A Machine Learning Model for Beam Deflection Curve Prediction: A Random Forest Approach with Multi-Material Validation

LUIZ CARLOS BRANDÃ£O JUNIOR[1]
RICARDO RODRIGUES MAGALHÃ£ES[1]

UFLA - Universidade Federal de Lavras
DCC - Departmento de Ciência da Computação
P.O. Box 3037 - Campus da UFLA 37200-000 - Lavras (MG)- Brazil
[1](luiz.junior11@estudante.ufla.br)
[2](ricardorm@ufla.br)

Abstract. Numerical simulation of complex engineering systems, such as those modeled using the Finite Element Method (FEM) or the Discrete Element Method (DEM), is often computationally intensive, limiting extensive parametric studies or optimization efforts. Surrogate models offer a promising alternative by enabling accelerated predictions. This work presents the development and rigorous validation of a machine learning (ML)-based methodology for creating surrogate models capable of predicting full structural deformation curves, point-by-point. To isolate and validate the ML approach, the classic case of a cantilever beam under a concentrated load at its free end was employed, for which the analytical solution, based on Euler-Bernoulli theory (including self-weight effects), is well established. A synthetic dataset was programmatically generated by calculating the analytical deflection at 51 equally spaced points along the beam length (from $x = 0$ to $x = L = 2.0$ m) for 13 distinct materials (varying Youngâs modulus, density, Poissonâs ratio, and yield strength), resulting in 663 records for a fixed beam geometry. A Random Forest regression model, trained on 80% of the dataset (530 points), was developed to map material properties and spatial position x to local deflection y. Evaluation on the test set (133 points) demonstrated high predictive accuracy, achieving a coefficient of determination (R^2) of 0.9991, a mean absolute error (MAE) of 0.2105 mm, and a root mean squared error (RMSE) of 0.4605 mm. An Out-of-Bag (OOB) R^2 score of 0.9983 further corroborated the model's generalization capability. The importance of this validation step, prior to applying the methodology to complex simulations where responses are obtained at discrete points, is discussed. The results demonstrate that the proposed methodology is robust and promising for developing fast and accurate surrogates for discretized structural response prediction.

Keywords: Cantilever Beam, Computational Engineering, Deformation Curve, Discrete Element Method, Finite Element Method, Machine Learning, Model Validation, Regression, Surrogate Model.

(Received June 9th, 2025 / Accepted June 28th, 2025)

1 Introduction

The analysis of mechanical behavior in structures and particulate systems is fundamental across various fields, including civil, mechanical, and materials engineering. Numerical methods such as the Finite Element Method (FEM) [8] and the Discrete Element Method (DEM) [9] enable high-fidelity simulation of complex scenarios, capturing geometric and material nonlinearities. However, the associated computational costs—particularly for three-dimensional or transient analyses—can be prohibitive. This limitation restricts extensive parametric studies, sensitivity analyses, design optimizations,

and, crucially, the calibration of model parameters [10]. For instance, parameter calibration in DEM is particularly challenging due to the complex relationships between microscopic model parameters and the observed macroscopic response [11].

In this context, surrogate models emerge as an effective means to mitigate computational demands [12]. Built using statistical or ML techniques, surrogate models learn to map system inputs (e.g., design parameters, material properties, boundary conditions) to outputs (e.g., structural responses, stress/strain fields) based on a limited set of simulations or experimental data. Once trained, surrogate models provide near-instantaneous predictions, enabling analyses that would otherwise be computationally infeasible. The application of ML techniques to create surrogate models has proven especially promising across various domains of computational engineering, including the acceleration of DEM parameter calibration [11, 13, 14].

Despite the clear potential of ML-based surrogate models, a critical challenge remains: ensuring their accuracy and generalization capability. It is imperative to validate that the ML model captures the underlying physical relationships rather than merely memorizing the training data, particularly before applying it to complex problems where the ground truth is difficult or expensive to obtain. Establishing confidence in the ML methodology thus necessitates rigorous validation using scenarios with known analytical solutions.

