Advances in Computational Design, *Vol. 7, No. 1 (2022) 1-17*
https://doi.org/10.12989/acd.2022.7.1.001

Numerical solution of beam equation using neural networks and evolutionary optimization tools

Mehdi Babaei[1a], Arman Atasoy*[2], Iman Hajirasouliha[3b], Somayeh Mollaei[1c] and Maysam Jalilkhani[4d]

[1]*Department of Civil Engineering, University of Bonab, Bonab, Iran*
[2]*Department of Civil Engineering, Istanbul Rumeli University, Istanbul, Turkey*
[3]*Department of Civil & Structural Engineering, The University of Sheffield, Sheffield, U.K.*
[4]*Department of Civil Engineering, Urmia University of Technology, Urmia, Iran*

(Received April 5, 2021, Revised August 11, 2021, Accepted September 3, 2021)

Abstract. In this study, a new strategy is presented to transmit the fundamental elastic beam problem into the modern optimization platform and solve it by using artificial intelligence (AI) tools. As a practical example, deflection of Euler-Bernoulli beam is mathematically formulated by 2nd-order ordinary differential equations (ODEs) in accordance to the classical beam theory. This fundamental engineer problem is then transmitted from classic formulation to its artificial-intelligence presentation where the behavior of the beam is simulated by using neural networks (NNs). The supervised training strategy is employed in the developed NNs implemented in the heuristic optimization algorithms as the fitness function. Different evolutionary optimization tools such as genetic algorithm (GA) and particle swarm optimization (PSO) are used to solve this non-linear optimization problem. The step-by-step procedure of the proposed method is presented in the form of a practical flowchart. The results indicate that the proposed method of using AI tools in solving beam ODEs can efficiently lead to accurate solutions with low computational costs, and should prove useful to solve more complex practical applications.

Keywords: artificial neural networks; elastic beam deflection; euler-bernoulli beam; genetic algorithm; ordinary differential equation; particle swarm optimization

1. Introduction

Most of physical systems in applied science and engineering are formulated in the form of a differential equation for which no analytical solution is imagined. In such cases, numerical methods can be used as effective tools to solve these differential equations. Use of artificial intelligence can provide a more computationally efficient alternative to tackle this problem. This method has been increasingly used in different fields of engineering disciplines in the past decade,

*Corresponding author, Assistant Professor, E-mail: arman.atasoy@rumeli.edu.tr
[a]Assistant Professor, E-mail: m.babaei@ubonab.ac.ir
[b]Associate Professor, E-mail: i.hajirasouliha@sheffield.ac.uk
[c]Assistant Professor, E-mail: s.mollaei@ubonab.ac.ir
[d]Assistant Professor, E-mail: m.jalilkhani@uut.ac.ir

 ISSN: 2383-8477 (Print), 2466-0523 (Online)

especially for the complicated cases where the conventional methods fail to provide sufficient performance (Jafarian *et al.* 2017, Sun *et al*. 2019). In most of these studies, the neural networks were utilized for machine learning and solving complex problems such as solving differential equations.

While there are several studies focused on solving differential equations addressing different engineering problems, there are only a few cases for which analytical solutions are achievable (e.g., Magill *et al*. 2018, Nabian and Meidani 2019). This implies that in most cases, ODEs are solved using numerical techniques. One of the widely used differential equations in common practice is the elastic beam equation obtained from Bernoulli beam theory. It is noted that unlike theoretical sciences, in the beam deflection problem, for most practical applications there is no need to have high precision in the final design solution. For example, even 10% of errors may be tolerable from the engineers' perspective. Also, in formulation of the beam equation, several approximations are used which reduce the accuracy of the solutions relative to the actual beam deflection. Non-distortion cross-section of bending plane and ignorance of shear deformation in Bernoulli beam are some of these assumptions. Considering this, several studies have been conducted in the past on how to model structural design problems with artificial intelligence tools (Babaei and Sheidaii 2013, Babaei and Sheidaii 2014, Fang *et al*. 2021). While the present study is in line with these studies, it addresses a structural analysis problem in a different field of engineering.

Different neural networks have been previously used to solve differential equations. For example, Magill *et al*. (2018) introduced a technique based on the canonical correlation analysis of the singular vector to measure the generality of neural network layers through a series of continuous parametric tasks. Several methods have been also adopted for solving differential equations, some of which can provide equation solutions in arrays that contain values of the answer function in the set of selected points in the domain. Others use basic functions for analytical solution and convert the problem into a set of simultaneous algebraic equations. It should be noted that most of the previous works in the field of solving differential equations are focused on the solution of algebraic equations, which results from the discretization of the solution domain. The solution of the linear equation system is also introduced to the Hopfield Neural Network (HNN) architecture. The analytical minimization of the network energy function yields the solution of the equation system (Lee and Kang 1990, Yentis and Zaghloul 1996). Lagaris *et al*. (1998) also developed a method for solving the ordinary differential equations and partial differential equations (PDEs), which relied on the ability to approximate the function of backpropagation networks and resulted in a closed-form derivable differential equation.

As one of the early studies, Dissanayake and Phan-Thien (1994) showed that neural networks can be efficiently used to solve differential equations. To this end, the neural network is trained to simulated the following function:

$$\mathrm{L} = \int_{\Omega} \|\mathrm{G[u](x)}\|^2 \mathrm{dV} + \int_{\partial\Omega} \|\mathrm{B[u](x)}\|^2 \mathrm{dS} \tag{1}$$

The output of this functional is then minimized to find the solution of the differential equation G[u]=0, which should be solved over the solution domain Ω. The second term of the right hand side contains the boundary conditions, i.e., B[u]=0. This term integrates the violation values of boundary conditions over Ω∂. The training input-data required for neural network is sampled from Ω. It should be mentioned that similar approaches have been initially proposed by Van. Milligan *et al*. (1995) and Lagaris *et al*. (1998), and further improved by Schmidhuber (Schmidhuber 2015)

and Yadav *et al*. (Yadav *et al*. 2015). Sirignano and Spiliopoulos (2018) and Berg and Nyström (2018) also showed that the points used for neural network training could be randomly sampled from the domain rather than using a meshing, which significantly improved the accuracy of the predicted results.

In general, there are two main reasons for studying neural networks (NNs) in differential equations. First, consistency of NNs can robustly simulate the unknown behavior of the studied systems. Second, the numerical algorithms, which are designed for solving differential equations with neural networks, can be efficiency used for predicting the system behavior. Nevertheless, if a neural network was trainable for any arbitrary type of input data with arbitrary size of information, it would be eventually possible to solve a wide range of equations. However, due to the limitations on the performance of neural networks and their high computational costs, the practical applications of these methods are generally limited (Magill *et al*. 2018). Previously, the first author presented a general method, in which the nonlinear differential equations were solved with the Fourier expansion tools and metaheuristic algorithms (Babaei 2013). Here, this idea is further developed by coupling the neural networks (NNs) with the new-porn meta-optimization algorithms.

