MATHEMATICA MONTISNIGRI
Vol LVI (2023)

# MACHINE LEARNING AND NEURAL NETWORKS BASED APPROACH FOR DEFLECTION PREDICTION OF EULER-BERNOULLI BEAM EQUATIONS

**Z. RASULOV[1] AND U. (BABUSCU) YESIL [1]**

[1] Yildiz Technical University, Istanbul, Turkey
*Corresponding author. E-mail: rasulovzaurr@gmail.com

**DOI: 10.20948/mathmontis-2023-56-8**

**Summary.** Beam-like structures are widespread but essential systems that have been extensively studied for centuries. Although several proposed solutions are effective, the time consumption and the difficulty of reconstructing the problem are the major disadvantages of these methods. This paper offers a new methodology for finding solutions to beam problems based on Machine Learning and Neural Networks with different optimization algorithms. Various regression models are compared on numerically stimulated Euler-Bernoulli beam modelling.

## 1 INTRODUCTION

Beams are structural components that are essential in several structural systems. They have been extensively investigated for more than a century (see [1-4] and the references therein). The calculation of deflection is crucial because of its importance in operation serviceability. Hence, several numerical approximation algorithms have been found to find solutions to such problems. One of the most prominent and efficient methodologies for finding solutions to differential equations and mathematical models is the Finite Element Method (FEM). By discretizing the domain, defining finite elements (FEs), and generating the mesh, the deflections of the beam are calculated. When the problem size becomes large, time needed to solve the resulting systems may range from hours to days, and if the input parameters need to be adjusted, even slightly, the simulations have to be re-done from scratch. Recent advances in machine learning algorithms and their successful applications in various fields demonstrate that, if properly chosen and trained, these models can significantly improve conventional techniques.

Although many methodologies have been introduced for better convergence, time is vital in solving beam problems. For this reason, researchers have been investigating the implementation of Machine Learning (ML) and Neural Networks (NNs) models in such problems for the last several years. The possibility of predicting the bending moment of Euler–Bernoulli beam and drilling rise with Linear Regression (LR) and Recurrent Neural Networks was investigated in [5]. The methodology for finding the approximations using NNs with a new optimization algorithm was found by Tianyu et.al. [6]. The prediction of steel-concrete composite beams was mentioned by Thirumalaiselvi et al. [7], while new ML implementations to predict reinforced concrete beams with and without stirrups via applying Random Forest (RF) were proposed by Junfei et al [8]. Yang et al. [9] introduced a novel method for finding the optimal initial electron beam parameters of a Linac, which aimed the

**2020 Mathematics Subject Classification:** 74G15, 74S05, 74S99.
**Key words and Phrases:** Machine Learning, Neural Network, FEM, Euler-Bernoulli Beams, Deflection.

usage of statistical and ML algorithms for Monte-Carlo model. The traditional methodology to detect vibration-based destructions in civil structures to the implementations by ML and DL was modified by Avci et.al. [10]. The implementation of a data-driven approach to the shear strength of the SFRC beam was described by Rahman et. al. [11]. They applied Regression models and Ensemble Tree algorithms. On the other hand, Tsiatas and Aristotelis described the prediction of laminated composited beams using ML algorithms [12], and the implementation of ML algorithms to predict the residual stress in electron beams by Monte Carlo simulations was introduced by Debasish, et al. [13]. In [14], Ye and Yi introduced the application of a convolutional neural network to determine the working-condition beam pumping units using batch normalization. Furthermore, in [15], Mahesh et al. introduced an ML approach to predict the stress results of quadratic tetrahedral elements via the stress results calculated by the FEM.

In addition, several researchers introduced the application of DL to the solution of ordinary and partial differential equations (ODE, PDE). Lagaris et.al. presented and generalized the application of NNs with the discretized domains and usage of trial functions for solving the first and second-order ordinary differential equations, systems of equations, and partial differential equations [16]. Han et al. [17] implemented NNs for higher order differential equations using the sigmoid activation function with the Nelder-Mead simplex algorithm, whereas the methodology for solving the problems based on the Galerkin methodology was proposed by Sirignano and Spilopoulos [18]. Moreover, Raissi et. al. improved the existing methods and introduced physics-based deep learning [19].

The primary purpose of this paper is to introduce a new methodology for finding the solutions to Euler-Bernoulli beam problems and predicting the deflections. We use traditional ML models, such as LR, PR. DT, RF, KNN and NNs. After giving the overview of the Euler-Bernoulli beam problem and the FEM overview, we describe the methodology and the optimization algorithms used in the calculations. Finally, we provide numerical examples by comparing the results found by the proposed methods and FEM or exact solutions. We used MATLAB for FEM calculations and Python for ML and DL implementation.