This work addresses this need by developing and rigorously validating an ML-based pipeline for predicting structural response curves, point-by-point. The cantilever beam under a concentrated tip load serves as the benchmark case, offering a well-established analytical solution via Euler-Bernoulli beam theory. We demonstrate how a Random Forest model [15], trained on synthetically generated data (incorporating self-weight effects), can accurately learn the relationship between material properties and spatial position along the beam to predict local deflection.

The primary objective of this study is to present a validated methodology for constructing ML-based surrogate models to predict deformation curves. The rigorous validation in this fundamental case provides a proof of concept, demonstrating ML's capability to capture physical phenomena with both efficiency and precision. This paves the way for the methodology's application to more complex and computationally demanding simulations.

The main contributions of this study are: (i) the explicit demonstration that a Random Forest model can accurately learn the functional response (deflection curve) of a classic structural problem from tabular data; (ii) the rigorous validation of the methodology in a canonical case with a known analytical solution, establishing a benchmark for future applications; and (iii) the presentation of a complete and reproducible pipeline (data generation $\rightarrow$ training $\rightarrow$ evaluation) that can be adapted to create surrogate models for complex numerical simulations.

The remainder of this paper is organized as follows. Section 2 details the methodology, including the analytical case, data generation, and the ML model. Section 3 describes the implementation and case study. Section 4 presents and discusses the results. Finally, Section 5 concludes the paper and suggests directions for future work.

1.1 Related Work

The use of machine learning to create surrogate models in engineering is a rapidly growing field. This section reviews other approaches for predicting beam deflection and discusses their context relative to our work.

Traditional ML models, such as Artificial Neural Networks (ANNs), Support Vector Machines (SVM), and the Random Forest (RF) algorithm used in this study, have been widely applied to regression and classification tasks in engineering [5]. For instance, RF and other tree-based models are often favored for their robustness on tabular data, their ability to handle nonlinear relationships without extensive data preprocessing, and their inherent mechanism for estimating feature importance [15].

More recently, deep learning techniques have been explored for structural analysis. Zhang et al. [3] used Convolutional Neural Networks (CNNs) to predict dynamic properties of beams directly from raw cross-section images. Their approach learns relevant geometric features automatically, bypassing the need for manual feature engineering like calculating the moment of inertia. This image-based method presents an alternative to our tabular, feature-based approach and is particularly powerful for design optimization where geometry is a variable. However, its main strength lies in complex geometries, and it may represent an unnecessary level of complexity for problems with fixed, simple cross-sections.

Another advanced technique gaining traction is Physics-Informed Neural Networks (PINNs). PINNs integrate the governing physical equations (as differential equations) into the neural network's loss function, ensuring the model's predictions adhere to physical laws. Sahin et al. [4] demonstrated the use of PINNs as a surrogate model of a reinforced concrete

beam, showcasing a path toward creating hybrid digital twins. The strength of PINNs is their ability to produce physically consistent results even with sparse data, but their implementation is more complex and requires the governing equations to be known and expressible in a differential form.

Our work addresses a critical gap identified in this landscape. While advanced models like CNNs and PINNs are powerful, they are often directly applied to complex problems where the ground truth is computationally expensive to obtain. A crucial step is often missed: **the rigorous validation of the entire ML pipeline on a canonical problem with a known analytical solution.** Our approach, using a well-established Random Forest model on the classic cantilever beam problem, is not intended to introduce a novel ML architecture. Instead, its primary contribution is to **establish a clear, verifiable, and robust validation pipeline.** By demonstrating high fidelity in this fundamental case, we build the necessary confidence to apply this methodology to more complex scenarios where analytical solutions are unavailable (like FEM/DEM calibration [11]) and where the computational savings of a fast and reliable surrogate model are most critical [2].

2 Material and Methods

The methodology adopted in this work combines the generation of high-fidelity synthetic data, based on known analytical solutions, with the training and validation of a Machine Learning model to act as a fast and accurate surrogate.