In this study, a systematic method is presented for analyzing engineering problems using artificial intelligence tools such as neural networks (NNs), genetic algorithm (GA) and particle swarm optimization (PSO). The big challenge which is tackled here is the method of transmitting a fundamental engineer problem from classic form to its artificial-intelligence presentation. A simplified computational technique is proposed to provide training data for NNs, while several approaches are evaluated to obtain the best learning process. Supervised training strategy is employed to train the NN for simulating ordinary differential equations (ODEs) substitution. Integrated residual of the network is defined as the fitness function required to achieve the optimization target leading to the elastic curve of the beam. The method can be then extended in future studies to be applied as an effective method for solving a wide range of differential equations.

2. Problem statement

First, the selected problem statement is discussed in this part. Subsequently, the AI tools used in the proposed method are briefly introduced. Finally, it is explained how the novel AI-based procedure is generated for solving the beam equation, as a case study example.

2.1 Differential equation of beam deflection

The general form of an ordinary differential equation is as follows:

$$G\left[u^{(n)}, u^{(n-1)}, \cdots, u\right](x) = 0 \tag{2}$$

The boundary and initial conditions are also represented by: $B(x_i) = B_i,\ I(x_j) = I_j$. The deflection of a Bernoulli beam can be formulated as the following ODE under the geometrical boundary conditions:

$$\mathrm{y}(0) = \mathrm{y}_0, \mathrm{y}'(0) = \mathrm{y}'_0, \mathrm{y}(\mathrm{L}) = \mathrm{y}_\mathrm{L}, \mathrm{y}'(\mathrm{L}) = \mathrm{y}'_\mathrm{L} y'' - \frac{M(x)}{EI} = 0 \tag{3}$$

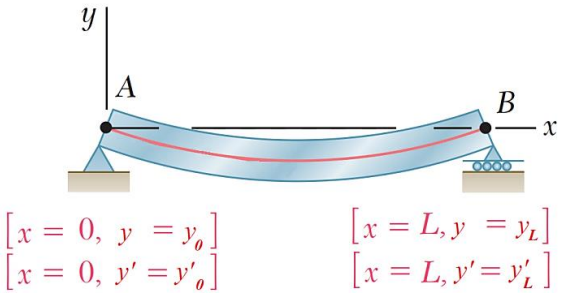

Fig. 1 Bernoulli beam and its boundary conditions

Fig. 1 shows the geometry of a Bernoulli beam and its boundary conditions.

The solution of the above ODE may be a continuous function or a discretely valued numerical function. The analytical form of solution can be typically expressed as:

$$y(x) \cong \hat{y}(x) \tag{4}$$

where $\hat{y}(x)$ is an analytical function which approximates the solution of the beam equation. Alternatively, the solution can be given in the form of the following sequence, which contains the solution values at selected points:

$$y(x) \cong \{y_i\} = [y_0 \ y_1 \ y_2 \ y_3 \ \dots \ y_k] \tag{5}$$

In the current study, the aim is to seek for the best sequence of deflection values. Subsequently, spline interpolation can be used to evaluate the deflection at other x-stations.

2.2 Using neural networks for solving ODEs

Artificial neural networks are one of the main tools of artificial intelligence for machine learning. They artificially model biological neural networks, while their learning potential is originated from the connectivity of their components, so-called artificial neurons. Artificial neurons represent simplified models of natural neurons, which consist of dendrites, axons, nucleons or nuclei, and axon terminals. The biological neural networks can regulate their inputs and outputs by modifying the synaptic connections. Similarly, the artificial neural networks are generally trained by adjusting the weight and bias factors. Typical components of an artificial neuron are shown in Fig. 2.

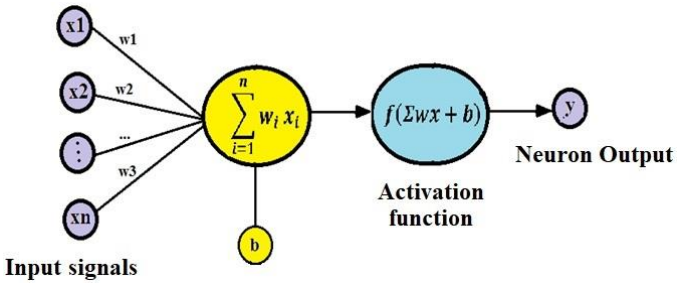

Fig. 2 Components of an artificial neuron in the neural network (Omondi and Rajapakse 2006)

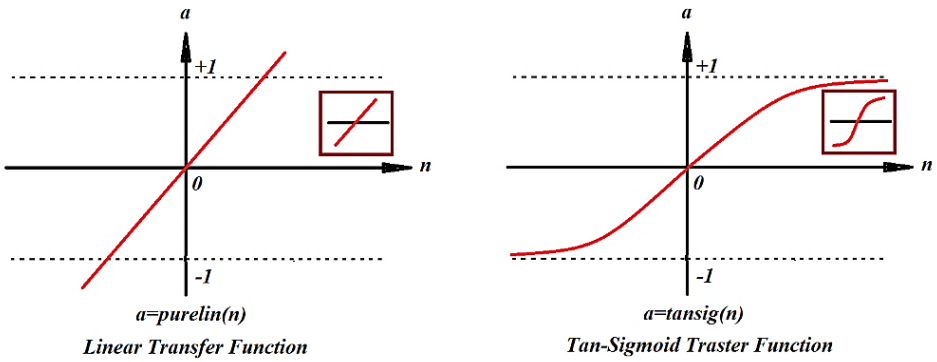

Fig. 3 Neural network transfer functions which are used in this work

The x_i inputs to the artificial neurons are weighted by the w_i factors, summed up with the bias b, and introduced into the activation or transfer function. This function is usually *a* Step, Linear, or Sigmoid function. The most commonly used transfer functions, which are also implemented in this study, are presented in Fig. 3.

Artificial neural networks can predict the solution of the similar problems through their prior experiences. This indicates that the training examples should contain accurate and purposeful information of the system, incorporating the shape, pattern and overall similar reactions to the expected solutions. Otherwise, the accuracy of the predictions may significantly reduce. Besides, if the number of samples is too high or the data pattern is hard-to-recognized, the computational cost significantly increases, which can make the method impractical.

The neural networks vary in terms of neuron structure, the number of layers, their connections (network architecture), and training algorithm. The perceptron, backpropagation, and radial networks are some of the most widely used networks. The number of layers in a network plays key role in the efficiency and performance of the system. The last layer is called the output layer, while the other layers are considered as hidden layers. The number of hidden layers can be one or more.

In the present study, to conduct training process of the NN, a backpropagation network with one output layer containing a linear transfer function and two hidden layers of sigmoidal and linear transfer functions are used. The architecture of this network is shown in Fig. 4. According to the insight obtained from vast number of trial-and-error, this network has been identified as one of the most powerful neural networks in predicting the behavior of beam differential equation.

