations are (i) very time consuming, and (ii) require advanced know-how and software that go beyond the available resources of typical civil engineering firms, this paper aims to demonstrate the potential of Artificial Neural Networks (ANN) to effectively predict the maximum displacements and the critical

**Editado por / Edited by:**
Eva O.L. Lantsoght

**Recibido / Received:**
12/02/2019

**Aceptado / Accepted:**
18/03/2019

**Publicado en línea / Published online:**
25/07/2019

velocity in railway beams under moving loads. Four ANN-based models are proposed, one per load velocity range ([50, 175] ∪ [250, 300] m/s; ]175, 250[ m/s) and per displacement type (upward or downward). Each model is function of two independent variables, a frictional parameter and the load velocity. Among all models and the 663 data points used, a maximum error of 5.4 % was obtained when comparing the ANN- and FE-based solutions. Whereas the latter involves an average computing time per data point of thousands of seconds, the former does not even need a millisecond. This study was an important step towards the development of more versatile (i.e., including other types of input variables) ANN-based models for the same type of problem.

**Keywords:** Artificial Neural Networks; Railway Beam; Frictionally Damped Foundation; Moving Load; Maximum Displacements; Critical Velocity.

## INTRODUCTION

The study of the dynamic behavior of beams on foundations subjected to moving loads with possible applications in high-speed railway track design has been a topic of interest in the literature. In particular, the existence of a critical velocity of the load for which the beam's oscillation amplitudes become very large has been demonstrated [1]. The serviceability of a high-speed railway track depends on the limitation of these dynamic amplifications [2-3], that is, depends on the ability of its substructure to dissipate the energy transmitted by the moving loads. This substructure is composed of many stones of several sizes and shapes, interacting through surfaces that are almost always in persistent frictional contact.

Since the main mechanism governing the interaction between the infrastructure's constituents is unilateral frictional contact mechanics, a novel non-smooth foundation model, which is closer to the true frictional dissipative nature of the ballast than the viscous model, was proposed in [4]. The goal of that study was to generalize, for more realistic behaviors, the analyses in [5-7] so that they could be of interest for high-speed railway engineers. Thus, a finite element (FE) program was developed within a MATLAB [8] environment to analyze the dynamic behavior of a Euler-Bernoulli beam on a foundation composed of continuous distributions of linear elastic springs and Coulomb frictional dissipators/dampers. This "Winkler-Coulomb"-type foundation is represented in Fig. 1 under a simply supported beam. It assumes that, in parallel with a linear elastic Winkler foundation, there is a reaction per unit length that, at each cross section of the beam, obeys to Coulomb's friction law: the frictional dissipators apply an instantaneous reaction per unit length at cross section $x$ and time instant $t$ depending on the sign of the transverse velocity of that cross section, i.e. where (i) $f_u$ denotes the maximum force per unit length that the system of frictional dissipators of the foundation may support, (ii) is the transverse velocity of the cross section, and (iii) if and if . The expression of the reaction force is an algebraic inclusion [9, 10], meaning that at the instants of vanishing velocity the reaction may belong to an interval and at the instants of velocity sign change the reaction per unit length is discontinuous. This reaction is very different from the one provided by a continuous distribution of the traditional linear viscous dampers, , where $c$ is the viscous damping coefficient per unit length. In both cases the reaction opposes the velocity but, while viscous damping provides a reaction that is proportional to the local

**doi:** http://dx.doi.org/10.18272/aci.v11i2.1365

velocity itself, the frictional reaction is limited to the interval $[-f_u, +f_u]$ and is independent of the magnitude of the velocity (see Fig. 2 in [4]). In that study a time stepping algorithm specially designed to deal with non-smooth dynamical systems was for the first time applied to beams on distributed friction foundations and new conclusions on critical velocities, maximal displacements and dynamic amplification factors were drawn.

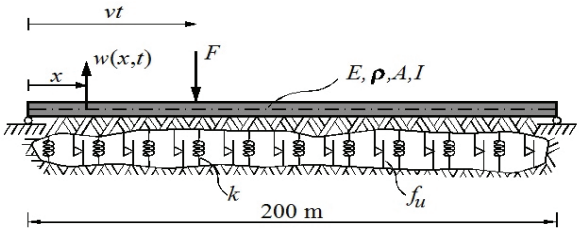


**FIGURE 1.** Beam on a frictionally damped continuous foundation under a moving load.

Since the FE analyses in [4] are (i) very time consuming (thus unfeasible for fast engineering estimations), and (ii) require advanced know-how and software that go beyond the available resources of typical civil engineering firms, this paper aims to demonstrate the potential of Artificial Neural Networks (ANN) to effectively predict the maximum displacements in railway tracks on frictional foundations, as function of the frictional parameter ($fu$) and load velocity ($v$). This is an important step towards the future development of much more versatile ANN-based analytical models for the same type of problem. The difference will be the inclusion of more independent (input) variables, such as the foundation stiffness modulus $k$, the applied load magnitude $F$, and geometrical/mechanical properties of the railway beam (see Fig. 1).

## FE-BASED MODEL AND DATA GATHERING

The authors considered a horizontal simply supported linear elastic Euler-Bernoulli beam (see Fig. 1) of 200 m length, cross-sectional area and central moment of inertia respectively equal to $A$ and $I$, mass density $\rho$ and Young's modulus E. The properties of the beam are summarized in Table 1 and correspond to the ones of a UIC60 rail. Previous studies [5-7] showed that a 200 m simply supported beam is a good finite length model to approximate the behavior of an infinite beam on elastic foundation with a single moving load. The beam is supposed to be connected to a fixed foundation bed by a system of linear elastic springs, with stiffness per unit length denoted by $k$, associated in parallel with a continuous distribution of friction dampers with a maximum force per unit length $f_u$. A downward concentrated force $F = 83.4$ kN, corresponding to half of the load per axle of a Thalys high speed train locomotive, acts on the beam moving from left to right at a constant velocity $v$ (numerical results considered $v$ ranging between 50 m/s and 300 m/s at intervals of 5 m/s). The motion of the beam is governed by a partial differential inclusion (eq. (2) in [4]) that is (i) semi-discretized in space, using the finite element method, as a system of ordinary differential inclusions (eq. (5) in [4]), and (ii) integrated in time using a special implementation of the Non-smooth Contact Dynamics Method (NSCD) developed by Moreau [11] and Jean [12], adapted to distributed Coulomb friction. The stiffness modulus of the foundation is $k = 250$ kN/

m2 and thirteen different values of the maximum force per unit length of the frictional dampers $f_u$ (0, 1, 2, 3, 3.5, 4, 5, 6, 7, 7.5, 8, 9 and 10 kN/m) are considered. All simulations employed a 200-element uniform mesh and a time step $h$ such that $vh$ = 0.1 m (i.e., in each time step the load progresses 10 cm independently of its velocity $v$). The suitability of the mesh refinement and time step was assessed in [4]. The self-weight of the beam is not considered, as in [4], in order to allow the comparison of results.

The maximum upward (positive $w_{max}$) and downward (negative $w_{max}$) transverse displacements of the beam under moving load are obtained as function of the frictional parameter $f_u$ and load velocity $v$, using the FE program mentioned before. For each pair of $f_u$ and $v$ values, the computational time to obtain the corresponding upward and downward maximum displacements ranged between 4000 and 4800 seconds, when using a computer with an Intel® Core™ i5-3470 CPU @ 3.20 GHz, 8 GB of RAM, and a 64-bit Operating System.

The aforementioned data was then used to feed the neural networks analyzed in this work. After some experiments, and aiming to obtain sufficiently accurate (maximum error smaller or around 5%) models, it was decided to develop four independent ANN-based models, namely for: (i) negative $w_{max}$ ($v$ = [50, 175] ∪ [250, 300] m/s), (ii) negative $w_{max}$ ($v$ = ]175, 250[ m/s), (iii) positive $w_{max}$ ($v$ = [50, 175] ∪ [250, 300] m/s), and (iv) positive $w_{max}$ ($v$ = ]175, 250[ m/s). The choice for these data ranges is easily understandable if one observes the last figure in this manuscript. Each model is described in sub-sections 4.1-4.4, respectively, and characterized by two independent ($f_u$, $v$) and one dependent (positive or negative $w_{max}$) variables. The datasets used in the development and final testing of each model are available in [13].

**TABLE 1.** Properties of the UIC60 rail [4].

| Property | Value |
|---|---|
| Young's modulus ($E$) | 210 GPa |
| Central area moment of inertia ($I$) | $3055 \times 10^{-8}$ m$^4$ |
| Cross-sectional area ($A$) | $7684 \times 10^{-6}$ m$^2$ |
| Density ($\rho$) | 7800 kg/m$^3$ |

# ARTIFICIAL NEURAL NETWORKS

## Brief Introduction

Since artificial neural networks (ANN) are a machine learning technique widely described in the scientific literature, please refer to [14] for a more thorough presentation of this topic. ANN's computing power makes them suitable to efficiently solve small to large-scale complex problems, which can be attributed to their (i) massively parallel distributed structure and (ii) ability to learn and generalize, i.e, produce reasonably accurate outputs for inputs not used during the learning phase. The general ANN structure consists of several nodes disposed in $L$ vertical layers (input layer, hidden layers, and output layer)

and connected between them, as depicted in Fig. 2. Associated to each node in layers 2 to ***L***, also called neuron, is a linear or nonlinear transfer function, which receives an input and transmits an output. In this work, only feedforward networks were implemented, i.e. output from any node is only transmitted to nodes located in subsequent layers.