## 2 FEM FORMULATION OF THE EULER-BERNOULLI BEAM PROBLEM

The general definition of Euler-Bernoulli beam equation is as follows

$$\big(b(x)w''(x)\big)'' = f(x) \tag{1}$$

for $x \in [0, L]$ [20]. Here $w(x)$ is the transverse deflection of the beam, L is the length of the beam, $f(x)$ is the transversely distributed load and $b(x) = EI$ is the product of the modulus of elasticity $E$ and the moment of inertia $I$ of the beam. Appropriate boundary conditions should be satisfied by $w(x)$ and for solving the problem four boundary conditions are needed. For the weak solution of the beam, an isolated typical element $\Omega^e = (x_e, x_{e+1})$ is selected and constructed the weak form over this element. Eq. (2) is obtained by applying the Galerkin method [20].

$$\int_{x_e}^{x_{e+1}} \Big[\big(b(x)w''(x)\big)'' - f(x)\Big]\varphi(x)dx = 0 \tag{2}$$

Here $\varphi(x)$ is a test function that is twice differentiable with respect to x. The first term of the Eq. (2) is integrated twice by parts to trade two differentiations to the test function $\varphi$, while

retaining two derivatives of the dependent variable $w$; i.e., the differentiation is distributed equally between the test function and the dependent variable. The weak form can be expressed, by applying the partial integrating twice, as;

$$B(\varphi, w) - I(\varphi) = 0 \tag{3}$$

where

$$B(\varphi, w) = \int_{x_e}^{x_{e+1}} b(x)\varphi''(x)w''(x)dx \tag{4}$$

$$I(\varphi) = \int_{x_e}^{x_{e+1}} \varphi(x)f(x)dx + \left[\varphi(x)(b(x)w''(x)' - \varphi'(x)b(x)w''(x))\right]\Big|_{x_e}^{x_{e+1}} = 0 \tag{5}$$

Hermite cubic interpolation (cubic spline) functions are used for Euler-Bernoulli beams which can be expressed as;

$$\phi_1^e = 1 - 3\left(\frac{x - x_e}{h_e}\right)^2 + 2\left(\frac{x - x_e}{h_e}\right)^3,$$

$$\phi_2^e = -(x - x_e)\left(1 - \frac{x - x_e}{h_e}\right)^2,$$

$$\phi_3^e = 3\left(\frac{x - x_e}{h_e}\right)^2 - 2\left(\frac{x - x_e}{h_e}\right)^3,$$

$$\phi_4^e = -(x - x_e)\left[\left(\frac{x - x_e}{h_e}\right)^2 - \frac{x - x_e}{h_e}\right] \tag{6}$$

where $x_{e+1} = x_e + h_e$.

The variational form Eq. (3) requires that the interpolation functions of an element are continuous with nonzero derivatives up to order two. A four-parameter polynomial should be selected for $w(x)$ because there are four conditions (two per node) in an element

$$w(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 \tag{7}$$

By using the continuity conditions; $w^e(x)$ is obtained by the Galerkin method as:

$$w^e(x) = \sum_{i=1}^{4} u_i^e \phi_i^e \tag{8}$$

where the functions $\phi_i^e$ are the cubic spline functions and the coefficients $u_i^e$ are generalized displacements. FEM model of the Euler-Bernoulli beam can be obtained by substituting the FE interpolation Eq. (6) for $w$ and the $\phi_i$ for the weight function $\varphi$ into the weak form Eq. (3). The *i*th algebraic equation of the FE model is;

$$\sum_{j=1}^{4} K_{ij}^e u_j^e = F_i^e \tag{9}$$

where

$$K_{ij}^e = \int_{x_e}^{x_{e+1}} b(x)\frac{d^2\phi_i^e}{dx^2}\frac{d^2\phi_j^e}{dx^2}dx,$$

$$F_i^e = \int_{x_e}^{x_{e+1}} \phi_i^e \, f dx + Q_i^e \tag{10}$$

In Eq, (10) $Q_i^e$ denote the shear forces (where i is an odd number), bending moments (where i is an even number), $[K^e]$ is the stiffness matrix and $\{F^e\}$ is the force vector of the beam element [20].

Numerical approximations of the problems are found using FEM with Hermite polynomials. To avoid high discrepancy between the targets and predictions, a dataset with high number of finite elements are created. The usage of Hermite polynomials verify the $C^1$-continiutiy of the model. Because of being 3rd order polynomials, the second derivative of the functions is linear and reach maximum at endpoints.

## 3 MEASUREMENTS SIMULATION

ML models employ large amounts of data to obtain a strong performance. The models are applicable for calculation of complex structures with high accuracy and less time of implementation.

For the LR, PR, DT, RF and k-NN, training, validation and test sets have been generated by giving an external force and solving the equations by using their exact/approximate solutions. Afterward, the displacements at some points are calculated, and deployed into our models. Following the validation of the usage of models on validation sets, the values of deflections are predicted through test sets. The logic of creating the datasets is described in the end of the Section 2. In case of NNs, PyTorch library is used, in which the differential equations to the left hand side are deployed, and the exact/approximate equation regarding load $f(x)$ to its right hand side. Thus, the optimization algorithms are implemented to predict the solutions by creating the nodes, which accept the values obtained through the discretization by NNs.

The models used for calculation are summarized below.

### 3.1 Linear Regression

Linear Regression [21] is a straightforward and efficient ML algorithm which explains the model's linear relation and the variability of the data. Following the satisfaction of Gauss-Markov assumptions, the model should be optimized using the Stochastic Gradient Descent (SGD) methodology.