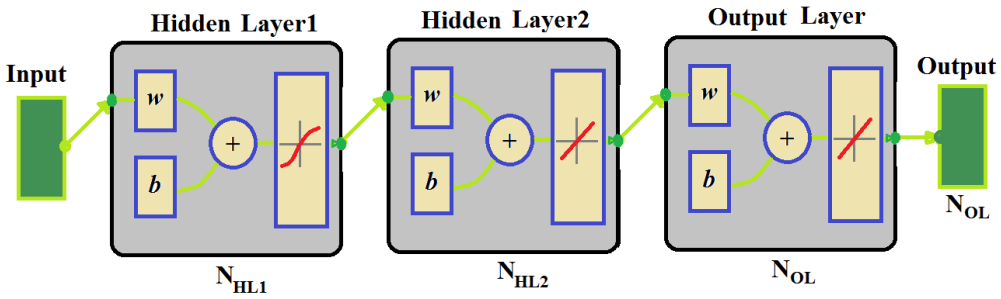

Fig. 4 Neural network architecture used in this study

2.3 Particle Swarm Optimization algorithm

Particle Swarm Optimization (PSO), also known as birds flocking algorithm, is inspired from the flock of the birds or fish colony. It has been widely used for dealing with large-scale nonlinear optimization problems in the field of continuous and discrete optimization (Ye *et al*. 2016, 2018). The population of PSO is called flock, while each individual in the population is considered as a particle (Luh *et al*. 2011). PSO performs the search operation by adjusting the position and path of the particles in a feasible solution space based on information received from colony particles, including the best up-to-now experiences of each particle and its neighbors.

Eberhart and Kennedy first introduced the original algorithm for PSO (Eberhart and Kennedy 1995). In summary, each flock is composed of N_p moving particles in the solution space, each representing a potential solution. The particles are identified by their position and velocity and fly in the n-dimensional space to find the optimal solution using the best position of their neighbors as well as their previous best position. Each solution is interpreted as the coordinates of the particles in the search space. The key positions include the best position of the particle (Pbest) and the best position of the surrounding particles (*Gbest*) (Koza 1992). Using these two positions, PSO relations can be written as follows:

$$V_i^d(k+1) = \omega V_i^d(k) + c_1 \times rand1_i^d \times \left(Pbest_i^d(k) - X_i^d(k)\right) + c_2 \times rand2_i^d \times \left(Gbest_i^d(k) - X_i^d(k)\right) \quad (6)$$

$$X_i^d(k+1) = X_i^d(k) + V_i^d(k+1) \quad (7)$$

where $v_i^d(k)$ denotes the current velocity of the design variable, the superscript d is the particle number, the index i represents the i^{th} design variable, and k is the iteration number. $rand1_i^d$ and $rand2_i^d$ are two random numbers between 0 and 1. $Pbest_i^d(k)$ is the best position, seen by the particle d, and $Gbest^d(k)$ is the best position explored by all the particles. ω, c_1 and c_2 are inertia weight, individual learning factor, and social learning factor, respectively. These components PSO algorithm is graphically presented in Fig. 5.

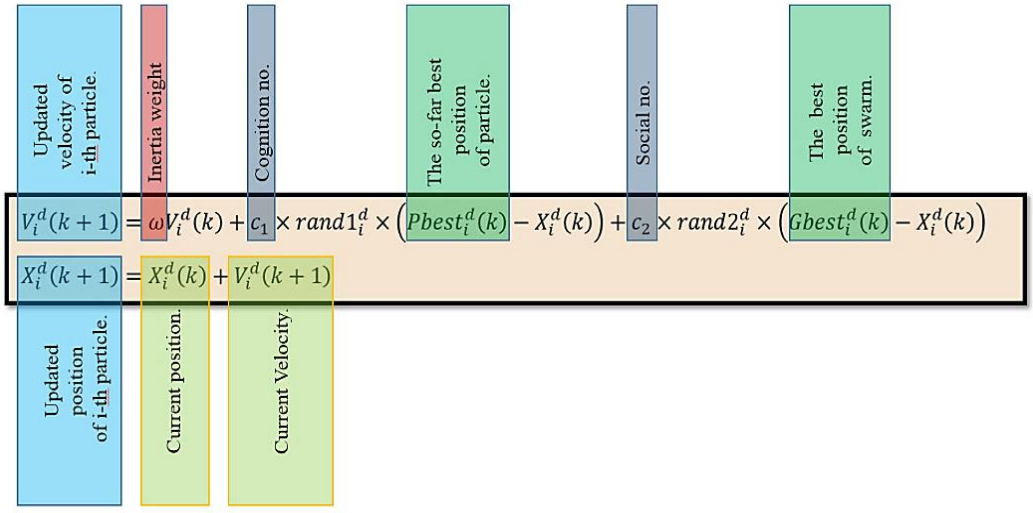

Fig. 5 Components and parameters of PSO (Particle Swarm Optimization) algorithm

2.4 Genetic algorithms

GA is an evolutionary optimization algorithm which was initially developed by John Holland (Holland 1992). GA simulates natural genetic rule in a digital space to evolve the quality of individuals for minimization goal. This method has been widely used for optimization of non-linear structural elements and systems (e.g., Phan *et al*. 2020, Parastesh *et al*. 2021, Mojtabaei *et al*. 2021). Similar to PSO, GA starts the searching operation by a randomly generated set of individuals known as chromosome. Each chromosome experiences a set of reproduction operators to increase their fitness. Finally, the most fitted one is identified as the best solution. In general, GA conducts the following steps to solve an optimization problem:

• Evaluation: The fitness function, $F(x)$, is computed for each chromosome X.

• Selection: Two or more individuals are selected to pass into the mating pool.

• Reproduction: Offspring (or so-called children) are produced by the selected individuals coming from selection step, namely parents. Two basic operations are applied to the parents to generate offspring: crossover and mutation (Sivanandam and Deepa 2008). To generate new individuals, a part of the generation is separated for mutation operation. This basically prevents the algorithm trapping in local minima, and also plays a dominant role in recovering the lost genetic materials. Fig. 6 schematically shows a simple example of crossover and mutation on two artificial chromosomes.

• Replacement: A few individuals from the old population are left aside and replaced by new ones.

• Stopping criteria: GA checks a series of stopping conditions and the algorithm halts if one or more of them are satisfied.

Repetition of the aforementioned procedure often converges to global minimum. Or, one of the stopping criteria ends the operation and gives the best founded solution.