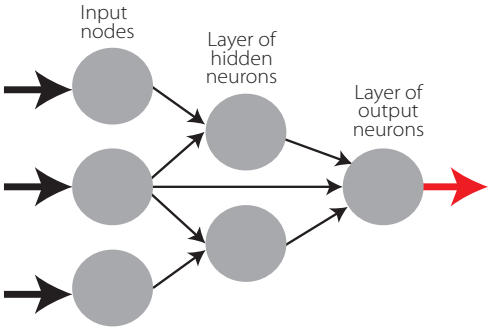


**FIGURE 2.** Example of a feedforward network with node structure 3-2-1

## Learning

Learning is nothing else than determining network unknown parameters through some algorithm in order to minimize network's performance measure, typically a function of the difference between predicted and target (desired) outputs. When ANN learning is iterative in nature, it consists of three phases: (i) training, (ii) validation, and (iii) testing. From previous knowledge, examples or data points are selected to train the network, grouped in the so-called training dataset. During an iterative learning, while the training dataset is used to tune network unknowns, a process of cross-validation takes place by using a set of data completely distinct from the training counterpart (the validation dataset), so that the generalization performance of the network can be attested. Once 'optimum' network parameters are determined, typically associated to a minimum of the validation performance curve (called early stop – see Fig. 3), many authors still perform a final assessment of model's accuracy, by presenting to it a third fully distinct dataset called 'testing'. Heuristics suggests that early stopping avoids overfitting, i.e. the loss of ANN's generalization ability.

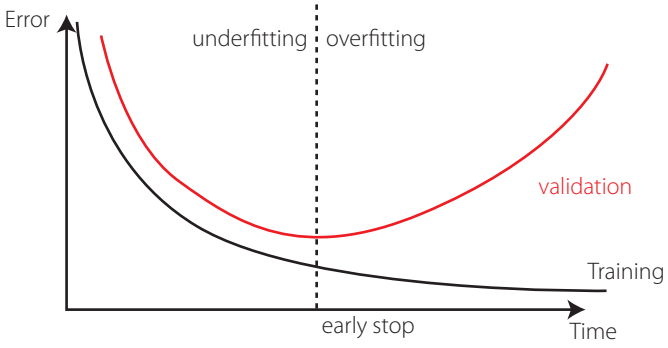


**FIGURE 3.** Assessing ANN's generalization ability via cross-validation.

## Implemented ANN features

The 'behavior' of any ANN depends on many 'features', having been implemented 15 ANN features in this work (including data pre/post processing ones). For those features, it is important to bear in mind that no ANN guarantees good approximations via extrapolation, i.e. the implemented ANNs should not be applied outside the input variable ranges used for network training. Since there are no objective rules dictating which method per feature guarantees the best network performance for a specific problem, an extensive parametric analysis (composed of nine parametric sub-analyses) was carried out to find 'the optimum' net design. A description of all methods/formulations implemented for each ANN feature (see Tabs. 2-4) – they are a selection from state of art literature on ANNs, including both traditional and promising modern techniques, can be found in previous published works (e.g., [14, 15]) – the reader might need to go through it to fully understand the meaning of all variables reported in this manuscript. The whole work was coded in MATLAB [8], making use of its neural network toolbox when dealing with popular learning algorithms (1-3 in Tab. 4). Each parametric sub-analysis (SA) consists of running all feasible combinations (also called 'combos') of pre-selected methods for each ANN feature, in order to get performance results for each designed net, thus allowing the selection of the best ANN according to a certain criterion. The best network in each parametric SA is the one exhibiting the smallest average relative error (called performance) for all learning data.

**TABLE 2.** Implemented ANN features (F) 1-5.

| FEATURE METHOD | F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|---|
| | Qualitative Var Represent | Dimensional Analysis | Input Dimensionality Reduction | % Train- Valid- Test | Input Normalization |
| 1 | Boolean Vectors | Yes | Linear Correlation | 80-10-10 | Linear Max Abs |
| 2 | Eq Spaced in ]0,1] | No | Auto-Encoder | 70-15-15 | Linear [0, 1] |
| 3 | - | - | - | 60-20-20 | Linear [-1, 1] |
| 4 | - | - | Ortho Rand Proj | 50-25-25 | Nonlinear |
| 5 | - | - | Sparse Rand Proj | - | Lin Mean Std |
| 6 | - | - | No | - | No |

**TABLE 3.** Implemented ANN features (F) 6-10.

| FEATURE METHOD | F6 | F7 | F8 | F9 | F10 |
|---|---|---|---|---|---|
| | Output Transfer | Output Normalization | Net Architecture | Hidden Layers | Connectivity |
| 1 | Logistic | Lin [a, b] = 0.7[$\varphi_{min}$, $\varphi_{max}$] | MLPN | 1 HL | Adjacent Layers |
| 2 | - | Lin [a, b] = 0.6[$\varphi_{min}$, $\varphi_{max}$] | RBFN | 2 HL | Adj Layers + In-Out |
| 3 | Hyperbolic Tang | Lin [a, b] = 0.5[$\varphi_{min}$, $\varphi_{max}$] | - | 3 HL | Fully-Connected |
| 4 | - | Linear Mean Std | - | - | - |
| 5 | Bilinear | No | - | - | - |
| 6 | Compet | - | - | - | - |
| 7 | Identity | - | - | - | - |

**TABLE 4.** Implemented ANN features (F) 11-15.

| FEATURE METHOD | F11 | F12 | F13 | F14 | F15 |
|---|---|---|---|---|---|
| | Hidden Transfer | Parameter Initialization | Learning Algorithm | Performance Improvement | Training Mode |
| 1 | Logistic | Midpoint (W) + Rands (b) | BP | NNC | Batch |
| 2 | Identity-Logistic | Rands | BPA | - | Mini-Batch |
| 3 | Hyperbolic Tang | Randnc (W) + Rands (b) | LM | - | Online |
| 4 | Bipolar | Randnr (W) + Rands (b) | ELM | - | - |
| 5 | Bilinear | Randsmall | mb ELM | - | - |
| 6 | Positive Sat Linear | Rand [-Δ, Δ] | I-ELM | - | - |
| 7 | Sinusoid | SVD | CI-ELM | - | - |
| 8 | Thin-Plate Spline | MB SVD | - | - | - |
| 9 | Gaussian | - | - | - | - |
| 10 | Multiquadratic | - | - | - | - |
| 11 | Radbas | - | - | - | - |

## Network Performance Assessment

Several types of results were computed to assess network outputs, namely (i) maximum error, (ii) % errors greater than 3%, and (iii) performance, which are defined next. All abovementioned errors are relative errors (expressed in %) based on the following definition, concerning a single output variable and data pattern,

$$e_{qp} = 100 \left| \frac{d_{qp} - y_{qLp}}{d_{qp}} \right| \quad (1)$$

where (i) $d_{qp}$ is the $q^{th}$ desired (or target) output when pattern p within iteration $i$ ($p$=1,…, $P_i$) is presented to the network, and (ii) $y_{qLp}$ is net's $q^{th}$ output for the same data pattern. Moreover, denominator in eq. (1) is replaced by 1 whenever $|d_{qp}| < 0.05 - d_{qp}$ in the nominator keeps its real value. This exception to eq. (1) aims to reduce the apparent negative effect of large relative errors associated to target values close to zero.

### Maximum Error

This variable measures the maximum relative error, as defined by eq. (1), among all output variables and learning patterns.

### Percentage of Errors > 3%

This variable measures the percentage of relative errors, as defined by eq. (1), among all output variables and learning patterns, that are greater than 3%.

### Performance

In functional approximation problems, network performance is defined as the average relative error, as defined in eq. (1), among all output variables and data patterns being evaluated (e.g., training, all data).

### Software Validation

Several benchmark datasets/functions were used to validate the developed software, involving low- to high-dimensional problems and small to large volumes of data. Due to paper length limit, validation results are not presented herein but they were made public online [16].

## RESULTS AND PROPOSED ANN-BASED MODELS

Aiming to reduce the computing time by cutting in the number of combos to be run – note that all features combined lead to hundreds of millions of combos, the whole parametric simulation was divided into nine parametric sub-analyses (SAs), where in each one feature 7 only takes a single value. This measure aims to make the performance ranking of all combos within each SA analysis more 'reliable', since results used for comparison are based on target and output datasets as used in ANN training and yielded by the designed network, respectively (they are free of any postprocessing that eliminates output normalization effects on relative error values). Whereas (i) the 1st and 2nd SAs aimed to select the best methods from features 1, 2, 5, 8 and 13 (all combined), while adopting a single popular method for each of the remaining features ($F_3$: 6, $F_4$: 2, $F_6$: {1 or 7}, $F_7$: 1, $F_9$: 1, $F_{10}$: 1, $F_{11}$: {3, 9 or 11}, $F_{12}$: 2, $F_{14}$: 1, $F_{15}$: 1 – see Tables 2-4) – SA 1 involved learning algorithms 1-3 and SA 2 involved the ELM-based counterpart, (ii) the 3rd – 7th SAs combined all possible methods from features 3, 4, 6 and 7, and concerning all other features, adopted the methods integrating the best combination from the aforementioned first SA, (iii) the 8th SA combined all possible methods from features 11, 12 and 14, and concerning all other features, adopted the methods integrating the best combination (results compared after postprocessing) among the previous five sub-analyses, and lastly (iv) the 9th SA combined all possible methods from features 9, 10 and 15, and concerning all other features, adopted the methods integrating the best combination from the previous analysis. The microprocessors used for ANN simulations have the following features: OS: Win10Home 64bits, RAMs: 48.0 / 128 GB, Local Disk Memory: 1 TB, CPUs: Intel® Core™ i7 8700K @ 3.7-4.7 GHz / i9 7960X @ 2.8-4.2 GHz. The datasets used in the development and final testing of all ANN models are available in [13]. In what follows, the parametric analysis results and the proposed ANN-based models are presented for each of the four problems addressed, namely: (i) negative $w_{max}$ ($v$ = [50, 175] ∪ [250, 300] m/s), (ii) negative $w_{max}$ ($v$ = ]175, 250[ m/s), (iii) positive $w_{max}$ ($v$ = [50, 175] ∪ [250, 300] m/s), and (iv) positive $w_{max}$ ($v$ = ]175, 250[ m/s). It is important to note that, in this manuscript, whenever a vector is added to a matrix, it means the former is added to all columns of the latter (valid in MATLAB).