A hyperplane can fit the n-dimensional space, where n is the number of independent variables. The general form of the linear equation applied in the LR model is as follows:

$$\hat{y} = \sum_{i=0}^{k} \alpha_k x + \varepsilon \tag{11}$$

In Eq. (11), $\alpha_k$ is the regression coefficients, where $\alpha_0$ is the bias of the model, $x$ is a vector of independent variables, $\varepsilon$ is an error term and $\hat{y}$ is a predicted value of the function. Once defined the equation, the cost function of the LR, Root Mean Squared Error (RMSE) function (Eq. (12)), is minimized.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2} \tag{12}$$

In Eq. (12), $y$ is the exact solution, and $\hat{y}$ is the predicted solution.

### 3.2 Polynomial Regression

Polynomial Regression [22] is a model used to determine the nonlinear relationships between the variables. Fitting the algorithm, it finds the optimum values for the problems. It is

worth mentioning that PR reduces the unbiased variance of variables under the Gauss-Markov assumptions' condition.

The general form of PR is given as:

$$\hat{y} = \sum_{i=0}^{n} \beta_n x^n + \varepsilon \tag{13}$$

### 3.3 Decision Trees

Decision Trees [23] is a supervised ML algorithm, which aims to generate a tree model from the training data by continuously questioning the input data. Once the input data has reached the end of the tree, the target is determined.

To begin with, the feature space $R^n$ is partioned iteratively according to the splitting attribute. After partitioning the space, each region in the final section is determined as a target. Reaching these regions, the model automatically determines the target value. The equation for the DT is given as:

$$h(x) = \sum_{i=1}^{L} \alpha_i \tag{14}$$

where $x \in R_j$, and $R_j$ is the disjoint region of the ending of DT.

### 3.4 Random Forest

Another supervised algorithm based on the combination of several decision trees is random Forest [24]. The tree classifiers are selected randomly from the inputs. Then, they are used to design DTs using the replacement (bootstrapping) method.

According to the investigations, a positive correlation between the number of trees and the accuracy rate was observed. Obtaining the results from the trees, it finds the average of the results using the following formula:

$$y = \frac{1}{N} \sum_{i=1}^{n} \hat{y}_n(x) \tag{15}$$

where N is the total number of trees and $\widehat{y_n}$ is the average of each tree.

### 3.5 k-nearest neighbour (KNN)

k-nearest neighbour [25] algorithm is a non-parametric and lazy algorithm we use for regression. The model predicts the value of a new data by estimating the distances between the closest k values. The mathematical definition of the algorithm is:

**Definition 1.** Assume the pairs $(x_1, y_1) \dots (x_n, y_n)$. The values are included in the set of $R^d \times \{1, 2\}$, where d is the distance between the values and y is the label of x, such that $x|y = r \sim P_r$ for $(r = 1,2)$ and $P_r$ – probability distribution. Taking into account any norm $||\cdot||$ on $\mathbb{R}^d$ and $x \in \mathbb{R}^d$, assume that $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ is the reordering of the training data such that $\left||x_1 - x|\right| \leq \left||x_2 - x|\right| \leq \left||x_3 - x|\right| \leq \cdots \leq ||x_n - x||$ [25].

### 3.6 Neural Networks

Consider the following fourth order differential equation;

$$H\left(\vec{x}, w(\vec{x}), \nabla w(\vec{x}), \nabla^2 w(\vec{x}), \nabla^3 w(\vec{x}), \nabla^4 w(\vec{x})\right) = 0 \tag{16}$$

The boundary conditions can be of any form for the differential equation. In the equation, $\vec{x} = (x_1, x_2, x_3 \dots x_n) \in R^n$, and $D \subset R^n$ represents the domain. To find the solution of Eq. (16), we apply the collocation method, after which the domain D and its boundaries S are discretized into $\widehat{D}, \hat{S}$ respectively. Following this, the Eq. (16) is rewritten as

$$H\left(\overrightarrow{x_l}, w(\overrightarrow{x_l}), \nabla w(\vec{x}_l), \nabla^2 w(\overrightarrow{x_l}), \nabla^3 w(\overrightarrow{x_l}), \nabla^4 w(\overrightarrow{x_l})\right) = 0 \qquad (17)$$

In NNs, the primary aim is the minimization of the cost function. To begin with, the trial function should be introduced.

$$(\vec{x_i}) = (\vec{x_i}) + (\vec{x_i}, (\vec{x_i}, \vec{p})) \qquad (18)$$

where $N(\vec{x_i}, \vec{p})$ is a single-output FFN function with parameters $\vec{p}$ and the input vector $\overrightarrow{x_i}$. In Eq. (18), the first term should satisfy the boundary conditions of the problem, while the second term is a function dealing with minimization problems. By Eq. (17) and (18), the minimization function is defined as:

$$\min_{\vec{p}} \sum_{\overrightarrow{x_i} \in D} \left(H\left(\overrightarrow{x_l}, w_t(\overrightarrow{x_l}), \nabla w_t(\overrightarrow{x_l}), \nabla^2 w_t(\overrightarrow{x_l}), \nabla^3 w_t(\overrightarrow{x_l}), \nabla^4 w_t(\overrightarrow{x_l})\right)\right)^2 \qquad (19)$$

The minimization of Eq. (19) is considered as training of NNs where the error approaches zero. The estimation of the error term involves both the minimization of the function and its derivatives. Therefore, to calculate the gradient of the error functions, we find the gradient of the derivatives of the error functions.