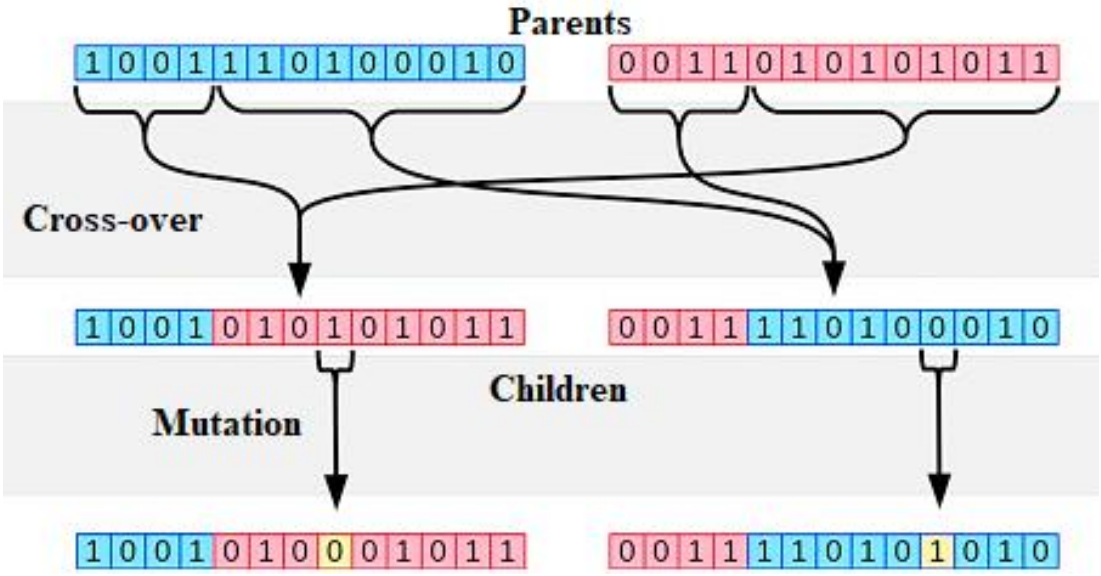

Fig. 6 Genetic algorithm operators on binary individuals: cross-over, mutation (adopted from www.sciencedirect.com/topics/medicine-and-dentistry/genetic-operator)

2.5 Proposed Problem-Solving Method

As discussed before, this work attempted to solve the beam-deflection problem using artificial intelligence tools. Accordingly, the neural network is coupled with PSO and GA to present a new and efficient problem solving procedure. Utilizing neural networks requires primary data, so-called training data for network training. To produce this data, some distinctive smoothness and continuity features of beam deformation are used to refine the training data and make it more targeted. Then, the smooth function of spline interpolation is used for the interpolation of the deflections at discrete points. It should be noted that, under the applied loading, the beam deflection is a one-way curvature curve, and therefore, the generated data for beam deflections are ranging from 0 to a small positive value, e.g., 0.05 m. In this way, the engineering insight is coupled with the processing capacity of AI tools to conduct a knowledge-based system. The implementation steps of the new method can be summered as follows:

• 1st step: Introduce the required problem information as follows:

- Geometric configuration of the beam as well as the length of the beam L, and its moment of inertia $I = I(x)$
- Boundary and initial conditions at the ends: $y(0) = y_0$, $y'(0) = y'_0$, $y(L) = y_L$, $y'(L) = y'_L$.
- Mechanical properties of material such as modulus of elasticity E.
- Loading function $Q(x)$ and internal moment function $M(x)$ versus to the position component x.

• 2nd step: The beam is divided over its length. It should be noted that creating a large number of sub-domains at this stage does not necessarily lead to higher precision since the variables are adjusted to work independent of the mesh properties. In this case, 3 or 5 points are often sufficient. To apply the boundary conditions, the mesh points should include the boundary points:

$$\text{Mesh points: } \{x_i\} = \{(x_0 = 0) < x_1 < x_2 < \cdots < (x_{Nx} = L)\} \tag{8}$$

where Nx is the number of points over the x-axis. It is noted that using a large number of points is also undesired since it requires more output neurons in the developed neural networks. In this work, Nx is set to 10~30 according to the available computational equipment.

• 3rd step: A limited number of mesh points (e.g., three, five, seven) are sampled through the beam length. These points are used for training the neural network:

$$\text{Sampled points: } \{x_{s,j}\} = \{x_{s,1}, x_{s,2}, \dots, x_{s,Ns}\} \tag{9}$$

where Ns is the number of sampled points.

The positions of these points are arbitrary but they should be appropriately distributed through the beam to identify the overall layout of the beam deflection.

• 4th step: Hundreds of deflection vectors are generated for each sampled x-vector:

$$\{y_{s,j}\} = \{y_{s,1}, y_{s,2}, \dots, y_{s,Ns}\} \tag{10}$$

In this study, 800 set of input vectors are generated for the three sampling case in the selected examples.

• 5th step: Using $\{x_{s,j}\}$ and $\{y_{s,j}\}$ and boundary conditions $y(0) = y_0$, $y(L) = y_L$, we constitute the following vectors:

$$\{0,\ x_{s,j},\ L\} \leftrightarrow \{y_0\ ,\ y_{s,j}\ ,\ y_L\} \tag{11}$$

Subsequently, the deflection values are interpolated at the mesh points:

$$\{y_i\} = \{y_0, y_1, y_2, \dots, y_{Nx} = y_L\} \tag{12}$$

This process is applied to each $\{y_{s,j}\}$ data set. In general, for such cases using spline interpolation is advised rather than other interpolation methods. It is noted that the given displacement conditions, $y(0) = y_0$, $y(L) = y_L$, are manually inserted into the input vectors. This is the simplest way of dealing with the geometrical boundary conditions in the presented method.

• 6th step: The first- and second-order derivatives of the deflection curve are calculated by using the following finite difference relations:

$$y_i' = \frac{y_{i+1} - y_i}{\Delta x}, \; y_i' = \frac{y_{i+2} - y_i}{2\Delta x}, \; y_i' = \frac{-3y_i + 4y_{i+1} - y_{i+2}}{2\Delta x} \tag{13}$$

$$y_i'' = \frac{y_{i+1}' - y_i'}{\Delta x}, \; y_i'' = \frac{y_{i+2}' - y_i'}{2\Delta x}, \; y_i'' = \frac{y_i - 2y_{i+1} + y_{i+2}}{\Delta x^2} \tag{14}$$

Then, the following vectors can be obtained for each set of $\{y_i\}$:

$$\{y_i'\} = \left\{y_0', y_1', y_2', \dots, y_{Nx}' = y_L'\right\} \tag{15}$$

$$\left\{y_i''\right\} = \left\{y_0'', y_1'', y_2'', \dots, y_{Nx}'' = y_L''\right\} \tag{16}$$

The slope boundary conditions, if available, are applied to y'_i-vector by substituting their given values for y'_0 and y'_L in the vector.