### Negative $w_{max}$ ($v$ = [50, 175] ∪ [250, 300] m/s)

ANN feature methods used in the best combo from each of the abovementioned nine parametric SAs are specified in Table 5 (see Tables 2-4). Table 6 shows the corresponding relevant results for those combos and the 481-point final testing dataset (which includes the ANN learning/development dataset), namely (i) maximum error, (ii) percentage of errors larger than 3%, (iii) performance (all described in sub-section 3.4, and evaluated for all learning data), (iv) total number of hidden nodes in the model, and (v) average

computing time per example (including data pre- and post-processing). All results shown in Table 6 are based on target and output datasets computed in their original format, i.e. free of any transformations due to output normalization and/or dimensional analysis. Summing up the ANN feature combinations for all parametric SAs, a total of 219 combos were run for this problem.

**TABLE 5.** ANN feature (F) methods used in the best combo from each parametric sub-analysis (SA).

| SA | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 6 | 2 | 5 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 2 | 1 | 2 | 6 | 2 | 5 | 7 | 1 | 1 | 1 | 1 | 3 | 2 | 5 | 1 | 3 |
| 3 | 1 | 2 | 6 | 2 | 5 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 4 | 1 | 2 | 6 | 2 | 5 | 1 | 2 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 5 | 1 | 2 | 6 | 4 | 5 | 1 | 3 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 6 | 1 | 2 | 6 | 2 | 5 | 7 | 4 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 7 | 1 | 2 | 6 | 4 | 5 | 3 | 5 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 8 | 1 | 2 | 6 | 4 | 5 | 3 | 5 | 1 | 1 | 1 | 3 | 5 | 3 | 1 | 3 |
| 9 | 1 | 2 | 6 | 4 | 5 | 3 | 5 | 1 | 3 | 3 | 3 | 5 | 3 | 1 | 3 |

The proposed model is the one, among the best ones from all parametric SAs, exhibiting the lowest maximum error (SA 7 – a Neural Network Composite (NNC)). Aiming to allow implementation of this model by any user, all variables/equations required for (i) data preprocessing, (ii) ANN simulation, and (iii) data postprocessing, are presented in the following sub-sections. The proposed model is an NNC made of 7 ANNs with architecture MLPN and a distribution of nodes/layer equal to 2-8-1 for every network. Concerning connectivity, all networks are partially-connected, and the hidden and output transfer functions are all Hyperbolic Tangent. All networks were trained using the LM algorithm. After design, the average NNC computing time concerning the presentation of a single example (including data pre/postprocessing) is 3.12E-05 s; Fig. 4 depicts a simplified scheme of each NNC network. Finally, all relevant performance results concerning the proposed NNC are illustrated in sub-section 4.1.4.

**TABLE 6.** Performance results for the best design from each parametric SA and for the final testing dataset (includes the ANN learning/development dataset): ANN and NNC.

| | ANN | | | | |
|---|---|---|---|---|---|
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | 5.6 | 1.0 | 3.7 | 8 | 7.95E-05 |
| 2 | 15.3 | 2.3 | 20.4 | 40 | 2.89E-05 |
| 3 | 5.7 | 1.0 | 3.7 | 8 | 2.88E-05 |
| 4 | 6.3 | 0.9 | 3.5 | 8 | 2.91E-05 |
| 5 | 5.7 | 1.0 | 2.7 | 8 | 3.64E-05 |
| 6 | 5.3 | 1.0 | 4.8 | 8 | 4.86E-05 |

| | | | | | |
|---|---|---|---|---|---|
| 7 | 6.1 | 1.0 | 4.6 | 8 | 2.75E-05 |
| 8 | 7.5 | 1.0 | 4.4 | 8 | 4.66E-05 |
| 9 | 3.2 | 0.6 | 0.2 | 9 | 4.36E-05 |

| | NNC | | | | |
|---|---|---|---|---|---|
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | - | - | - | - | - |
| 2 | - | - | - | - | - |
| 3 | - | - | - | - | - |
| 4 | - | - | - | - | - |
| 5 | - | - | - | - | - |
| 6 | - | - | - | - | - |
| 7 | 2.7 | 0.4 | 0.0 | 8 | 3.12E-05 |
| 8 | 5.8 | 0.8 | 2.3 | 8 | 4.89E-05 |
| 9 | - | - | - | - | - |

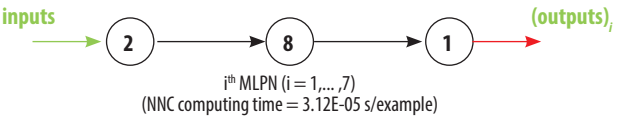


**FIGURE 4.** Proposed NNC made of 7 partially-connected MLPNs – simplified scheme.

## Input Data Preprocessing

For future use of the proposed NNC to simulate new data $Y_{1,sim}$ (a 2 x $P_{sim}$ matrix) concerning $P_{sim}$ patterns, the same data preprocessing (if any) performed before training must be applied to the input dataset. That preprocessing is defined by the methods used for ANN features 2, 3 and 5 (respectively 2, 6 and 5 – see Table 2). Next, the necessary preprocessing to be applied to $Y_{1,sim}$, concerning features 2, 3 and 5, is fully described.

### *Dimensional Analysis and Dimensionality Reduction*

Since no dimensional analysis (***d.a.***) nor dimensionality reduction (***d.r.***) were carried out, one has

$$\left\{Y_{1,sim}\right\}_{d.r.}^{after} = \left\{Y_{1,sim}\right\}_{d.a.}^{after} = Y_{1,sim} \tag{2}$$

### *Input Normalization*

After input normalization, the new input dataset $\{Y_{1,sim}\}_{n}^{after}$ is defined as function of the previously determined $\{Y_{1,sim}\}_{dr}^{after}$, and they have the same size, reading

$$\left\{Y_{1,sim}\right\}_n^{after} = \left(\left\{Y_{1,sim}\right\}_{d.r}^{after} - \mathrm{INP}(:,1)\right) ./ \mathrm{INP}(:,2)$$

$$\mathrm{INP} = \begin{bmatrix} 5000 & 3166.27801301405 \\ 161.527777777778 & 82.2184600671207 \end{bmatrix} \quad (3)$$

where one recalls that (i) INP(:, j) represents column j of matrix INP, and (ii) operator './' divides row i in the numerator by INP(i, 2).

### ANN-Based Analytical Model

Once determined the preprocessed input dataset $\{Y_{1,sim}\}_n^{after}$ (a 2 x $P_{sim}$ matrix), the next step is to present it to the proposed NNC to obtain the predicted output dataset $\{Y_{3,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), which will be given in the same preprocessed format of the target dataset used in learning. In order to convert the predicted outputs to their 'original format' (i.e., without any transformation due to normalization or dimensional analysis), some postprocessing might be needed, as discussed in 4.1.3. Next, the mathematical representation of the proposed NNC is given, so that any user can implement it to determine $\{Y_{3,sim}\}_n^{after}$, thus contributing to diminish the generalized opinion that ANNs are 'black boxes':

$$\left\{Y_{3,sim}\right\}_n^{after} = \left\{Y_{3,sim}\right\}_{n(0)}^{after} + \sum_{i=1}^{6} \left\{Y_{3,sim}\right\}_{n(i)}^{after} \quad (4)$$

being

$$Y_{2(i)} = \varphi_2\left(W_{1-2(i)}^T \left\{Y_{1,sim}\right\}_n^{after} + b_{2(i)}\right)$$

$$\left\{Y_{3,sim}\right\}_{n(i)}^{after} = \varphi_3\left(W_{2-3(i)}^T Y_{2(i)} + b_{3(i)}\right) \quad (5)$$

where i = 0,..., 6 and

$$\varphi_2 = \varphi_3 = \varphi(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \quad (6)$$

and arrays $W_{j\text{-}s(i)}$ and $b_{s(i)}$ are stored online in [17], aiming to avoid an overlong article and ease model's implementation by any interested reader.