2.1 The Validation Case: Cantilever Beam

To rigorously validate the ML's ability to learn the structural response, the classic problem of the cantilever beam was selected. A prismatic beam of constant length $L = 2.0m$ was considered, with a rectangular cross-section of base $b = 0.05m$ and height $h = 0.10m$. The beam is fixed at the end $x = 0$ and free at the end $x = L$. A constant vertical concentrated load $F = 500N$ is applied at the free end ($x = 2.0m$). The coordinate system is defined with the x-axis along the beam's length (from the fixed end) and the y-axis representing the vertical deflection (positive downwards). The configuration is shown in Figure 1.

2.2 Reference Analytical Solution

Assuming linear elastic behavior and small deflections, the Euler-Bernoulli beam theory provides the analytical solution for the deflection $y(x)$ [16]. The Moment of

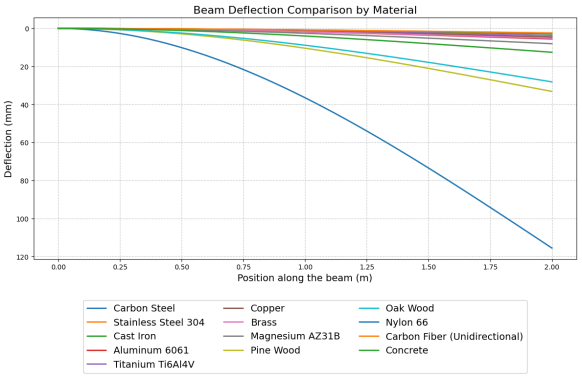

Figure 1: Schematic of the cantilever beam fixed at $x = 0$, with load $F = 500N$ applied at $x = L = 2.0m$. (Note: Figure shows deflection curves for all materials used in the study).

Inertia of the constant rectangular cross-section is $I = (bh^3)/12 = (0.05m \times (0.10m)^3)/12 \approx 4.167e-6m^4$.

The deflection $y_F(x)$ due to the concentrated load F at the tip is given by:

$$y_F(x) = \frac{Fx^2}{6EI}(3L - x) \tag{1}$$

where E is the Young's Modulus of the material.

Additionally, the deflection $y_w(x)$ caused by the beam's self-weight, uniformly distributed, was considered. The weight per unit length is $w = \rho g A$, where ρ is the material density, $g = 9.81\,\text{m/s}^2$ is the acceleration due to gravity, and $A = bh = 0.005\,\text{m}^2$ is the cross-sectional area. The deflection due to self-weight is:

$$y_w(x) = \frac{wx^2}{24EI}(6L^2 - 4Lx + x^2) \tag{2}$$

The total deflection $y(x)$ at any point x along the beam is the linear superposition of these two effects, serving as the "ground truth" for this study:

$$y(x) = y_F(x) + y_w(x) \tag{3}$$

2.3 Synthetic Database Generation

A synthetic database was programmatically generated using Python, with the NumPy and Pandas libraries, to provide training and testing data for the ML model. The process followed these steps:

1. **Fixed Parameters:** The values of L, b, h, F, and g were set as constants, as specified in Section 2.1. The values of I and A were pre-calculated.

2. **Material Selection:** A list of 13 diverse materials (detailed in Table 1) was compiled, covering

ferrous and non-ferrous metals, woods, polymers, and composites. For each material, nominal values for its intrinsic properties were assigned: Young's Modulus (E), Density (ρ), Poisson's Ratio (ν), and Yield Strength (gma_y).

3. **Spatial Discretization:** To capture the deformation curve, the beam length ($L = 2.0m$) was discretized into $N_x = 51$ equally spaced points, x_i, ranging from $x_0 = 0$ to $x_{50} = L$. This was implemented using the function `np.linspace(0, L_const, num_x_points)`.

4. **Deflection Calculation:** For each of the 13 materials, the weight per unit length w was calculated. Then, for each of the 51 points x_i, the total deflection $y(x_i)$ was calculated using (3).

5. **Data Structuring:** Each calculation produced a record containing the material name, its four properties (E, ρ, ν, gma_y), the position x_i, and the corresponding deflection $y(x_i)$ (converted to millimeters). All records were organized into a Pandas DataFrame, totaling $13 \times 51 = 663$ samples. The complete dataset, along with examples and source code, is publicly available on GitHub at: `https://github.com/zolpy/Fixed_Beam`. are presented in Table 2.