Let us assume a multilayer perceptron with n input units, where a hidden layer contains sigmoid units and a linear output unit. For an input vector, the output of the network is

$$N = \sum_{i=1}^{H} v_i s(z_i) \qquad (20)$$

where $z_i = \sum_{j=1}^{N} w_{ij} x_j + u_i$ and $w_{ij}$ shows the weight from the input unit $j$ to the hidden unit $i$, $v_i$ represents the weight from the hidden unit $i$ to the output, $u_i$ is the bias of hidden unit $i$ and $s(z) = \tanh(z)$. Moreover, H is the number of sigmoid units.

Finding the derivatives of Eq. (20),

$$\frac{\partial^k N}{\partial x_i^k} = \sum_{i=1}^{H} v_i w_{ij}^k s_i^k \qquad (21)$$

Considering Eq. (21),

$$\frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \dots \frac{\partial^{\alpha_n}}{\partial x_n^n} N = \sum_{i=1}^{n} v_i s_i^A \prod_{k=1}^{n} w_{ik}^{\lambda_k} \qquad (22)$$

where $A = \sum_{i=1}^{n} \alpha_i$. By Eq. (22), the derivative of the network with respect to its inputs equals to FFANN with one hidden layer.

After finding the derivative of the error from the network parameters, it is easy to implement minimization algorithms, which should be chosen according to the cost functions. Furthermore, the error rate should be minimized by:

$$E[\vec{p}] = \sum_i [\frac{d^4 w_t}{dt} - f(x_i, w(x_i), w'(x_i), w''(x_i), w'''(x_i)] \qquad (23)$$

During the implementation of the NNs, we use different activation functions and optimization algorithms to compare the results and define the most feasible solution. In this

work, we use Limited-Memory BFGS (L-BFGS) [26], Adam [27], AdaGrad [28] and Adadelta [29] optimization algorithms.

### 3.6.1 L-BFGS Optimization Algorithm

Limited-memory BFGS (L-BFGS) optimization algorithm is based on the family of quasi-Newton methods approximating the BFGS.

L-BFGS uses the inverse Hessian matrix for variable calculation. The ability of the L-BFGS algorithm to store only vectors demonstrating the approximate values causes the model to have linear memory fitting better than other optimization models.

Let us define the search direction as $d_k = -H_k h_k$, where $H_k$ is the Hessian matrix generated by limited memory quasi-Newton method. Then, the mathematical definition of the methodology is written as

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T \tag{24}$$

where $\rho_k = \frac{1}{s_k^T y_k}, V_k = I - \rho_k y_k s_k^T$, $I$ is an identity matrix. Using the optimization algorithms, [26] introduced the L-BFGS method for nonlinear equations with the global convergence.

### 3.6.2 Adam Optimization Algorithm

An extension of the SGD, continuously updating weights more efficiently, is called Adam optimization algorithm. The mathematical definition of the optimization model is the combination of different gradient descent methodologies. The capability to increase the speed of the gradient descent algorithms using weighted averages of the derivatives is the reason for us to prefer this model.

Consider the momentum as

$$w_{t+1} = w_t - \frac{\alpha_t}{\sqrt{v_t + \varepsilon}} \frac{\partial L}{\partial w_t} \tag{25}$$

$$v_t = \beta v_{t-1} + (1 - \beta) \frac{\partial L}{\partial w_t} \tag{26}$$

In the Eq. (25) – (26), $w_t$ – weight, $\alpha_t$ – learning rate, $L$ – loss function, $v_t$ – sum of square of previous gradients, $\beta$ – step size, $\varepsilon$ – discrepancy Using Eq. (25) and (26), the following equation is obtained:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t} v_t - \beta_2 v_{t-1} - (1 - \beta_2) \left(\frac{\partial L}{\partial w_t}\right)^2 \tag{27}$$

Finally, the Adam optimizer can be written as

$$w_{t+1} = w_t - \widehat{m_t} \left(\frac{\alpha}{\sqrt{\widehat{v_t}} + \varepsilon}\right) \tag{28}$$

where $\widehat{m}_t = \frac{v_t}{1 - \beta_2^t}$ and $\hat{v}_t = \frac{v_t(1 - \beta_1^t)}{(1 - \beta_2^t) m_t}$.

### 3.6.3 Adagrad Optimization Algorithm

Adagrad is a modified SGD with a pre-defined learning rate. The mathematical definition of the algorithm is

$$w_{i+1} = w_i - \frac{\eta}{\sqrt{G_{i,i}}}\nabla_i \tag{29}$$

where $G_{i,i} = \sum_{i=1}^{t} \nabla_{i,j}^2$. Also, it is worth mentioning that the denominator in Eq. (29) represents L2 norm.