### Output Data Postprocessing

In order to transform the output dataset obtained by the proposed NNC, $\{Y_{3,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), to its original format ($Y_{3,sim}$), i.e. without the effects of dimensional analysis and/or output normalization (possibly) taken in target dataset preprocessing prior training, one has

$$Y_{3,sim} = \left\{Y_{3,sim}\right\}_n^{after} \quad (7)$$

adopted in the proposed model.

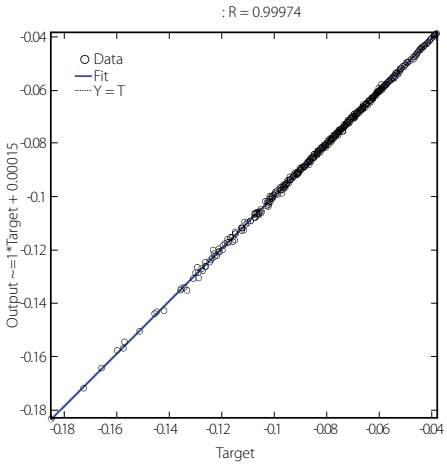


**FIGURE 5.** Regression plot for the proposed NNC.

## Performance Results

Finally, the results yielded by the proposed NNC for the 481-point final testing dataset (which includes the ANN learning/development counterpart), in terms of performance variables defined in sub-section 3.4, are presented in this sub-section in the form of two graphs: (i) a regression plot (Fig. 5), where network target and output data are plotted, for each data point, as *x*- and *y*- coordinates, respectively – a measure of quality is given by the Pearson Correlation Coefficient (*R*); and (ii) a plot (Fig. 6) indicating (for all data) the (ii1) maximum error, (ii2) percentage of errors larger than 3%, and (ii3) average error (called performance).

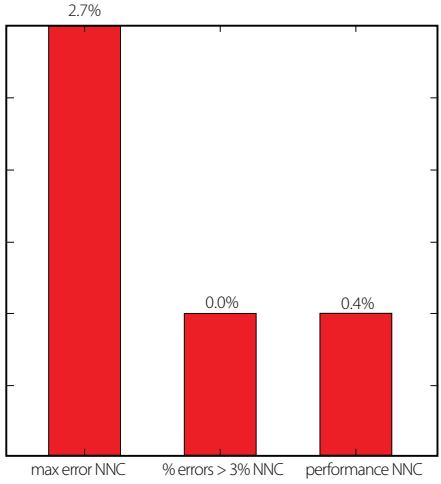


**FIGURE 6.** Maximum and average (performance) errors for the proposed NNC.

## Negative $w_{max}$ ($v$ = ]175, 250[ m/s)

ANN feature methods used in the best combo from each of the abovementioned nine parametric SAs are specified in Table 7 (numbers represent the method number as in Tables 2-4). Table 8 shows the corresponding relevant results for those combos and the 208-point final testing dataset (which includes the ANN learning/development dataset), namely (i) maximum error, (ii) percentage of errors larger than 3%, (iii) performance (all described in sub-section 3.4, and evaluated for all learning data), (iv) total number of hidden nodes in the model, and (v) average computing time per example (including data pre- and post-processing). All results shown in Table 8 are based on target and output datasets computed in their original format, i.e. free of any transformations due to output normalization and/or dimensional analysis. Summing up the ANN feature combinations for all parametric SAs, a total of 204 combos were run for this problem.

The proposed model is the one, among the best ones from all parametric SAs, exhibiting the lowest maximum error (SA 9 - a Neural Network Composite (NNC)). Aiming to allow implementation of this model by any user, all variables/equations required for (i) data preprocessing, (ii) ANN simulation, and (iii) data postprocessing, are presented in the following sub-sections. The proposed model is an NNC made of 3 ANNs with architecture RBFN and a distribution of nodes/layer given by 2-3-3-3-1 for every network. Concerning connectivity, all networks are partially-connected (see Fig. 7), and the hidden and output transfer functions are all Gaussian RBF and Hyperbolic Tangent, respectively. All networks were trained using the LM algorithm. After design, the average NNC computing time concerning the presentation of a single example (including data pre/postprocessing) is 7.87E-05 s. Fig. 7 depicts a simplified scheme of each NNC network. Finally, all relevant performance results concerning the proposed NNC are illustrated in sub-section 4.2.4.

**TABLE 7.** ANN feature (F) methods used in the best combo from each parametric sub-analysis (SA).

| SA | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 6 | 2 | 5 | 1 | 1 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 2 | 1 | 2 | 6 | 2 | 3 | 7 | 1 | 2 | 1 | 1 | 9 | 2 | 5 | 1 | 3 |
| 3 | 1 | 2 | 6 | 1 | 5 | 1 | 1 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 4 | 1 | 2 | 6 | 1 | 5 | 1 | 2 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 5 | 1 | 2 | 6 | 3 | 5 | 1 | 3 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 6 | 1 | 2 | 6 | 1 | 5 | 7 | 4 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 7 | 1 | 2 | 6 | 1 | 5 | 3 | 5 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 8 | 1 | 2 | 6 | 1 | 5 | 3 | 5 | 2 | 1 | 1 | 11 | 2 | 3 | 1 | 3 |
| 9 | 1 | 2 | 6 | 1 | 5 | 3 | 5 | 2 | 3 | 2 | 11 | 2 | 3 | 1 | 3 |

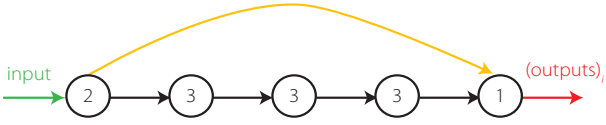


**FIGURE 7.** Proposed NNC made of 3 partially-connected RBFNs – simplified scheme.

### Input Data Preprocessing

For future use of the proposed NNC to simulate new data $Y_{1,sim}$ (a 2 x $P_{sim}$ matrix) concerning $P_{sim}$ patterns, the same data preprocessing (if any) performed before training must be applied to the input dataset. That is defined by the methods used for ANN features 2, 3 and 5 (respectively 2, 6 and 5 – see Table 2). Next, the necessary preprocessing to be applied to $Y_{1,sim}$ is fully described.

**TABLE 8.** Performance results for the best design from each parametric SA and for the final testing dataset (includes the ANN learning/development dataset): ANN and NNC.

| | ANN | | | | |
|---|---|---|---|---|---|
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | 10.7 | 2.8 | 38.9 | 8 | 1.68E-04 |
| 2 | 102.8 | 16.7 | 69.7 | 50 | 5.50E-05 |
| 3 | 12.1 | 3.1 | 43.3 | 8 | 5.26E-05 |
| 4 | 12.0 | 3.1 | 44.7 | 8 | 4.11E-05 |
| 5 | 20.7 | 3.0 | 38.5 | 8 | 8.19E-05 |
| 6 | 22.8 | 3.0 | 40.9 | 8 | 3.84E-05 |
| 7 | 18.7 | 2.7 | 32.7 | 8 | 4.47E-05 |
| 8 | 14.2 | 3.0 | 39.4 | 8 | 5.63E-05 |
| 9 | 11.4 | 1.8 | 13.9 | 9 | 5.10E-05 |

| | NNC | | | | |
|---|---|---|---|---|---|
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | - | - | - | - | - |
| 2 | - | - | - | - | - |
| 3 | - | - | - | - | - |
| 4 | - | - | - | - | - |
| 5 | 12.0 | 2.5 | 29.8 | 8 | 1.16E-04 |
| 6 | 8.0 | 1.0 | 6.3 | 8 | 6.63E-05 |
| 7 | 8.4 | 1.8 | 20.2 | 8 | 5.52E-05 |
| 8 | - | - | - | - | - |
| 9 | 5.2 | 1.1 | 4.8 | 9 | 7.87E-05 |

#### *Dimensional Analysis and Dimensionality Reduction*

Since no dimensional analysis (***d.a.***) nor dimensionality reduction (***d.r.***) were carried out, one has

$$\left\{Y_{1,sim}\right\}_{d.r.}^{after} = \left\{Y_{1,sim}\right\}_{d.a.}^{after} = Y_{1,sim} \tag{8}$$

***Input Normalization***

After input normalization, the new input dataset is defined as function of the previously determined , and they have the same size, reading

$$\left\{Y_{1,sim}\right\}_{n}^{after} = \left(\left\{Y_{1,sim}\right\}_{d.r}^{after} - \mathrm{INP}(:,1)\right) ./ \mathrm{INP}(:,2)$$
$$\mathrm{INP} = \begin{bmatrix} 5000 & 3171.29986868837 \\ 212.5 & 23.1146212230639 \end{bmatrix} \tag{9}$$

where one recalls that operator './' divides row ***i*** in the numerator by INP(***i***, 2).