• 7th step: Target data sets are prepared for training the neural network. First, the internal moment function at mesh points $\{x_i\}$ is evaluated to obtain discrete value of $\{M_i\}$.

$$\{M_i\} = \{M_0, M_1, M_2, \dots, M_{Nx}\} \tag{17}$$

Subsequently, the residual of the differential equation at mesh points is calculated through the following function:

$$\{Res_i\} = \{EIy_i'' - M_i\} \tag{18}$$

This equation produces *Ns* target data set for training neural network. Each set has *Nx* elements inside.

• 8th step: Extensive number trial-and-error conducted in this study shows that feed-forward backpropagation network with $88 - 44 - Nx$ architecture provides a suitable network for this problem (i.e., it can simulate the behavior of the differential equation using minimum number of neurons). The input data of the network is presented by $\{y_{s,j}\}$ vectors, while the target data is shown by $\{Res_i\}$ vectors corresponding to the input data.

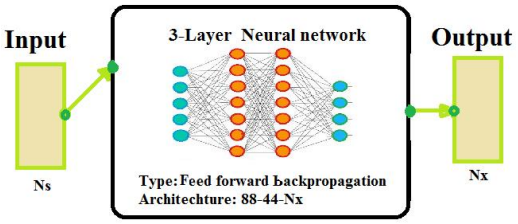

Fig. 7 Neural network used in this work

• 9th step: The trained neural network is introduced to PSO or GA to be used in the optimization process. Regardless of the selected optimization algorithm, a well-defined lose function (LF) is required. Sum of the weighted residual values is an appropriate choice:

$$LF(\{y_i\}) = \sum_{i=1}^{Nx} |w_i Res_i| \tag{19}$$

where w_i is weighting factor that it may take different at the mesh points. It is noted that additional constraints such as force boundary conditions, could be included in the loos function via penalty function strategy or any other constraint-handling approaches.

Fig. 8 presents the flowchart of the proposed method.

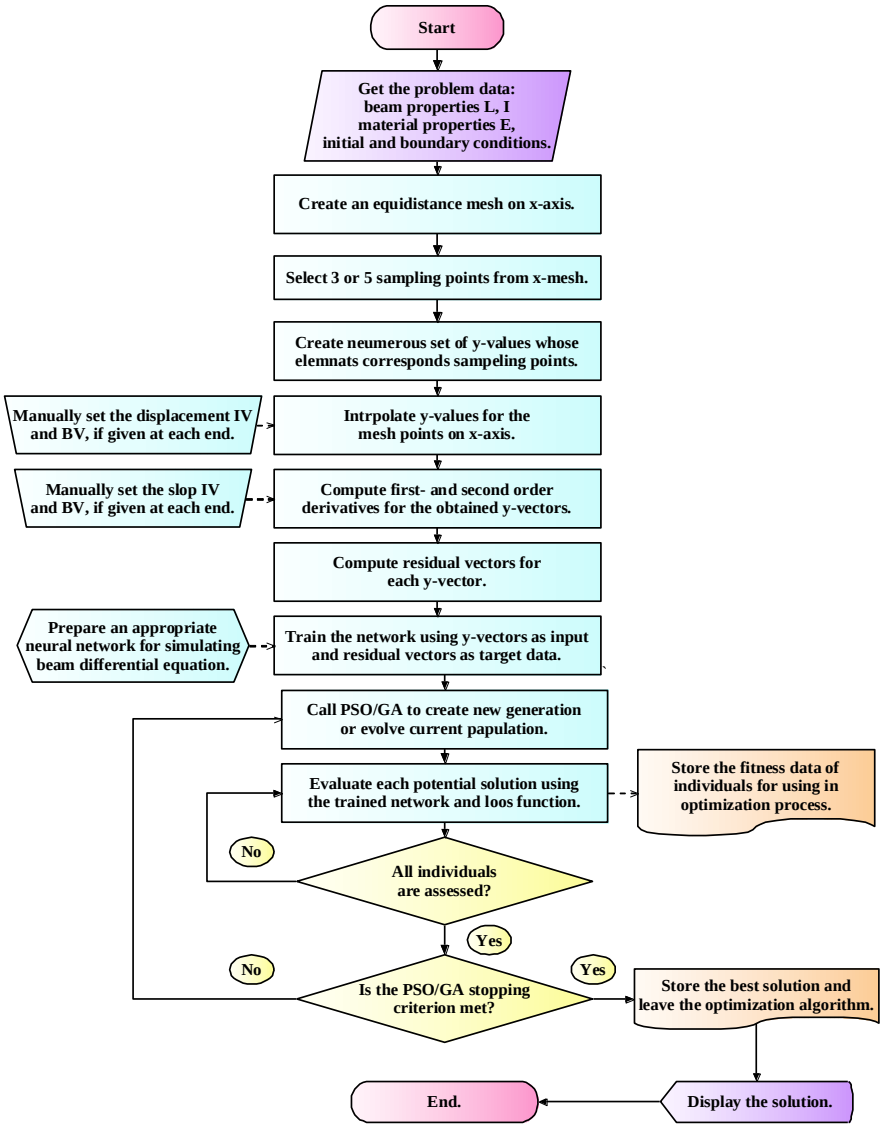

Fig. 8 Flowchart of the presented method for solving beam equation by NN and PSO/GA

3. Results and discussions

To demonstrate the efficiency of the proposed method, two practical examples are investigated in this section. The first example shows the generality of the proposed method, while the second example is a more complicated problem which is used to investigate the efficiency of the proposed idea for practical applications. The key parameters of the adopted NN, PSO, and GA tools used to solve these examples are listed in Table 1 for simple regeneration of the results.

3.1 Example I

A well-known benchmark beam analysis is used as the first example. As shown in Fig. 9, a 6 m long steel joint beam is considered under uniformly distributed load q = 10 kN/m ≈ 1 tonf/m. The modulus of elasticity of steel is $E = 2 \times 10^5$ MPa and the moment of inertia of the beam around the bending axis is I = 5505 cm^4. The internal moment of this beam is obtained from static equilibrium as:

$$M = \frac{qL}{2}x - \frac{qx^2}{2} \tag{20}$$

The differential equation of the beam deflection is presented as follows:

$$y'' = -\frac{1}{EI} \times (\frac{qL}{2}x - \frac{qx^2}{2}) \tag{21}$$

Table 1 Basic configuration parameters of the implemented tools in examples I and II

AI tool	Type and configuration parameters
Neural network	• Feed forward backpropagation NN • 3-layer network • Architecture: [88,44, Nx] • Transfer functions: {'tansig','purelin','purelin'} • Epochs #: 120 • Number of training data set: 800~1200 • Vector size of input data: 3~5 • Vector size of target data: 15~30
Particle swarm optimization	• Basic version of PSO. • Papulation size: 160 • Social factor: 1.5 • Cognition factor: 0.5 • Generation: 80 • Lower bounds: -0.003 meter • Upper bounds: 0.000
Genetic algorithm	• Basic version of GA available in MATLAB Software. • Papulation size: 200 • Crossover rate: 0.8 • Mutation rate: 0.05 • Generation:120 • Lower bounds: -0.005 meter • Upper bounds: 0.000