2.4 Machine Learning Model

To model the relationship between material properties, spatial coordinates, and the resulting beam deflection, the `RandomForestRegressor` algorithm [15], as implemented in the Scikit-learn library [17], was adopted. This algorithm was selected due to a combination of desirable characteristics: (i) robustness against overfitting, afforded by its ensemble nature and inherent bootstrapping; (ii) strong and consistent performance across a wide range of regression problems involving tabular data; (iii) the capacity to capture complex, non-linear relationships between input variables without requiring explicit feature transformations; and (iv) its ability to provide interpretable measures of feature importance, which are especially valuable in scientific and engineering contexts where model transparency is critical.

While other machine learning algorithmsâsuch as Support Vector Regression (SVR), Gradient Boosting Machines (GBM), or Deep Neural Networks (DNN)âcan potentially offer comparable or superior accuracy under certain conditions, Random Forests offer a compelling balance between accuracy, interpretability, and computational efficiency. These attributes make it particularly suitable for this initial validation study, where the goal is not only to achieve high predictive performance, but also to understand the behavior and limitations of the surrogate model when approximating a well-defined physical system.

The model development followed a structured pipeline, comprising the following stages:

- **Features (Input `X`):** The model was trained using five predictors: `['E_Pa', 'Density_kg_m3', 'Poisson_ratio', 'Yield_Strength_Pa', 'x_m']`, which represent the materialâs elastic modulus (E), density (ρ), Poissonâs ratio (ν), yield strength (σ_y), and spatial coordinate along the beam (x). All variables were expressed in SI base units to ensure dimensional consistency. Importantly, while ν and σ_y do not directly influence the analytical elastic deflection equations ((1), (2)), they were intentionally included to evaluate the model's ability to perform implicit feature selection. This is a relevant capability in data-driven modeling, particularly when working with high-dimensional data where domain knowledge alone may not suffice to determine feature relevance a priori.

- **Target (Output `y`):** The modelâs output was the analytically computed total beam deflection at a given position x, measured in millimeters and recorded as `'y_deflection_mm'`.

- **Data Pre-processing:** Minimal data preparation was required. No normalization or scaling of input features was applied, as Random Forests are invariant to the scale and monotonic transformations of input variables. Moreover, since the input features were already numeric and derived from the synthetic data generation process, no handling of missing values or categorical encoding was necessary. This simplicity highlights a practical advantage of tree-based methods in engineering applications.

- **Train/Test Split:** The final dataset, consisting of 663 samples, was randomly divided into a training set (80%, 530 samples) and a test set (20%, 133 samples). To ensure reproducibility and comparability, a fixed random seed (`random_state=42`) was used during the splitting process.

- **Model Training:** The `RandomForestRegressor` model was instantiated with `n_estimators=100` (number

Table 1: Nominal Properties of Materials Used in the Study.

Material	**Young's Mod.** (GPa)	**Density** (kg/m^3)	**Poisson's Ratio**	**Yield Str.** (MPa)
Ferrous Metals				
Carbon Steel	200	7850	0.30	250
Stainless Steel 304	193	7900	0.29	215
Cast Iron	170	7200	0.26	130
Non-Ferrous Metals				
Aluminum 6061	69	2700	0.33	240
Titanium Ti6Al4V	114	4430	0.34	830
Copper	117	8960	0.34	70
Brass	100	8500	0.35	120
Magnesium AZ31B	45	1770	0.35	150
Woods				
Pine Wood	10	500	0.37	30
Oak Wood	12	750	0.35	40
Polymers / Composites				
Nylon 66	3	1140	0.40	50
Uni. Carbon Fiber	150	1600	0.30	1500
Concrete	30	2400	0.20	3

Table 2: Example of the Structure and Values of the Synthetic Database (First and Last 5 Rows).