### 3.6.4 AdaDelta Optimization Algorithm

Another formulation of AdaGrad is AdaDelta, in which the learning rate decreases faster. Moreover, the learning rate is one for some problems, which is still being researched. L

Let us define the leaky updates as following:

$$s_t = \rho s_{t-1} + (1-\rho) g_t^2 \tag{30}$$

where $s_t$ stores a leaky average of the second moment of the gradient. Consider the following equation

$$x_t = x_{t-1} - g_t' \tag{31}$$

where $g_t' = \frac{\sqrt{\Delta x_{t-1}+\varepsilon}}{\sqrt{s_t+\varepsilon}} g_t$, Then, the Adadelta algorithm can be written as

$$\Delta x_t = \rho \Delta x_{t-1} + (1-\rho) g_t'^2 \tag{32}$$

## 4 NUMERICAL RESULTS AND DISCUSSION

In this section, three numerical problems are solved to show the accuracy and efficiency of the proposed methodologies.

**Problem 1.** Consider the following beam equation for $x \in [0,2]$;

$$\left(e^x w''(x)\right)'' + e^x w(x) = e^x(x^7 - 8x^6 + 66x^5) + e^x(148x^4 - 584x^3 - 384x^2) + e^x(1440x - 576) \tag{33}$$

with the following boundary conditions;

$$w(0) = w(2) = w(0) = w''(2) = 0 \tag{34}$$

The exact solution of the Problem 1[30] is

$$w(x) = x^3(x-2)^4 \,. \tag{35}$$

The trial function defined for the Problem 1 is:

$$w_t(x) = x^2(2-x)^3 N(x) \tag{36}$$

**Problem 2.** Consider the following beam problem given in Fig. 1, where load distribution varies linearly, and is defined as

$$q(x) = q_0\left(1 - \frac{x}{L}\right) \tag{37}$$

where $q_0 = 24\frac{kN}{m}$ and $L = 3\,m$. $M_0 = 0\,kNm, E = 200 \cdot 10^6\frac{kN}{m^2}$, $I = 29 \cdot 10^6\,mm^4, F_0 = 60\,kN$ for $x \in [0, L]$ [20].

The boundary conditions of the problem are:

$$w(0) = w'(0) = 0, w(3) = 0.1042, w'(3) = 0.1596 \tag{38}$$

With reference to the given data for Problem 2 the following model can be created [17].

$$5800w^{iv}(x) = 24\left(1 - \frac{x}{3}\right) \tag{39}$$

The exact solution of the Problem 2 is given in Eq. (40) [20].

$$w(x) = \frac{24\cdot 3^4}{120*5800}\left(10\left(\frac{x}{3}\right)^2 - 10\left(\frac{x}{3}\right)^3 + 5\left(\frac{x}{3}\right)^4 - \left(\frac{x}{3}\right)^5\right) + \frac{60\cdot 3^3}{6*5800}\left(-\left(\frac{x}{3}\right)^3 + 3\left(\frac{x}{3}\right)^2\right) \tag{40}$$

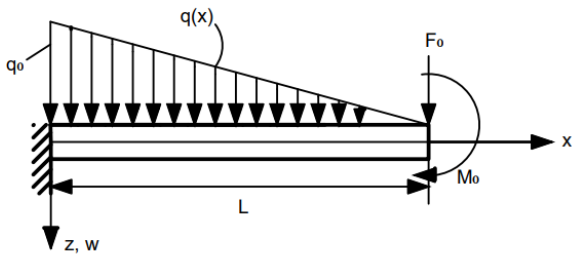


Figure 1. Cantilever Beam Problem under linearly varying load distribution.

The trial function defined for the Problem 2 is:

$$w_t(x) = 0.01003x^3 - 0.0185x^2 + x^2(3-x)^2N(x) \tag{41}$$

**Problem 3.** Consider the following Euler-Bernoulli beam problem given in Fig. 2. The problem data for using the ML prediction are created by FEM data because of unknown exact solution. The solutions are obtained for 5, 6, 10 and 15 number of FEs to provide the accuracy of the solution. Fig. 3 shows the mesh sensitivity for FEM solutions of Problem 3. It can be seen from the graph given in Fig. 3 that deflection function converges each other for 5 and more than 5 finite elements.

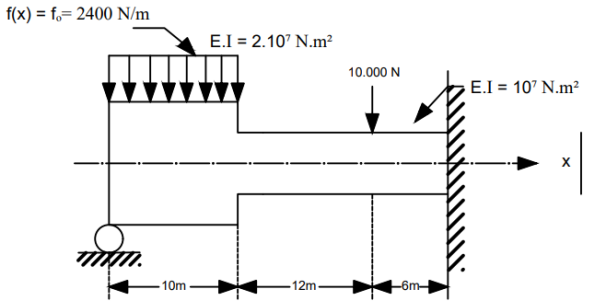


Figure 2. Sample Euler-Bernoulli beam problem

FEM solution of deflection function $\left(w(x)\right)$ for minimum number of FE (3 number of FE) is given in Eq. (42).