## ANN-Based Analytical Model

Once determined the preprocessed input dataset $\{Y_{1,sim}\}_{n}^{after}$ (a 2 x $P_{sim}$ matrix), the next step is to present it to the proposed NNC to obtain the predicted output dataset $\{Y_{5,sim}\}_{n}^{after}$ (a 1 x $P_{sim}$ vector), which will be given in the same preprocessed format of the target dataset used in learning. To convert the predicted outputs to their 'original format' (i.e., without any transformation due to normalization or dimensional analysis), some postprocessing might be needed, as described in 4.2.3. Next, the mathematical representation of the proposed NNC is given, so that any user can implement it to determine $\{Y_{5,sim}\}_{n}^{after}$ :

$$\left\{Y_{5,sim}\right\}_{n}^{after} = \left\{Y_{5,sim}\right\}_{n(0)}^{after} + \sum_{i=1}^{2} \left\{Y_{5,sim}\right\}_{n(i)}^{after} \tag{10}$$

where ($i$ = 0, 1, 2)

$$\begin{aligned} Y_{2(i)} &= \varphi_2\left(s_{(i)}\right) \\ aux_{2(i)}(l_1, p) &= \left\| \left\{Y_{1,sim}\right\}_{n}^{after}(:, p) - W_{1-2(i)}(:, l_1) \right\| \\ s_{(i)} &= aux_{2(i)} .\mathrm{x}\ b_{2(i)} \end{aligned}$$

$$\begin{aligned} Y_{3(i)} &= \varphi_3\left(s_{(i)}\right) \\ aux_{3(i)}(l_1, p) &= \left\| Y_{2(i)}(:, p) - W_{2-3(i)}(:, l_1) \right\| \\ s_{(i)} &= aux_{3(i)} .\mathrm{x}\ b_{3(i)} \end{aligned}$$

$$\begin{aligned} Y_{4(i)} &= \varphi_4\left(s_{(i)}\right) \\ aux_{4(i)}(l_1, p) &= \left\| Y_{3(i)}(:, p) - W_{3-4(i)}(:, l_1) \right\| \\ s_{(i)} &= aux_{4(i)} .\mathrm{x}\ b_{4(i)} \end{aligned} \tag{11}$$

$$\left\{Y_{5,sim}\right\}_{n(i)}^{after} = \varphi_5\left(W_{1-5(i)}^{T}\left\{Y_{1,sim}\right\}_{n}^{after} + W_{4-5(i)}^{T} Y_{4(i)} + b_{5(i)}\right)$$

and (i) $p$=1,…, $P_{sim}$, $I_1$=1,2, 3, (ii) operator '.x' multiplies every element in row s of the first array by element s of second array (a vector), yielding an array of the same size of the first, and (iii)

$$\varphi_2 = \varphi_3 = \varphi_4 = \varphi(s) = e^{-s^2}$$
$$\varphi_5 = \varphi_5(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \quad (12)$$

Arrays $W_{j\text{-}s(i)}$ and $b_{s(i)}$ are stored online in [18].

## Output Data Postprocessing

In order to transform the output dataset obtained by the proposed NNC, $\{Y_{5,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), to its original format ($Y_{5,sim}$), i.e. without the effects of dimensional analysis and/or output normalization (possibly) taken in target dataset preprocessing prior training, one has

$$Y_{5,sim} = \left\{Y_{5,sim}\right\}_n^{after} \quad (13)$$

since no output normalization nor dimensional analysis were adopted in the proposed model.

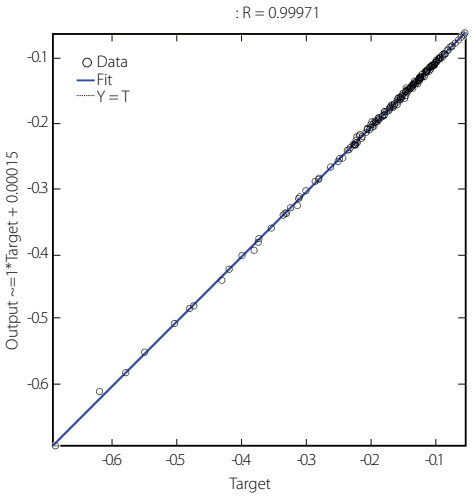


**FIGURE 8.** Regression plot for the proposed NNC.

## Performance Results

Finally, the results yielded by the proposed NNC for the 208-point final testing dataset (which includes the ANN learning/development counterpart), in terms of performance variables defined in sub-section 3.4, are presented in this sub-section in the form of two graphs: (i) a regression plot (Fig. 8), where network target and output data are plotted, for each data point, as $x$- and $y$- coordinates, respectively; and (ii) a plot (Fig. 9) indicating (for all data) the (ii1) maximum error, (ii2) percentage of errors larger than 3%, and (ii3) average error (called performance).

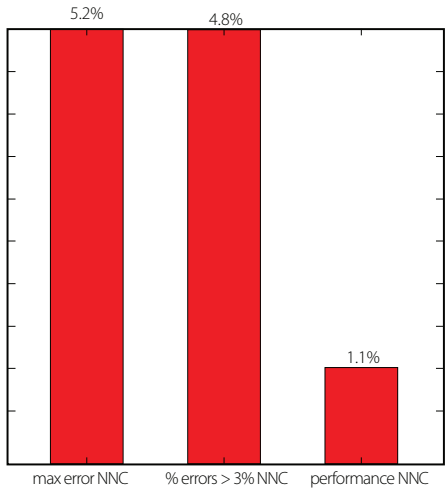


**FIGURE 9.** Maximum and average (performance) errors for the proposed NNC.

## Positive $w_{max}$ ($v = [50, 175] \cup [250, 300]$ m/s)

ANN feature methods used in the best combo from each of the abovementioned nine parametric SAs are specified in Table 9 (numbers represent the method number as in Tables 2-4). Table 10 shows the corresponding relevant results for those combos and the 481-point final testing dataset (which includes the ANN learning/development dataset), namely (i) maximum error, (ii) percentage of errors larger than 3%, (iii) performance (all described in sub-section 3.4, and evaluated for all learning data), (iv) total number of hidden nodes in the model, and (v) average computing time per example (including data pre- and post-processing). All results shown in Table 10 are based on target and output datasets computed in their original format, i.e. free of any transformations due to output normalization and/or dimensional analysis. Summing up the ANN feature combinations for all parametric SAs, a total of 219 combos were run for this problem.

**TABLE 9.** ANN feature (F) methods used in the best combo from each parametric sub-analysis (SA).

| SA | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 6 | 2 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 2 | 1 | 2 | 6 | 2 | 2 | 7 | 1 | 2 | 1 | 1 | 9 | 2 | 5 | 1 | 3 |
| 3 | 1 | 2 | 6 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 4 | 1 | 2 | 6 | 3 | 3 | 1 | 2 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 5 | 1 | 2 | 6 | 1 | 3 | 1 | 3 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 6 | 1 | 2 | 6 | 4 | 3 | 7 | 4 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 7 | 1 | 2 | 6 | 4 | 3 | 7 | 5 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 8 | 1 | 2 | 6 | 4 | 3 | 7 | 5 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 3 |
| 9 | 1 | 2 | 6 | 4 | 3 | 7 | 5 | 1 | 3 | 3 | 1 | 2 | 3 | 1 | 3 |

The proposed model is the one, among the best ones from all parametric SAs, exhibiting the lowest maximum error (SA 9). Aiming to allow implementation of this model by any user, all variables/equations required for (i) data preprocessing, (ii) ANN simulation, and (iii) data postprocessing, are presented in the following sub-sections. The proposed model is a single MLPN with 5 layers and a distribution of nodes/layer given by 2-3-3-3-1. Concerning connectivity, the network is fully-connected, and the hidden and output transfer functions are all Logistic and Identity, respectively. The network was trained using the LM algorithm (1500 epochs). After design, the average network computing time concerning the presentation of a single example (including data pre/postprocessing) is 2.49E-05 s; Fig. 10 depicts a simplified scheme of some of network key features. Finally, all relevant performance results concerning the proposed ANN are illustrated in sub-section 4.3.4.

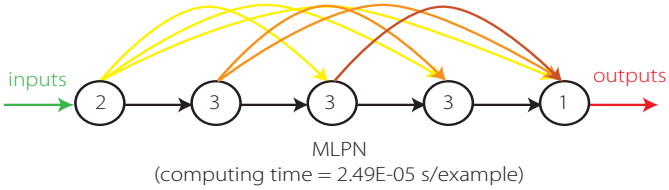


**FIGURE 10.** Proposed 2-3-3-3-1 fully-connected MLPN– simplified scheme.

**Table 10.** Performance results for the best design from each parametric SA and for the final testing dataset (includes the ANN learning/development dataset): ANN and NNC.

| | ANN | | | | |
|---|---|---|---|---|---|
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | 7.4 | 0.5 | 7.1 | 8 | 2.58E-05 |
| 2 | 8.2 | 0.7 | 8.5 | 60 | 3.43E-05 |
| 3 | 6.7 | 0.6 | 7.7 | 8 | 4.52E-05 |
| 4 | 6.9 | 0.5 | 7.5 | 8 | 2.70E-05 |
| 5 | 7.0 | 0.5 | 7.3 | 8 | 2.62E-05 |
| 6 | 6.2 | 0.5 | 8.3 | 8 | 2.65E-05 |
| 7 | 6.7 | 0.5 | 6.7 | 8 | 3.72E-05 |
| 8 | 6.4 | 0.5 | 7.5 | 8 | 2.91E-05 |
| 9 | 3.7 | 0.2 | 0.8 | 9 | 2.49E-05 |
| | **NNC** | | | | |
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | - | - | - | - | - |
| 2 | - | - | - | - | - |
| 3 | - | - | - | - | - |
| 4 | 6.9 | 0.4 | 5.8 | 8 | 2.87E-05 |
| 5 | - | - | - | - | - |
| 6 | 3.7 | 0.2 | 0.2 | 8 | 3.29E-05 |
| 7 | 6.4 | 0.5 | 6.7 | 8 | 4.16E-05 |
| 8 | 6.2 | 0.5 | 7.1 | 8 | 2.98E-05 |
| 9 | - | - | - | - | - |

## Input Data Preprocessing

For future use of the proposed ANN to simulate new data $Y_{1,sim}$ (a 2 x $P_{sim}$ matrix) concerning $P_{sim}$ patterns, the same data preprocessing (if any) performed before training must be applied to the input dataset. That preprocessing is defined by the methods used for ANN features 2, 3 and 5 (respectively 2, 6 and 3 – see Table 2). In what follows, the necessary preprocessing to be applied to $Y_{1,sim}$ is fully described.