Material	**E** (Pa)	**Density** (kg/m^3)	**Poisson**	**Yield Str.** (Pa)	**x** (m)	**y Defl.** (mm)
Carbon Steel	200e9	7850	0.30	250e6	0.00	0.0000
Carbon Steel	200e9	7850	0.30	250e6	0.04	0.0017
Carbon Steel	200e9	7850	0.30	250e6	0.08	0.0067
Carbon Steel	200e9	7850	0.30	250e6	0.12	0.0149
Carbon Steel	200e9	7850	0.30	250e6	0.16	0.0262
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Concrete	30e9	2400	0.20	3e6	1.84	11.0720
Concrete	30e9	2400	0.20	3e6	1.88	11.4407
Concrete	30e9	2400	0.20	3e6	1.92	11.8101
Concrete	30e9	2400	0.20	3e6	1.96	12.1800
Concrete	30e9	2400	0.20	3e6	2.00	12.5502

of trees), `random_state=42` (for consistent results), `n_jobs=-1` (to parallelize training across all available cores), and `oob_score=True` (to compute the Out-of-Bag error as an internal cross-validation metric). The model was then fitted to the training data using the `fit()` method.

This pipeline was designed to be both effective and interpretable, ensuring traceability of results and laying the groundwork for future extensions to more complex physical systems. The full process of training the model, evaluating its predictions on a per-material basis, and visualizing the results is described in detail in Algorithm 3. The subsequent assessment of input variable relevance, based on feature importance scores extracted from the trained model, is formalized in Algorithm 4.

2.5 Evaluation Metrics

The performance of the trained model was quantitatively evaluated on the test set using standard metrics for regression problems.

The Mean Absolute Error (MAE) measures the average magnitude of the errors between predicted and actual values:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

The Mean Squared Error (MSE) calculates the aver-

Algorithm 1 Synthetic Database Generation Algorithm

procedure GENERATEBEAMDEFLECTIONDATA
▷ 1. Fixed Parameters Definition
$F \leftarrow 500.0$ ▷ Applied force (N)
$L \leftarrow 2.0$ ▷ Beam length (m)
$b \leftarrow 0.05$ ▷ Section width (m)
$h \leftarrow 0.10$ ▷ Section height (m)
$g \leftarrow 9.81$ ▷ Gravity acceleration (m/s^2)
$I \leftarrow (b \cdot h^3)/12$ ▷ Moment of inertia
$A \leftarrow b \cdot h$ ▷ Cross-sectional area
▷ 2. Materials and Spatial Discretization
`MaterialsList` ← list of dictionaries with properties `(Name, E, ρ, ν, σ_y)`
`x_points` ← 51 evenly spaced points from 0 to L ▷ Similar to `np.linspace`
▷ 3. Data Generation
`DataRecords` ← empty list
for each `material` in `MaterialsList` **do**
Extract E, ρ, ν, σ_y from `material`
$w \leftarrow \rho \cdot g \cdot A$ ▷ Distributed load (N/m)
if $E > 0$ **and** $I > 0$ **then**
for each x in `x_points` **do**
$y_F \leftarrow \dfrac{F \cdot x^2 \cdot (3L - x)}{6 \cdot E \cdot I}$
$y_w \leftarrow \dfrac{w \cdot x^2 \cdot (x^2 + 6L^2 - 4Lx)}{24 \cdot E \cdot I}$
$y_{\text{total\_m}} \leftarrow y_F + y_w$ ▷ Total deflection (m)
$y_{\text{total\_mm}} \leftarrow y_{\text{total\_m}} \cdot 1000$ ▷ Converted to mm
`record` ← {Name, E, ρ, ν, σ_y, x, $y_{\text{total\_mm}}$}
Append `record` to `DataRecords`
end for
end if
end for
▷ 4. Data Structuring and Saving
`DataFrame` ← create data table from `DataRecords`
Save `DataFrame` to CSV file
end procedure