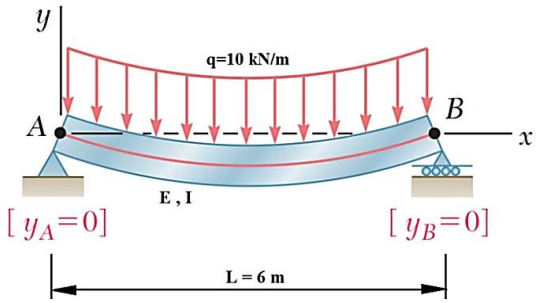

Fig. 9 Simple beam geometry under uniformly distributed load

Table 2 Numerical comparison of the results for example I

x (m)	Simple beam deflections y (m)				
Section	Analytical Solutions	3-Point Sampling (PSO)	3-Point Sampling (GA)	5-Point Sampling (PSO)	5-Point Sampling (GA)
0	0	0	0	0	0
0.3	-0.00244	-0.00239	-0.00251	-0.0021	-0.00251
0.6	-0.00481	-0.0047	-0.00494	-0.00423	-0.00494
0.9	-0.00705	-0.00691	-0.00725	-0.00635	-0.00725
1.2	-0.0091	-0.00896	-0.00938	-0.0084	-0.00938
1.5	-0.01092	-0.01081	-0.01128	-0.0103	-0.01128
1.8	-0.01246	-0.01243	-0.0129	-0.012	-0.0129
2.1	-0.0137	-0.01377	-0.0142	-0.01345	-0.0142
2.4	-0.0146	-0.01479	-0.01513	-0.01457	-0.01513
2.7	-0.01514	-0.01545	-0.01569	-0.0153	-0.01569
3	-0.01533	-0.01571	-0.01588	-0.0156	-0.01588
3.3	-0.01514	-0.01553	-0.01569	-0.01541	-0.01569
3.6	-0.0146	-0.01495	-0.01513	-0.01477	-0.01513
3.9	-0.0137	-0.01399	-0.01421	-0.01374	-0.01421
4.2	-0.01246	-0.01271	-0.01292	-0.01236	-0.01292
4.5	-0.01092	-0.01112	-0.0113	-0.0107	-0.0113
4.8	-0.0091	-0.00927	-0.0094	-0.00881	-0.0094
5.1	-0.00705	-0.0072	-0.00727	-0.00674	-0.00727
5.4	-0.00481	-0.00494	-0.00496	-0.00454	-0.00496
5.7	-0.00244	-0.00253	-0.00252	-0.00228	-0.00252
6	0	0	0	0	0

From the theory of beams, the analytical solution of this equation is in the form of:

$$y = -\frac{qx}{24EI}(L^3 - 2Lx^2 + x^3) \tag{22}$$

To compare the exact solution and that one obtained from the proposed method, the beam deflection is calculated by 3- and 5-point sampling separately as presented in Table 2.

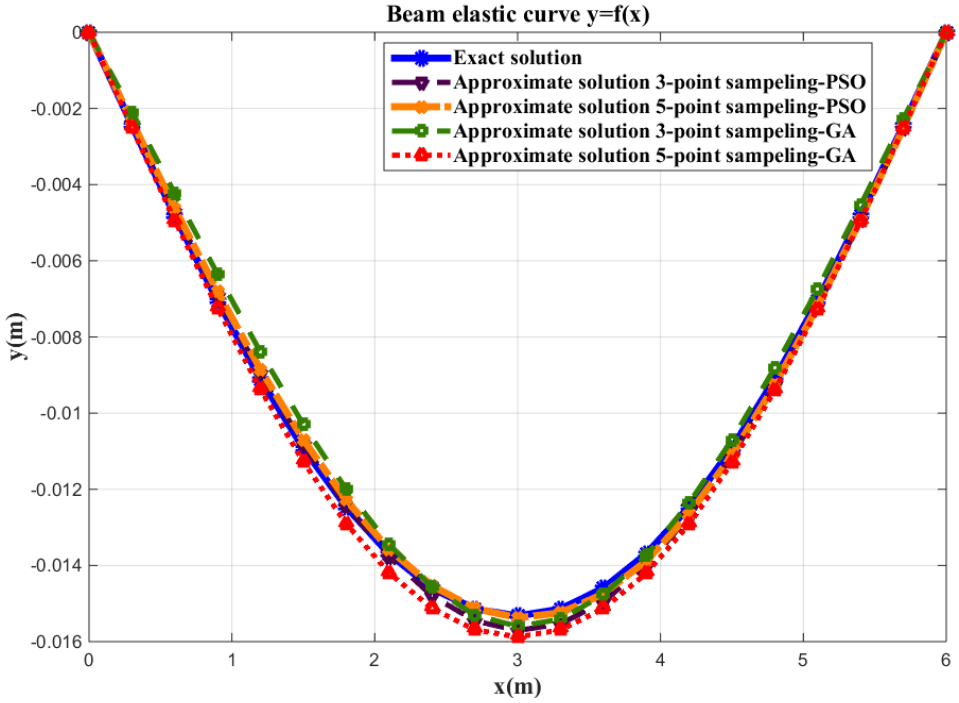

Fig. 10 Elastic beam deflection curve: comparison of the results for example I

The deflection curve corresponding to the values presented in Table 2 is plotted for 3- and 5-point sampling in Fig. 10.

3.2 Example II

The second example is an indeterminate beam with fixed supports and two statically unknown reactions. In structural analysis, these reactions are determined by using matrix analysis or finite element simulation. The elastic-curve can be then determined to check the code-design requirements. As shown in Fig. 11, the selected beam has a span length of 8 m. It is under combined uniformly distributed load of $q = 4$ *kN*/*m* (≈ 0.4 ton*f*/m) and concentrated load of $P = 16$ *kN* (≈ 1.6 ton*f*) at a distance $a = 6$ *m* from the left support. The flexural rigidity of the beam is EI = 110×10^6 $N.m^2$. The expression for bending moment is:

$$M = \begin{cases} \frac{q}{12}(6Lx - L^2 - 6x^2) + \frac{Pb^2}{L^3}(3a + b)x - \frac{Pab^2}{L^2} & for\ x < a \\ \frac{q}{12}(6Lx - L^2 - 6x^2) + \frac{Pa^2}{L^3}(a + 3b)(L - x) - \frac{Pa^2b}{L^2} & for\ x > a \end{cases} \tag{23}$$

Substituting the Eq. (23) into Eq. (3), the following elastic curve equation is obtained:

$$y = \begin{cases} \frac{q\,x^2}{24\,EI}(L - x)^2 + \frac{2Pb^2x^2}{12EIL^3}(3aL - 3ax - bx) & for\ x < a \\ \frac{q\,x^2}{24\,EI}(L - x)^2 + \frac{2Pa^2(L - x)^2}{12EIL^3}\left(3bL - 3b(L - x) - a(L - x)\right) & for\ x > a \end{cases} \tag{24}$$