$$w(x)=\begin{cases}\left(\begin{matrix}0.03856\left[-(x-0)\left(1-\left(\frac{x-0}{10}\right)^2\right]-0.2808\left[3\left(\frac{x-0}{10}\right)^2-2\left(\frac{x-0}{10}\right)^3\right]+\\ 0.01214\left[-(x-0)\left[\left(\frac{x-0}{10}\right)^2-\left(\frac{x-0}{10}\right)\right]\right]\end{matrix}\right); & 0\le x\le 10m\\ \left(\begin{matrix}-0.2808\left[1-3\left(\frac{x-10}{12}\right)^2+2\left(\frac{x-10}{12}\right)^3\right]+0{,}01214\left[-(x-10)\left(1-\frac{x-10}{12}\right)^2\right]-\\ 0{,}1103\left[3\left(\frac{x-10}{12}\right)^2-2\left(\frac{x-10}{12}\right)^3\right]-0{,}02752\left[-(x-10)\left[\left(\frac{x-10}{12}\right)^2-\left(\frac{x-10}{12}\right)\right]\right]\end{matrix}\right); & 10\le x\le 22m\\ -01103\left[1-3\left(\frac{x-22}{6}\right)^2+2\left(\frac{x-22}{6}\right)^3\right]-0{,}02752\left[-(x-22)\left(1-\frac{x-22}{6}\right)^2\right]; & 22\le x\le 28m\end{cases} \quad (42)$$

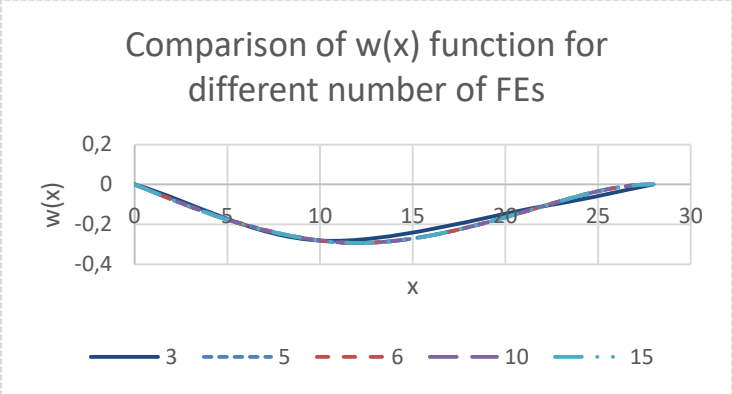


Figure 3. Mesh Sensitivity for FEM solutions of Problem 3.

The absolute errors between the models and the exact (for Problem1-2)/FEM (for Problem 3) solutions are compared in Tab. 1 (a-b-c), respectively, for Problem (1-2-3). As is clear from these tables, all methodologies converge strong. Still, the errors obtained by PR are approaching zero faster when compared to other models, which proves the efficiency of this methodology. The solutions obtained by DT are robust, while the solutions of NNs and RF are almost the same. However, the weak convergence of the models are observed for LR and KNN models.

To compare the performances of the models, MAEs of the algorithms are calculated in Tab. 2 for all Problems. It is clear that the performance of PR is the best for all the models as the values of the MAE is the smallest. The performance of DT is also strong, while the performances of LR and KNN are weak.

| x/Mod | PR | DT | NNs | RF | LR | KNN |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.4 | 2.54E-13 | 7.53E-08 | 1.12E-08 | 3.29E-06 | 6.36E-05 | 6.43E-06 |
| 0.8 | 1.31E-13 | 7.65E-08 | 3.65E-08 | 8.91E-06 | 4.26E-04 | 7.86E-06 |
| 1.2 | 1.07E-13 | 2.12E-07 | 1.16E-07 | 1.63E-06 | 8.36E-05 | 3.78E-06 |
| 1.8 | 5.63E-14 | 5.42E-08 | 6.45E-09 | 4.25E-07 | 2.85E-05 | 8.92E-05 |
| 2 | 0 | 0 | 0 | 5.63E-37 | 2.75E-25 | 9.36E-76 |

(a) For Problem 1

| x/Model | PR | DT | NNs | RF | LR | KNN |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.6 | 6.61E-17 | 8.75E-10 | 2.86E-09 | 1.26E-06 | 3.58E-04 | 4.78E-06 |
| 1.2 | 9.71E-17 | 3.23E-09 | 2.59E-09 | 9.29E-06 | 6.28E-03 | 2.67E-05 |
| 1.8 | 4.15E-16 | 1.95E-09 | 4.03E-06 | 1.25E-05 | 4.67E-03 | 4.78E-05 |
| 2.4 | 1.80E-16 | 7.77E-10 | 2.42E-06 | 7.09E-05 | 5.85E-03 | 4.60E-05 |
| 3.0 | 4.89E-17 | 7.97E-25 | 4.24E-07 | 5.79E-05 | 8.42E-03 | 1.43E-05 |

(b) For Problem 2

| x/Model | PR | DT | NNs | RF | LR | KNN |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 3.26E-16 | 5.46E-12 | 4.73E-11 | 8.35E-11 | 9.00E-05 | 1.26E-06 |
| 10 | 4.13E-14 | 3.28E-12 | 8.73E-10 | 4.26E-10 | 6.01E-05 | 7.94E-06 |
| 16 | 6.42E-14 | 7.53E-12 | 1.73E-10 | 9.67E-09 | 1.52E-04 | 4.26E-05 |
| 22 | 5.72E-13 | 5.72E-12 | 3.26E-10 | 3.67E-09 | 4.78E-04 | 1.27E-05 |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 |

(c) For Problem 3

Table 1. Comparisons of errors by exact solution with different ML and DL models.

| Model | Problem 1 | Problem 2 | Problem 3 |
|---|---|---|---|
| **PR** | 8.38E-14 | 6.82E-17 | 1.13E-13 |
| **DT** | 5.77E-08 | 1.16E-07 | 2.66E-11 |
| **NNs** | 9.02E-08 | 1.36E-06 | 2.27E-08 |
| **RF** | 2.55E-06 | 8.54E-05 | 9.43E-05 |
| **KNN** | 4.24E-05 | 9.59E-05 | 4.35E-04 |
| **LR** | 8.04E-05 | 5.66E-03 | 8.55E-04 |

Table 2. MAEs of the ML and DL models for all problems.