### *Dimensional Analysis and Dimensionality Reduction*

Since no dimensional analysis (***d.a.***) nor dimensionality reduction (***d.r.***) were carried out, one has

$$\left\{Y_{1,sim}\right\}_{d.r.}^{after} = \left\{Y_{1,sim}\right\}_{d.a.}^{after} = Y_{1,sim} \tag{14}$$

### *Input Normalization*

After input normalization, the new input dataset is defined as function of the previously determined , and they have the same size, reading

$$\begin{aligned} &\left\{Y_{1,sim}\right\}_{n}^{after} = \mathrm{INP}(:,1) + rab.\mathrm{x}\left(\left\{Y_{1,sim}\right\}_{d.r}^{after} - \mathrm{INP}(:,3)\right)./den \\ &\mathrm{INP} = \begin{bmatrix} -1 & 1 & 0 & 10000 \\ -1 & 1 & 50 & 300 \end{bmatrix} \\ &rab = \mathrm{INP}(:,2) - \mathrm{INP}(:,1) \\ &den = \mathrm{INP}(:,4) - \mathrm{INP}(:,3) \end{aligned} \tag{15}$$

where one recalls that operator '.x' multiplies component ***i*** in vector ***rab*** by all components in row ***i*** of subsequent term (analogous definition holds for './').

## ANN-Based Analytical Model

Once determined the preprocessed input dataset $\{Y_{1,sim}\}_n^{after}$ (a 2 x $P_{sim}$ matrix), the next step is to present it to the proposed ANN to obtain the predicted output dataset $\{Y_{5,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), which will be given in the same preprocessed format of the target dataset used in learning. In order to convert the predicted outputs to their 'original format' (i.e., without any transformation due to normalization or dimensional analysis), some postprocessing might be needed, as described in 4.3.3. Next, the mathematical representation of the proposed ANN is given, so that any user can implement it to determine $\{Y_{5,sim}\}_n^{after}$:

$$\begin{aligned} Y_2 &= \varphi_2\left(W_{1-2}^T\left\{Y_{1,sim}\right\}_n^{after} + b_2\right) \\ Y_3 &= \varphi_3\left(W_{1-3}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-3}^T Y_2 + b_3\right) \\ Y_4 &= \varphi_4\left(W_{1-4}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-4}^T Y_2 + W_{3-4}^T Y_3 + b_4\right) \\ \left\{Y_{5,sim}\right\}_n^{after} &= \varphi_5\left(W_{1-5}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-5}^T Y_2 + W_{3-5}^T Y_3 + W_{4-5}^T Y_4 + b_5\right) \end{aligned} \tag{16}$$

where

$$\varphi_2 = \varphi_3 = \varphi_4 = \varphi(s) = \frac{1}{1+e^{-s}}$$
$$\varphi_5 = \varphi_5(s) = s \quad (17)$$

Arrays $W_{j\text{-}s}$ and $b_s$ can be found online in [19].

### Output Data Postprocessing

In order to transform the output dataset obtained by the proposed ANN, $\{Y_{5,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), to its original format ($Y_{5,sim}$), i.e. without the effects of dimensional analysis and/or output normalization (possibly) taken in target dataset preprocessing prior training, one has

$$Y_{5,sim} = \left\{Y_{5,sim}\right\}_n^{after} \quad (18)$$

since no output normalization nor dimensional analysis were adopted in the proposed model.

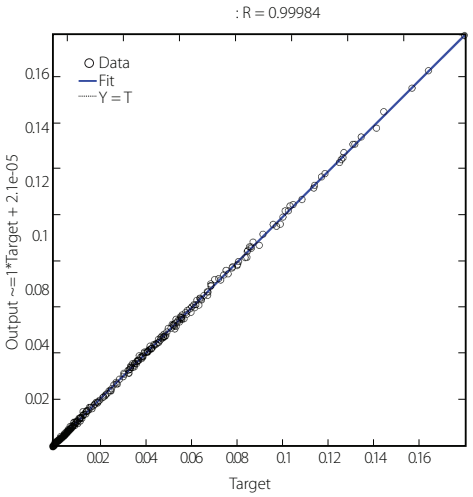


**FIGURE 11.** Regression plot for the proposed ANN.

### Performance Results

Finally, the results yielded by the proposed ANN for the 481-point final testing dataset (which includes the ANN learning/development counterpart), in terms of performance variables defined in sub-section 3.4, are presented in this sub-section in the form of two graphs: (i) a regression plot (Fig. 11), where network target and output data are plotted, for each data point, as ***x***- and ***y***- coordinates, respectively; and (ii) a plot (Fig. 12) indicating (for all data) the ($ii_1$) maximum error, ($ii_2$) percentage of errors larger than 3%, and ($ii_3$) average error (called performance).

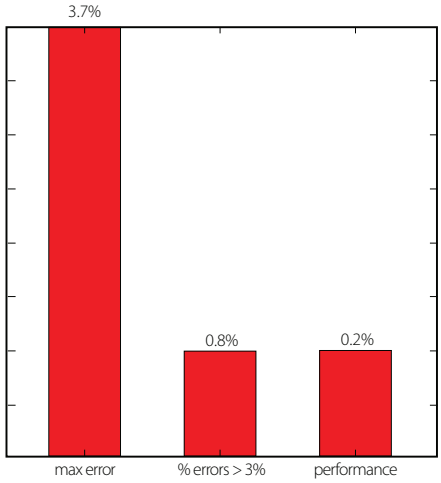


**FIGURE 12.** Maximum and average (performance) errors for the proposed ANN.

## Positive $w_{max}$ ($v = ]175, 250[$ m/s)

ANN feature methods used in the best combo from each of the abovementioned nine parametric SAs are specified in Table 11 (numbers represent the method number as in Tables 2-4). Table 12 shows the corresponding relevant results for those combos and the 208-point final testing dataset (which includes the ANN learning/development dataset), namely (i) maximum error, (ii) percentage of errors larger than 3%, (iii) performance (all described in sub-section 3.4, and evaluated for all learning data), (iv) total number of hidden nodes in the model, and (v) average computing time per example (including data pre- and post-processing). All results shown in Table 12 are based on target and output datasets computed in their original format, i.e. free of any transformations due to output normalization and/or dimensional analysis. Summing up the ANN feature combinations for all parametric SAs, a total of 219 combos were run for this problem.

**TABLE 11.** ANN feature (F) methods used in the best combo from each parametric sub-analysis (SA).

| SA | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 6 | 2 | 5 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 2 | 1 | 2 | 6 | 2 | 5 | 7 | 1 | 2 | 1 | 1 | 9 | 2 | 5 | 1 | 3 |
| 3 | 1 | 2 | 6 | 1 | 5 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 4 | 1 | 2 | 6 | 2 | 5 | 1 | 2 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 5 | 1 | 2 | 6 | 3 | 5 | 1 | 3 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 6 | 1 | 2 | 6 | 1 | 5 | 7 | 4 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 7 | 1 | 2 | 6 | 1 | 5 | 7 | 5 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 3 |
| 8 | 1 | 2 | 6 | 1 | 5 | 7 | 5 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 3 |
| 9 | 1 | 2 | 6 | 1 | 5 | 7 | 5 | 1 | 3 | 3 | 1 | 2 | 3 | 1 | 3 |

The proposed model is the one, among the best ones from all parametric SAs, exhibiting the lowest maximum error (SA 9 - a Neural Network Composite (NNC)). Aiming to allow implementation of this model by any user, all variables/equations required for (i) data preprocessing, (ii) ANN simulation, and (iii) data postprocessing, are presented in the following sub-sections. The proposed model is an NNC made of 4 ANNs with architecture MLPN and a distribution of nodes/layer given by 2-3-3-3-1 for every network. Concerning connectivity, all networks are fully-connected, and the hidden and output transfer functions are all Logistic and Identity, respectively. All networks were trained using the LM algorithm. After design, the average NNC computing time concerning the presentation of a single example (including data pre/postprocessing) is 4.08E-05 s; Fig. 13 depicts a simplified scheme of each NNC network. Finally, all relevant performance results concerning the proposed NNC are illustrated in sub-section 4.4.4.