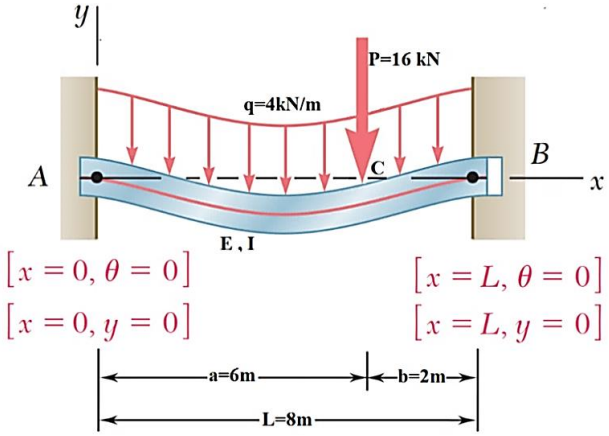

Fig. 11 Fixed-end beam geometry under uniformly distributed load plus concentrated load

Table 3 Numerical comparison of the results in example II

x (m)	Simple beam deflections y (m)				
Section	Analytical Solutions	3-Point Sampling (PSO)	3-Point Sampling (GA)	5-Point Sampling (PSO)	5-Point Sampling (GA)
0	0	0	0	0	0
0.5	-0.00028	-0.00027	-0.00028	-0.00028	-0.00028
1	-0.00098	-0.00093	-0.00097	-0.00095	-0.00098
1.5	-0.00193	-0.00184	-0.00191	-0.00186	-0.00195
2	-0.00297	-0.00288	-0.00297	-0.00291	-0.00304
2.5	-0.00398	-0.00392	-0.00401	-0.00395	-0.00409
3	-0.00484	-0.00482	-0.00487	-0.00486	-0.00497
3.5	-0.00548	-0.00547	-0.00549	-0.0055	-0.00559
4	-0.00582	-0.00573	-0.00584	-0.00576	-0.00592
4.5	-0.00583	-0.00552	-0.00588	-0.00555	-0.00592
5	-0.00549	-0.00492	-0.00556	-0.00493	-0.00557
5.5	-0.00481	-0.00405	-0.00483	-0.00405	-0.00482
6	-0.00382	-0.00303	-0.00373	-0.00303	-0.00374
6.5	-0.00259	-0.00199	-0.00248	-0.00199	-0.00252
7	-0.00136	-0.00106	-0.00129	-0.00105	-0.00136
7.5	-0.00039	-0.00035	-0.00039	-0.00035	-0.00045
8	0	0	0	0	0

To have a quantitative comparison, this equation is solved by 3- and 5-point sampling approach and their results are summarized in Table 3. In addition, a graphical plot is provided in Fig. 12 for visual comparison of the various methods.

The results indicate that using 3-point approximation do not provide satisfactory results for this unsymmetrical case. This deficiency is resolved by sampling more points from x-interval. It can be seen from Fig. 12 that the 5-point approximation leads to more accurate prediction of the

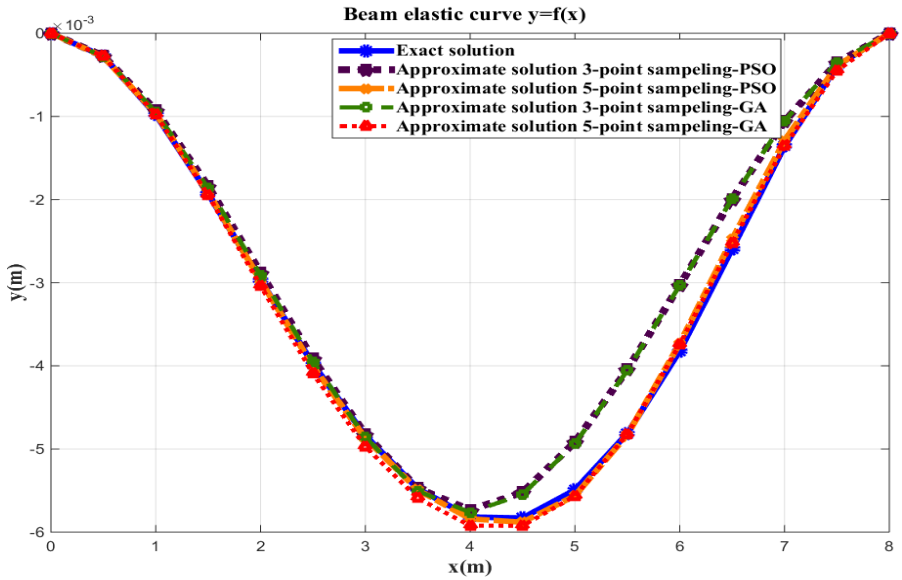

Fig. 12 Elastic beam deflection curve: comparison of results in example II

results. However, a higher level of precision could be also achieved by using more advanced neural networks and optimization algorithms. However, this would considerably increase the computational costs.

3.3 Discussions

The results presented in the previous sections indicate that the numerical deflections obtained from the proposed method have less than %10 of errors compared to the accurate analytical solutions. Figs. 10 and 12 show that the method could properly recognize the overall layout of the beam deflections under various loads with very low computational cost. One of the main challenges addressed in this study, was the convergence of neural network in the training process. The large amount of data required for achieving a good performance, made the network training hard to converge. Considering the available computational facilities, the number of sampling points was selected to be 3 and 5 while the number of layered neurons was set to 88 and 44, respectively. However, the use of high-processing computer systems would make it possible to add the number of sampling points or try other ideas such as using derivatives in the network input. While using more sampling points (can provide more accurate predictions, it significantly increases the computational cost of the network training. For example, even increasing the number of sampling points to 7 and 9 requires much higher number of neurons in the hidden layers that leads to a significant increase in the required network training time. On the other hand, while accurate results were obtained by using 800~1200 training data sets, increasing the number of learning data sets, e.g., to 2400~3200, could reduce the network simulation error. However, this would again require using higher spec CPU and memory capacity. The architecture and the type of neural networks can also highly affect the performance of the proposed method. While in this study a three-layer backpropagation neural network was successfully employed, this can be a good area for future investigations.

4. Conclusions

In this research, a systematic method was presented for solving Bernoulli beam equation using artificial intelligence (AI) tools. Neural networks (NNs) were coupled with evolutionary optimizers, genetic algorithm (GA) and particle swarm optimization (PSO), to develop an efficient AI-based method to solve ordinary differential equations (ODEs). The big challenge was the method of transmitting the classic beam problem into the artificial-intelligence form. Training efficient neural networks (NNs) for simulating differential equations was another problematic issue, which was resolved by using simplifying assumptions. Several trial-and-errors were conducted to find the most appropriate neural network for simulating the behaviour of 2^{nd}-order ODEs using minimum number of neurons. One of the most important advantages of this approach is the fact that if the error function values do not properly match the network output for the untrained input data, they can be easily detected to avoid large errors in prediction of the NN. From the applied engineering view-point, the results obtained from the proposed method are acceptable, and the solution confirms efficiency of the new method.