In turns of NNs, the same 4 optimization algorithms are applied for all problems. Obviously, from Tab. 3 (a-b-c), respectively for Problem (1-2-3), the L-BFGS optimization algorithm shows the best convergence to the exact solutions when comparing to other optimization algorithms. While the results found via Adam and AdaGrad are moderate, the performance of Adadelta is not satisfiable.

| x/Model | L-BFGS | Adam | Adagrad | Adadelta |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0.4 | 4.32E-08 | 3.53E-07 | 7.46E-06 | 1.27E-03 |
| 0.8 | 9.80E-08 | 7.78E-07 | 4.73E-05 | 2.75E-03 |
| 1.2 | 9.32E-08 | 7.58E-07 | 9.54E-05 | 4.42E-03 |
| 1.6 | 2.83E-07 | 8.14E-06 | 8.14E-05 | 1.02E-02 |
| 2.0 | 0 | 0 | 3.52E-27 | 7.52E-27 |

(a) For Problem 1

| x/Model | L-BFGS | Adam | Adagrad | Adadelta |
|---|---|---|---|---|
| 0.0 | 0 | 0 | 0 | 0 |
| 0.6 | 4.43E-08 | 6.80E-07 | 4.14E-06 | 4.55E-04 |
| 1.2 | 4.54E-08 | 6.12E-07 | 2.46E-06 | 4.52E-03 |
| 1.8 | 2.25E-07 | 4.63E-06 | 4.57E-05 | 8.52E-03 |
| 2.4 | 6.89E-07 | 1.70E-06 | 4.33E-05 | 7.40E03 |
| 3.0 | 6.25E-07 | 9.78E-06 | 1.06E-04 | 4.00E-03 |

(b) For Problem 2

| x/Model | L-BFGS | Adam | Adagrad | Adadelta |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 5 | 4.73E-11 | 4.25E-10 | 4.25E-10 | 1.34E-07 |
| 10 | 8.73E-10 | 3.15E-10 | 7.89E-09 | 4.32E-07 |
| 16 | 1.73E-10 | 6.47E-09 | 3.62E-08 | 7.36E-06 |
| 22 | 3.26E-10 | 5.46E-09 | 6.73E-08 | 1.78E-06 |
| 28 | 0 | 0 | 0 | 0 |

(c) For Problem 3

Table 3. Comparisons of errors obtained by various optimization algorithms of the NNs.

## 5 CONCLUSION

In this paper, new approximation algorithms for finding the numerical solutions to Euler-Bernoulli beam problems are investigated. The significant problems of the algorithms defined earlier are the time consumption and lack of accuracy. As a result of an increasing propensity in technology and ML and DL, a novel approximation approach is developed. In the paper, the novelties in this field are discussed, the definitions of FEM, and the ML and DL models. Following this, three numerical examples to show the efficiency and applicability of the proposed methods are provided. The obtained solutions either with the exact solutions or with the FEM solutions are compared.

Concerning the obtained results, we conclude that the proposed methodologies converged to the exact solutions faster and easier compared to the literature results. Some algorithms show strong convergence, while others converged weakly. Because of the vitality of convergence, the most efficient methodology for finding the solutions of the Euler-Bernoulli beam equations is PR. Moreover, the results by DT treated with the hyperparameters and NNs with L-BFGS optimization algorithm are also satisfactory. Even though the results by RF with the regularization parameters fluctuated, the results found via LR and KNN are not feasible.

Considering the results found in the paper, it can be summarized that ML and DL algorithms are be helpful and valuable when predicting the solutions to Euler-Bernoulli Beam Problems because of its accuracy and speed. Thus, the implementation of ML and DL models should also be expanded to the Mathematical and Physical fields. We are sure that this work opens the doors for many scientists and can provide better and easier solutions for future papers.

## REFERENCES


[1] O.A. Bauchau and J.I. Craig, *Structural Analysis. Solid Mechanics and Its Applications*, Springer, Vol.163 (2009).

[2] C.M. Wang, "Timoshenko beam-bending solutions in terms of Euler-Bernoulli solutions", *J. Eng. Mech*, **121** (6), 762-765 (1995).

[3] Z. Ding, "Natural frequencies of rectangular plates using a set of static beam functions in Rayleigh-Ritz method", *Journal of Sound and Vibration*, **189** (1), 81-87 (1996).

[4] D.A. Saravanos and P. R. Heyliger, "Mechanics and computational models for laminated piezoelectric beams, plates and shells", *Appl. Mech. Rev.*, **52** (10), 305-320 (1995).