**TABLE 12.** Performance results for the best design from each parametric SA and for the final testing dataset (includes the ANN learning/development dataset): ANN and NNC.

| | ANN | | | | |
|---|---|---|---|---|---|
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | 20.3 | 2.5 | 31.3 | 8 | 1.16E-04 |
| 2 | 135.1 | 10.3 | 61.5 | 50 | 4.17E-05 |
| 3 | 20.3 | 2.5 | 31.3 | 8 | 4.16E-05 |
| 4 | 23.7 | 2.6 | 30.8 | 8 | 4.23E-05 |
| 5 | 19.4 | 2.6 | 29.8 | 8 | 4.16E-05 |
| 6 | 20.3 | 2.9 | 35.1 | 8 | 3.50E-05 |
| 7 | 18.3 | 2.7 | 32.7 | 8 | 3.66E-05 |
| 8 | 21.3 | 2.7 | 34.1 | 8 | 5.32E-05 |
| 9 | 8.8 | 0.8 | 5.8 | 9 | 3.55E-05 |
| | NNC | | | | |
| SA | Max Error (%) | Performance All Data (%) | Errors > 3% (%) | Total Hidden Nodes | Running Time / Data Point (s) |
| 1 | - | - | - | - | - |
| 2 | - | - | - | - | - |
| 3 | - | - | - | - | - |
| 4 | - | - | - | - | - |
| 5 | - | - | - | - | - |
| 6 | 18.9 | 0.9 | 5.8 | 8 | 4.13E-05 |
| 7 | 19.1 | 1.2 | 10.6 | 8 | 4.16E-05 |
| 8 | 18.4 | 1.5 | 17.3 | 8 | 5.70E-05 |
| 9 | 5.4 | 0.4 | 1.9 | 9 | 4.08E-05 |

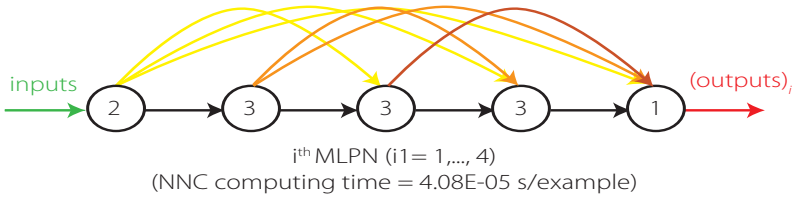


**FIGURE 13.** Proposed NNC made of 4 fully-connected MLPNs – simplified scheme.

## Input Data Preprocessing

For future use of the proposed NNC to simulate new data $Y_{1,sim}$ (a 2 x $P_{sim}$ matrix) concerning $P_{sim}$ patterns, the same data preprocessing (if any) performed before training must be applied to the input dataset. That preprocessing is defined by the methods used for ANN features 2, 3 and 5 (respectively 2, 6 and 5 – see Table 2). Next, the necessary preprocessing to be applied to $Y_{1,sim}$ is fully described.

### *Dimensional Analysis and Dimensionality Reduction*

Since no dimensional analysis (***d.a.***) nor dimensionality reduction (***d.r.***) were carried out, one has

$$\left\{Y_{1,sim}\right\}_{d.r.}^{after} = \left\{Y_{1,sim}\right\}_{d.a.}^{after} = Y_{1,sim} \tag{19}$$

### *Input Normalization*

After input normalization, the new input dataset is defined as function of the previously determined , and they have the same size, reading

$$\left\{Y_{1,sim}\right\}_{n}^{after} = \left(\left\{Y_{1,sim}\right\}_{d.r}^{after} - \mathrm{INP}(:,1)\right) ./\ \mathrm{INP}(:,2) \tag{20}$$

$$\mathrm{INP} = \begin{bmatrix} 5000 & 3171.90409844877 \\ 214.666666666667 & 22.2385441413878 \end{bmatrix}$$

where one recalls that operator './' divides row ***i*** in the numerator by INP(***i***, 2).

## ANN-Based Analytical Model

Once determined the preprocessed input dataset $\{Y_{1,sim}\}_n^{after}$ (a 2 x $P_{sim}$ matrix), the next step is to present it to the proposed NNC to obtain the predicted output dataset $\{Y_{5,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), which will be given in the same preprocessed format of the target dataset used in learning. In order to convert the predicted outputs to their 'original format' (i.e., without any transformation due to normalization or dimensional analysis), some postprocessing might be needed, as described in 4.4.3. Next, the mathematical representation of the proposed NNC is given, so that any user can implement it to determine $\{Y_{5,sim}\}_n^{after}$:

$$\left\{Y_{5,sim}\right\}_{n}^{after} = \left\{Y_{5,sim}\right\}_{n(0)}^{after} + \sum_{i=1}^{3} \left\{Y_{5,sim}\right\}_{n(i)}^{after} \tag{21}$$

being

$$\begin{aligned} Y_{2(i)} &= \varphi_2\left(W_{1-2(i)}^T\left\{Y_{1,sim}\right\}_n^{after} + b_{2(i)}\right) \\ Y_{3(i)} &= \varphi_3\left(W_{1-3(i)}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-3(i)}^T Y_{2(i)} + b_{3(i)}\right) \\ Y_{4(i)} &= \varphi_4\left(W_{1-4(i)}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-4(i)}^T Y_{2(i)} + W_{3-4(i)}^T Y_{3(i)} + b_{4(i)}\right) \\ \left\{Y_{5,sim}\right\}_{n(i)}^{after} &= \varphi_5\left(W_{1-5(i)}^T\left\{Y_{1,sim}\right\}_n^{after} + W_{2-5(i)}^T Y_{2(i)} + W_{3-5(i)}^T Y_{3(i)} + W_{4-5(i)}^T Y_{4(i)} + b_{5(i)}\right) \end{aligned} \tag{22}$$

where $i = 0,\ldots, 3$ and

$$\begin{aligned} \varphi_2 = \varphi_3 = \varphi_4 = \varphi(s) &= \frac{1}{1+e^{-s}} \\ \varphi_5 = \varphi_5(s) &= s \end{aligned} \tag{23}$$

Arrays $W_{j\text{-}s(i)}$ and $b_{s(i)}$ are stored online in [20].

## Output Data Postprocessing

In order to transform the output dataset obtained by the proposed NNC, $\{Y_{5,sim}\}_n^{after}$ (a 1 x $P_{sim}$ vector), to its original format ($Y_{5,sim}$), i.e. without the effects of dimensional analysis and/or output normalization (possibly) taken in target dataset preprocessing prior training, one has

$$Y_{5,sim} = \left\{Y_{5,sim}\right\}_n^{after} \tag{24}$$

since no output normalization nor dimensional analysis were adopted in the proposed model.

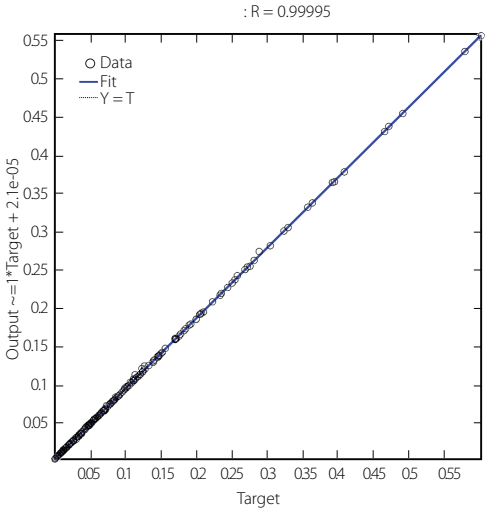


**FIGURE 14.** Regression plot for the proposed NNC.

### Performance Results

Finally, the results yielded by the proposed NNC for the 208-point final testing dataset (which includes the ANN learning/development counterpart), in terms of performance variables defined in sub-section 3.4, are presented in this sub-section in the form of two graphs: (i) a regression plot (Fig. 14), where network target and output data are plotted, for each data point, as ***x***- and ***y***- coordinates, respectively; and (ii) a plot (Fig. 15) indicating (for all data) the ($ii_1$) maximum error, ($ii_2$) percentage of errors larger than 3%, and ($ii_3$) average error (called performance).

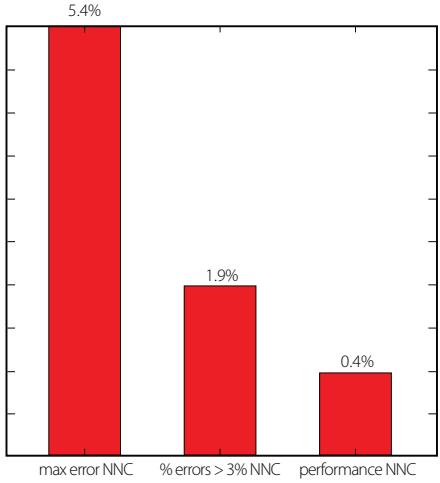


**FIGURE 15.** Maximum and average (performance) errors for the proposed NNC.

## CRITICAL VELOCITIES AND MAXIMUM DISPLACEMENTS PREDICTIONS

Eleven pairs of curves were obtained as output of the ANN-based models described in sub-sections 4.1-4.4. Each pair presents the maximum negative (downward) and positive (upward) displacement predictions as function of load velocity (from 50 to 300 m/s in intervals of 5 m/s) for different values of the maximal distributed friction force $f_u$, as depicted in Fig. 16 (two plots are presented for the sake of legibility). Note that the classic Winkler foundation case corresponds to the frictionless case ($f_u = 0$). Comparing the homologous curves in Fig.19 and Fig. 5(a) in [4], a very good visual agreement between the ANN-based predictions and the results of the mechanical model is observed. For a precise comparison, the obtained target (FE-based) and output (ANN-based) values can be found in [13].

The set of curves shows that the increase of the maximum frictional force per unit length ($f_u$) leads, as expected, to the reduction of the displacement peaks. The existence of a critical velocity, that is, a velocity that induces the beam's highest displacements, is also clear in Fig. 16. It is observed that, for small values of $f_u$, the value of the critical velocity is just slightly affected, whereas for larger frictional forces that value clearly rises.