Algorithm 2 Algorithm for Plotting Comparative Deflection Curves

procedure PLOTALLDEFLECTIONCURVES(`DataTable`)
▷ Input: Data table with columns [Material, x_m, y_deflection_mm]
▷ 1. Initialize the plotting environment
Create a new plot figure
▷ 2. Identify unique materials
`UniqueMaterials` ← list of unique material names from `DataTable`
▷ 3. Plot deflection curve for each material
for each `material_name` in `UniqueMaterials` **do**
`MaterialData` ← filter `DataTable` for rows with `material_name`
`x_positions` ← extract `x_m` column from `MaterialData`
`y_deflections` ← extract `y_deflection_mm` column from `MaterialData`
Plot `y_deflections` vs `x_positions` with label `material_name`
end for
▷ 4. Configure plot appearance
Invert Y-axis direction
Set plot title: `"Beam Deflection Comparison by Material"`
Set X-axis label: `"Position along the beam (m)"`
Set Y-axis label: `"Deflection (mm)"`
Add grid lines
Show plot legend
▷ 5. Save the plot
Save plot as high-resolution image (e.g., PNG)
end procedure

Algorithm 3 Algorithm for Model Training and Per-Material Performance Visualization

procedure TRAINANDVISUALIZEMODEL(`dataFile_path`)
▷ 1. Load and preprocess the dataset
`DataTable` ← load CSV data from `dataFile_path`
Translate material names in `DataTable` to English
▷ 2. Define features and target variable
`Features_X` ← columns {E_Pa, Density, Poisson_ratio, Yield_Strength, x_m}
`Target_y` ← column `y_deflection_mm`
▷ 3. Split data into training and testing sets
Split (`Features_X`, `Target_y`) into (`X_train`, `y_train`) and (`X_test`, `y_test`)
▷ 4. Initialize and train the machine learning model
`ML_Model` ← RandomForestRegressor (n_estimators=100, oob_score=True)
Train `ML_Model` on `X_train`, `y_train`
▷ 5. Evaluate overall model performance
`y_pred_test` ← predict using `ML_Model` on `X_test`
Compute overall MAE, R^2, and OOB score
Display overall metrics
▷ 6. Generate per-material performance plots
`UniqueMaterials` ← unique material names in `DataTable`
for each `material_name` in `UniqueMaterials` **do**
`MaterialData` ← filter `DataTable` for `material_name`
`X_material` ← extract features from `MaterialData`
`y_actual` ← extract target from `MaterialData`
`y_predicted` ← predict using `ML_Model` on `X_material`
Initialize a new plot figure
Plot `y_actual` vs. x-position (solid blue line with circle markers)
Plot `y_predicted` vs. x-position (dashed red line with 'x' markers)
Compute R^2 score for this material
Annotate plot with material-specific R^2 value
Set title, axis labels, grid, and invert Y-axis
Save or display the plot
end for
end procedure

Algorithm 4 Algorithm for Feature Importance Analysis and Visualization

procedure ANALYZEANDPLOTFEATUREIMPORTANCE(`dataFile_path`)
▷ 1. Load and prepare data
`DataTable` ← load CSV data from `dataFile_path`
`Features_X` ← select input feature columns from `DataTable`
`Target_y` ← select target column from `DataTable`
Split (`Features_X`, `Target_y`) into (`X_train`, `y_train`) and (`X_test`, `y_test`)
▷ 2. Train model and extract feature importances
`ML_Model` ← RandomForestRegressor()
Train `ML_Model` on `X_train`, `y_train`
`ImportanceScores` ← extract feature importances from `ML_Model`
`FeatureNames` ← get feature names corresponding to `ImportanceScores`
▷ 3. Prepare data for plotting
`ImportanceTable` ← map each feature name to its importance score
Sort `ImportanceTable` in descending order by importance score
▷ 4. Generate bar plot
Create horizontal bar plot using `ImportanceTable`
(features on Y-axis, importance scores on X-axis)
for each bar in the plot **do**
Add numerical label showing importance score next to the bar
end for
▷ 5. Configure and save the plot
Invert Y-axis to show most important feature at the top
Set plot title: `"Feature Importance Analysis"`
Set X-axis label: `"Relative Importance (Normalized)"`
Set Y-axis label: `"Input Feature"`
Add X-axis grid lines
Save plot as high-resolution image file
end procedure

age of the squares of the errors:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

The Root Mean Squared Error (RMSE) is the square root of the MSE, providing an error metric in the same units as the target variable:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6}$$

The Coefficient of Determination (R^2) represents the proportion of the variance in the dependent variable that is predictable from the independent variables:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{7}$$

In these equations, y_i are the actual values (from the test set), $\hat{y}_i$ are the values predicted by the model, $\bar{y}$ is the mean of the actual values, and n is the number of samples in the test set.