[5] E. Celledoni, H.S. Gustad, N. Kopylov and H.S. Sundklakk, Predicting Bending Moments with Machine Learning. In: Nielsen, F., Barbaresco, F. (eds) *Geometric Science of Information. GSI 2019*. Lecture Notes in Computer Science, Springer, Cham, Vol. 11712, (2019).

[6] T.Wang, W.A. Altabey, M. Noori and R. Ghiasi, "A deep learning based approach for response prediction of beam-like structes", *Structural Durability and Health Monitoring*, **14** (4), 315-338 (2020).

[7] A. Thirumalaiselvi, V. Mohit, N. Anandavalli and J. Rajasankar, "Response prediction of laced steel-concrete composite beams using machine learning algorithms", *Struct. Eng. And Mech.*, **66** (3), 399-409 (2018).

[8] J. Zhang, Y. Sun, G. Li, Y. Wang, J. Sun and J. Li, "Machine-learning-assisted shear strength prediction of reinforced concrete beams with and without stirrups", *Engineering with Computers*, **38**, 1293-1307 (2022).

[9] H.J. Yang, T.H. Kim, T. Schaarschmidt, D.W. Park, S.H. Kang, H.T. Chung and T.S. Suh, "A multivariate approach to determine electron beam parameters for a Monte Carlo 6 VM Linac model: Statistical and machine learning methods", *Physica Medica*, **93**, 38-45 (2022).

[10] O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj and D. J. Inman, "A review of vibration-based damage detection in civil structures: From traditional Methods to Machine Learning and Deep Learning applications", *Mech. Syst. And Signal Process*, **147**, 107077 (2021).

[11] J. Rahman, K. S. Ahmed, N. I. Khan, K. Islam and S. Mangalathu, "Data-driven shear strength prediction of steel fiber reinforced concrete beams using machine learning approach", *Eng. Struct.*, **233**, 111743 (2021).

[12] G.C. Tsiatas, K. Sotiris and E.C. Aristotelis, "Predicting the response of laminated composite beams: A comparison of machine learning algorithms", *Front. Built. Environment*, **8** (2022).

[13] D. Das, A.K. Das, D. Pratihar, G. Roy, "Prediction of residual stress in electron beam welding of stainless steel from process parameters and natural frequency of vibrations using machine-learning algorithms", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, **235** (11), 2008-2021 (2021).

[14] Z. Ye and Q.Yi, "Working-condition diagnosis of a beam pumping unit based on deep-learning convolution network", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, **236** (5), 2559-2573 (2022).

[15] S. Mahesh, S. Marco, R.S. Sharma, M. Praveenkumar, V. Wadagavi, L. Subbarao, "A machine learning approach to predict the stress results of quadratic tetrahedral elements", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, **236** (2), 1128-1135 (2022).

[16] I.E. Lagaris, A. Likas and D.I. Fituadus, "Artificial neural networks for solving ordinary and partial differential equations", *IEEE tractions on neural networks*, **9** (5), 987-1000 (1998).

[17] H. Jiequn. A. Jentzen and E. Weinan, "Solving high-dimensional partial differential equations using deep learning", *Proc. Natl. Acad. Sci.,* **115** (34), 8505-8510 (2018).
[18] J. Sirignano and K. Spilopoulos, "DGM: A deep learning algorithm for solving partial differential equations", *J. Comput. Phys.,* **375**, 1339-1364 (2018).
[19] M. Raissi, P. Perdikaris and G. Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations", arXiv preprint arXiv. 1711.10561 (2017).
[20] J.N. Reddy, *Introduction to the finite element method*, McGraw-Hill Education (2019).
[21] D.C. Montgomery, A.P. Elizabeth and G.G. Vining, *Introduction to linear regression analysis*, John Wiley & Sons (2021).
[22] E. Ostertagova, "Modelling using polynomial regression", *Procedia Engineering*, **49**, 500-506 (2012).
[23] Y. Song and L.U. Ying, "Decision tree methods: applications for classification and prediction", *Shanghai archives of physchiatry*, **27** (2), 130-135, (2015).
[24] A. Parmar, R. Katariya and V. Patel, "A review on random forest: An ensemble classifier". In: Hemanth, J., Fernando, X., Lafata, P., Baig, Z. (eds) *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*. Lecture Notes on Data Engineering and Communications Technologies, Springer, Cham, Vol 26. (2019).
[25] L.E. Peterson, "K-nearest neighbor", *Scholarpedia*, **4** (2), 1883, (2009).
[26] G. Yuan, Z. Wei and S. Lu, "Limited memory BFGS method with backtracking for symmetric nonlinear equations", *Math. And Comp. Model*, **54** (1-2), 367-377 (2011).
[27] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization", arXiv prepring arXiv: 1412.6980.2014.
[28] L. Anges and S. Francis, "Adagrad – an optimizer for stochastic gradient descent", *Int. J. In. Comput. Sci*, **6** (5), 566-568 (2019).
[29] M.D. Zeiler, "Adadelta: an adaptive learning rate method", arXiv preprint arXiv:1212.5701.2012.
[30] Y. Sarac, "On Approximate Solution of the Euler-Bernoulli Beam Equation via Galerkin Method", *Erzincan University Journal of Science and Technology*, **1** (2), 341-346, (2018).

Received, December 6, 2022