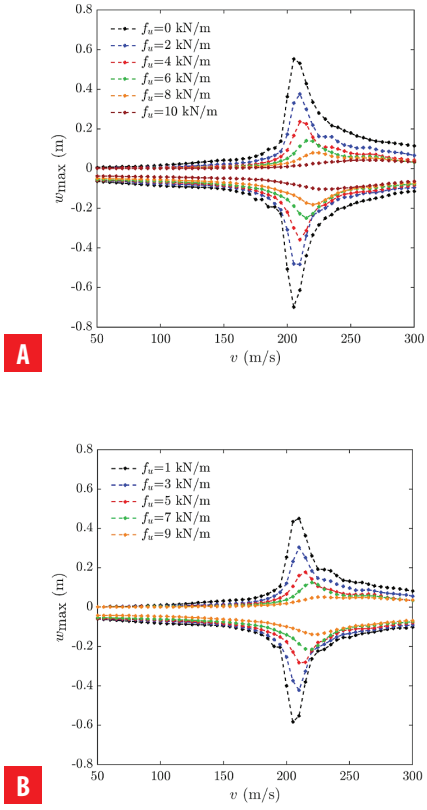


**FIGURE 16.** Maximum upward and downward ANN-based displacements for a frictionally damped beam as function of the load velocity: (a) $f_u = 0, 2, 4, 6, 8$ and 10 kN/m, (b) $f_u = 1, 3, 5, 7$ and 9 kN/m.

## DISCUSSION

In future publications it will be guaranteed that the validation and testing data subsets will be composed only by points where at least one variable (does not have to be the same for all) takes a value not taken in the training subset by that same variable. Based on very recent empirical conclusions by Abambres, the author believes it will lead to more robust ANN-based analytical models concerning their generalization ability (i.e. prediction accuracy for any data point within the variable ranges of the design data).

## FINAL REMARKS

This paper demonstrated the potential of Artificial Neural Networks (ANN) to effectively predict the maximum displacements and the critical velocities in railway beams under moving loads. Four ANN-based models were proposed, one per load velocity range ([50,

175] ∪ [250, 300] m/s; ]175, 250[ m/s) and per displacement type (upward or downward). Each model is function of two independent variables, a frictional parameter and the load velocity ($v$). The maximum and mean errors found for each aforementioned model (for all data), when comparing the ANN- and FE-based solutions, were: (i) {2.7, 0.4}% for downward displacements when $v$ = [50, 175] ∪ [250, 300] m/s, (ii) {5.2, 1.1}% for downward displacements when $v$ = ]175, 250[ m/s, (iii) {3.7, 0.2}% for upward displacements when $v$ = [50, 175] ∪ [250, 300] m/s, and (iv) {5.4, 0.4}% for upward displacements when $v$ = ]175, 250[ m/s. Furthermore, whereas the FE-based solution involves an average computing time per data point of thousands of seconds, the ANN counterpart does not even need a millisecond. More versatile ANN-based analytical models for the same type of problem may follow from this study by including more independent variables, such as the foundation stiffness modulus, the applied load magnitude, and the geometrical/mechanical properties of the railway beam.

## ACKNOWLEDGEMENTS

There are no conflicts of interest to disclose.

## AUTHOR CONTRIBUTIONS

Sections 1 and 2: Rita Corrêa, António Pinto da Costa, Fernando Simões; Sections 3, 4 and 5: Miguel Abambres; Section 6 and final review: All authors.

# ERRATUM

On March 2021, I, Miguel Abambres, found some bugs on my own ANN software, which affected the results presented in this paper. The software was debugged and the computational simulations rerun so that a new corrected version of the paper can be published during 2022. Details about the effects of the bugs on the published results will be provided.

**Software bugs in previous papers – sources and solutions**

On March 2021, Abambres found some bugs on his own ANN software, which had been used to yield ANN results for several papers published until that date, including this one. A not-so-severe bug was found in the definition of the 8th parametric sub-analysis, since it was not defined exactly as it was described in section "Parametric Analysis Results". The critical bugs were found in non-ELM (non-Extreme Learning Machine) learning algorithms for "mini-batch" and "online" training modes, and unintendedly caused the "% Train - Valid - Test" ANN feature to become "100 - 0 - 0" during neural net design, thus affecting the generalization potential of the proposed ANN. The bug sources were the following:

- "trainlm", "traingd" and "traingda" training functions do not allow incremental (online or mini-batch) training when using MATLAB neural network toolbox – regardless the data format employed in train(…) arguments, only batch training will be used.
- if net.divideMode is not specified when using MATLAB neural net toolbox, the default value for feedforward ANNs is 'sample'. However, the implemented data division (train / valid / test) for incremental training uses timestep (instead of sample) indexes, which means the assignment net.divideMode = 'time' was missing right before net.divideFcn = 'divideind'.

In order to overcome the above cited issues, Abambres first explored MATLAB's training functions that are said to allow incremental training, but found several limitations (reported in here https://archive.ph/r6Yw3 and here https://archive.ph/HsaC6). Thus, Abambres' final decision was to redefine the ANN parametric analysis in order to (i) remove all incremental non-ELM learning algorithms, and (ii) increase the ELM-based simulations while improving their cross-validation.

**Final Note**

For personal and professional reasons, I have decided not to write/publish (pro-bono) the new version of this paper for the results obtained with the debugged ANN software, even though the former were pretty satisfactory. Feel free to get in touch to amgg@mailfence.com if you need more information.

Miguel Abambres

## REFERENCES


[1] Frýba L (1972). Vibration of solids and structures under moving loads. Groningen: Noordhoff International Publishing.

[2] Madshus C, Kaynia A (2000). High-speed railway lines on soft ground: dynamic behaviour at critical train speed. Journal of Sound and Vibration, 231(3):689–701, doi: 10.1006/jsvi.1999.2647.

[3] Kaynia A, Madshus C, Zackrisson P (2000). Ground vibration from high-speed trains: pre- diction and countermeasure. Journal of Geotechnical and Geoenvironmental Engineering, 126(6):531–7.

[4] Toscano Corrêa R, Pinto da Costa A, Simões FMF (2018). Finite element modelling of a rail resting on a Winkler-Coulomb foundation and subjected to a moving concentrated load. International Journal of Mechanical Sciences, 140(May):432-45, doi: 10.1016/j.ijmecsci.2018.03.022.

[5] Dimitrovová Z, Rodrigues AFS (2012). Critical velocity of a uniformly moving load. Advances in Engineering Software, 50(August):44–56, doi: 10.1016/j.advengsoft.2012.02.011.

[6] Castro Jorge P, Pinto da Costa A, Simões FMF (2015a). Finite element dynamic analysis of finite beams on a bilinear foundation under a moving load. Journal of Sound and Vibration, 346(June):328–44, doi: 10.1016/j.jsv.2014.12.044.

[7] Castro Jorge P, Simões FMF, Pinto da Costa A (2015b). Dynamics of beams on non-uniform nonlinear foundations subjected to moving loads. Computers and Structures, 148(February):26–34, doi: 10.1016/j.compstruc.2014.11.002.

[8] The Mathworks, Inc (2017). MATLAB R2017a, User's Guide, Natick, USA.

[9] Glocker C (2001). Set-Values force laws. Lecture notes in applied and computational mechanics. Berlin, Heidelberg: Springer, ISBN 978-3-540-41436-0.

[10] Studer C (2009). Numerics of unilateral contacts and friction – Lecture notes in applied and computational mechanics. Berlin, Heidelberg: Springer, ISBN 978-3-642-01099-6.

[11] Moreau J (1994). Some numerical methods in multibody dynamics: application to granular materials. European Journal of Mechanics A/Solids, 13(4):94–114.

[12] Jean M (1999). The non-smooth contact dynamics method. Computer Methods in Applied Mechanics and Engineering, 177(3-4):235–57, doi: 10.1016/S0045-7825(98)00383-1.

[13] Authors (2018). ANN development + final testing datasets [Data set]. Zenodo. http://doi.org/10.5281/zenodo.1445912

[14] Abambres M, Corrêa R, Pinto da Costa A, Simões F (2019a). Potential of neural networks for maximum displacement predictions in railway beams on frictionally damped foundations. engrXiv (January), 1-62, doi: http://doi.org/10.31224/osf.io/m3b7j

[15] Abambres M, Marcy M, Doz G (2019b). Potential of Neural Networks for Structural Damage Localization. ACI – Avances en Ciencias e Ingenierías, 11(2), doi: http://dx.doi.org/10.18272/aci.v11i2.1305

[16] Researcher, The (2018). "Annsoftwarevalidation-report.pdf", figshare, doi:10.6084/m9.figshare.6962873.

[17] Developer (2018a). Negative wmax (v = [50, 175] ∪ [250, 300] m/s) [Data set]. Zenodo, http://doi.org/10.5281/zenodo.1462150

[18] Developer (2018b). Negative wmax (v = ]175, 250[ m/s) [Data set]. Zenodo, http://doi.org/10.5281/zenodo.1469207

[19] Developer (2018c). Positive wmax (v = [50, 175] ∪ [250, 300] m/s) [Data set]. Zenodo, http://doi.org/10.5281/zenodo.1469879

[20] Developer (2018d). Positive wmax (v = ]175, 250[ m/s) [Data set]. Zenodo, http://doi.org/10.5281/zenodo.1470854