References

Babaei, M. (2013), "A general approach to approximate solutions of nonlinear differential equations using particle swarm optimization", *Appl. Soft Comput*., **13**(7), 3354-3365. https://doi.org/10.1016/j.asoc.2013.02.005.

Babaei, M. and Sheidaii, M.R. (2013), "Optimal design of double layer scallop domes using genetic algorithm", *Appl. Mathem. Modelling*, **37**(4), 2127-2138. https://doi.org/10.1016/j.apm.2012.04.053.

Babaei, M. and Sheidaii, M.R. (2014), "Automated optimal design of double-layer latticed domes using particle swarm optimization", *Struct. Multidiscip. O.*, **50**(2), 221-235. https://doi.org/10.1007/s00158-013-1042-2.

Berg, J. and Nyström, K. (2018), "A unified deep artificial neural network approach to partial differential equations in complex geometries", *Neurocomput.*, **317**, 28-41. https://doi.org/10.1016/j.neucom.2018.06.056.

Dissanayake, M. and Phan-Thien, N. (1994), "Neural-network-based approximations for solving partial differential equations", *Commun. Numer. Meth. Eng.*, **10**(3), 195-201. https://doi.org/10.1002/cnm.1640100303.

Eberhart, R. and Kennedy, J. (1995), "Particle swarm optimization", *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, Australia, November.

Fang, Z., Roy, K., Chen, B., Sham, C.W., Hajirasouliha, I. and Lim, J.B. (2021), "Deep learning-based procedure for structural design of cold-formed steel channel sections with edge-stiffened and un-stiffened holes under axial compression", *Thin Wall. Struct*., **166**, 108076. https://doi.org/10.1016/j.tws.2021.108076.

Holland, J.H. (1992), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT press, U.S.

Jafarian, A., Mokhtarpour, M. and Baleanu, D. (2017). "Artificial neural network approach for a class of fractional ordinary differential equation", *Neural Comput. Appl.*, **28**(4), 765-773. https://doi.org/10.1007/s00521-015-2104-8.

Koza, J.R. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT press, Cambridge, Massachusetts, London, England.

Lagaris, I. E., Likas, A. and Fotiadis, D. (1998), "Artificial neural networks for solving ordinary and partial differential equations", *IEEE T. Neural. Networ.*, **9**(5), 987-1000. https://doi.org/10.1109/72.712178.

Lee, H. and Kang I.S. (1990), "Neural algorithm for solving differential equations", *J. Comput. Phys.*, **91**(1),

110-131. https://doi.org/10.1016/0021-9991(90)90007-N.
Luh, G.C., Lin, C.Y. and Lin, Y.S. (2011), "A binary particle swarm optimization for continuum structural topology optimization", *Appl. Soft Comput.*, **11**(2), 2833-2844. https://doi.org/10.1016/j.asoc.2010.11.013.
Magill, M., Qureshi, F., and de Haan, H.W. (2018), "Neural networks trained to solve differential equations learn general representations", *Adv. Neural Inform. Process. Syst.*, 1097-1105. https://arxiv.org/abs/1807.00042.
Mojtabaei, S.M., Hajirasouliha, I., Ye, J. (2021), "Optimization of cold-formed steel beams for best seismic performance in bolted moment connections", *J. Constr. Steel Res.*, **181**, 106621. https://doi.org/10.1016/j.jcsr.2021.106621.
Nabian, M.A. and Meidani, H. (2019). "A deep learning solution approach for high-dimensional random differential equations", *Probabilist. Eng. Mech.*, **57**, 14-25. https://doi.org/10.1016/j.probengmech.2019.05.001.
Omondi, A.R. and Rajapakse J.C. (2006), *FPGA Implementations of Neural Networks*, Springer, Boston, MA. https://doi.org/10.1007/0-387-28487-7.
Parastesh, H., Mojtabaei, S.M., Taji, H., Hajirasouliha, I. and Sabbagh, A.B. (2021), "Constrained optimization of anti-symmetric cold-formed steel beam-column sections", *Eng. Struct.*, **228**, 111452. https://doi.org/10.1016/j.engstruct.2020.111452.
Phan, D.T., Mojtabaei, S.M., Hajirasouliha, I., Ye, J. and Lim, J.B.P (2020), "Coupled element and structural level optimisation framework for cold-formed steel frames", *J. Constr. Steel Res.*, **168**, 105867. https://doi.org/10.1016/j.jcsr.2019.105867.
Schmidhuber, J. (2015), "Deep learning in neural networks: An overview", *Neural Networks*, **61**, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003.
Sirignano, J. and Spiliopoulos, K. (2018), "DGM: A deep learning algorithm for solving partial differential equations", *J. Comput. Phys.*, **375**, 1339-1364. https://doi.org/10.1016/j.jcp.2018.08.029.
Sivanandam, S.N. and Deepa, S.N. (2008), *Introduction to genetic algorithms*, Springer, Berlin, Heidelberg, Germany.
Sun, H., Hou, M., Yang, Y., Zhang, T., Weng, F. and Han, F. (2019). "Solving partial differential equation based on Bernstein neural network and extreme learning machine algorithm", *Neural Process. Lett.*, **50**(2), 1153-1172. https://doi.org/10.1007/s11063-018-9911-8.
Van Milligen, B.P., Tribaldos, V. and Jiménez, J.A. (1995), "Neural network differential equation and plasma equilibrium solver", *Phys. Rev. Lett.*, **75**(20), 3594. https://doi.org/10.1103/PhysRevLett.75.3594.
Yadav, N., Yadav, A. and Kumar, M. (2015), *An Introduction to Neural Network Methods for Differential Equations*, Springer, Netherlands.
Ye, J., Becque, J., Hajirasouliha, I., Mojtabaei, S.M. and Lim, J.B.P. (2018). "Development of optimum cold-formed steel sections for maximum energy dissipation in uniaxial bending". *Eng. Struct.*, **161**, 55-67. https://doi.org/10.1016/j.engstruct.2018.01.070.
Ye, J., Hajirasouliha, I., Becque, J. and Eslami, A. (2016). "Optimum design of cold-formed steel beams using particle swarm optimisation method", *J. Constr. Steel Res.*, **122**, 80-93. https://doi.org/10.1016/j.jcsr.2016.02.014.
Yentis, R. and Zaghloul, M. (1996), "VLSI implementation of locally connected neural network for solving partial differential equations", *IEEE T. Circ. Syst. I*, **43**(8), 687-690. https://doi.org/10.1109/81.526685.

TK