Additionally, the Out-of-Bag (OOB) R^2 score, calculated internally during the Random Forest training using data not seen by each individual tree, was recorded as an estimate of the model's generalization capability.

3 Implementation and Case Study

The described methodology was implemented using the Python programming language (version 3.13 used in original study, check compatibility if using different version). Database generation and manipulation were performed with the NumPy [18] library for numerical operations and Pandas [19] for DataFrame manipulation. Machine Learning model training and evaluation were conducted using the Scikit-learn library (version 1.6.1 used in original study) [17]. Visualization of results was done using the Plotly library [20].

The case study considered a cantilever beam with the fixed geometric and loading parameters defined in Section 2.1: $L = 2.0\,\text{m}$, $b = 0.05\,\text{m}$, $h = 0.10\,\text{m}$, and $F = 500\,\text{N}$. The acceleration due to gravity was $g = 9.81\,\text{m/s}^2$.

Thirteen distinct material types were included in the database, with their nominal properties listed in Table 1. For each material, the deflection was calculated at $N_x = 51$ points along the beam, resulting in 663 total data points.

The Random Forest Regressor model was configured with 100 trees (`n_estimators=100`) and other parameters as specified in Section 2.4 (`random_state=42`, `n_jobs=-1`, `oob_score=True`). Training was performed on the set of 530 samples, and final evaluation on the test set of 133 samples.

4 Results and discussion

4.1 Performance Metrics and Numerical Validation

The quantitative evaluation of the trained Random Forest model was performed on the test set (n=133), comparing the ML-predicted deflections ($\hat{y}_i$) with the analytically calculated values (y_i). The resulting performance metrics are summarized in Table 3.

The results demonstrate exceptional performance of the surrogate model. The Coefficient of Determination (R^2) of 0.9991 indicates that the model can explain over 99.9% of the variance present in the test set deflection data, suggesting an almost perfect fit. The average errors are extremely low, with an MAE of $0.2105mm$and an RMSE of $0.460mm$, confirming the high accuracy of the point-by-point predictions relative to the analytical reference solution. Additionally, the Out-of-Bag (OOB) R^2 score, estimated during training on data not seen by each individual tree, was 0.9983. The closeness between the test R^2 and the OOB R^2 reinforces the model's excellent generalization capability and indicates the absence of significant overfitting.

4.2 Visual Analysis of Predicted Curves

To complement the quantitative analysis, the model's ability to reproduce the spatial shape of the deflection curve was visually assessed. Figures 2 to 14 present the graphical comparison between the analytically calculated deflection curves (considered the ground truth in this study) and the curves predicted by the Random Forest model for each of the 13 materials.

An almost perfect visual agreement is observed in Figures 2 to 14 between the ML model's predictions and the analytical results for all materials and across the entire beam length ($0 \leq x \leq L$). The model accurately captures both the magnitude and the characteristic shape of the cantilever beam deflection curve, even for materials with orders of magnitude differences in their properties and resulting deflections. This demonstrates that the model not only predicts point values with low error but has also learned the functional representation of deflection along the spatial coordinate x.

4.3 Feature Importance Analysis

The Random Forest algorithm allows estimating the relative importance of each input feature in making predic-

Table 3: Performance Metrics of the Random Forest Model on the Test Set (n=133).

Metric	Acronym	Value
Mean Absolute Error	MAE	0.2105 mm
Mean Squared Error	MSE	0.2121 mm^2
Root Mean Squared Error	RMSE	0.4605 mm
Coefficient of Determination	R^2	0.9991
Out-of-Bag R^2 Score (Estimated)	OOB R^2	0.9983

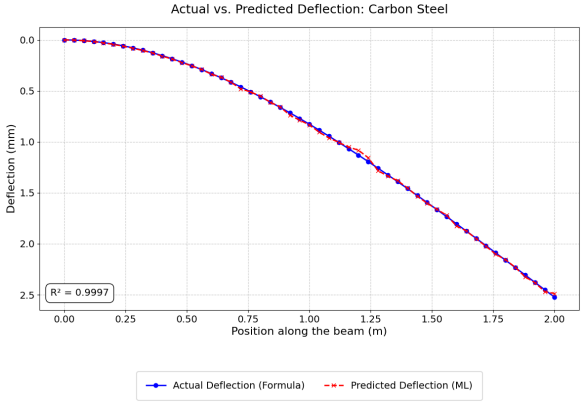

Figure 2: Comparison of analytical vs. predicted (ML) deflection for Carbon Steel.

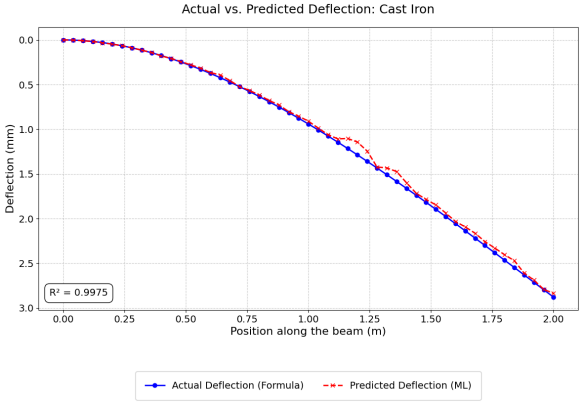

Figure 4: Comparison of analytical vs. predicted (ML) deflection for Cast Iron.

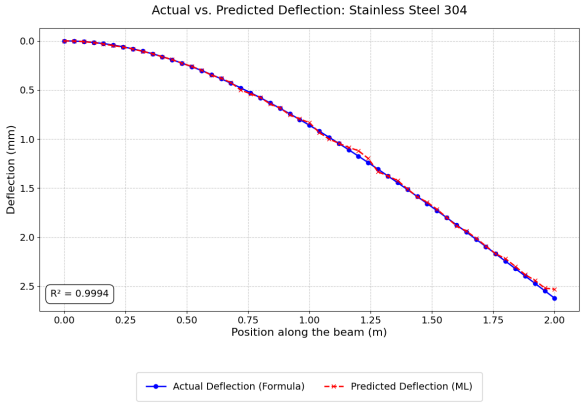

Figure 3: Comparison of analytical vs. predicted (ML) deflection for Stainless Steel 304.

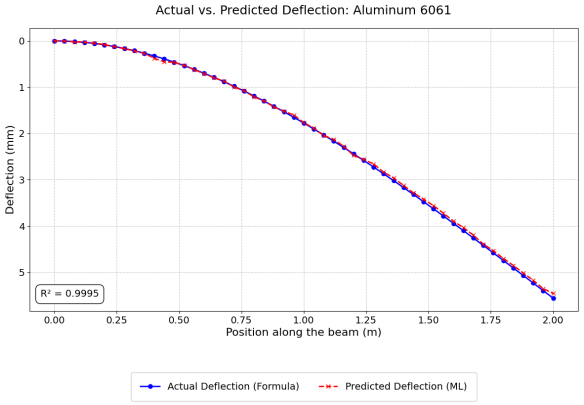

Figure 5: Comparison of analytical vs. predicted (ML) deflection for Aluminum 6061.

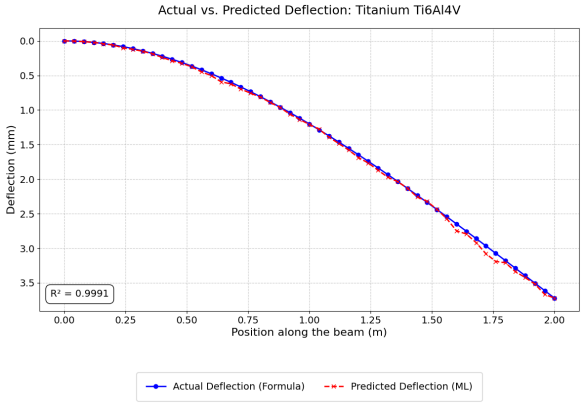

Figure 6: Comparison of analytical vs. predicted (ML) deflection for Titanium Ti6Al4V.

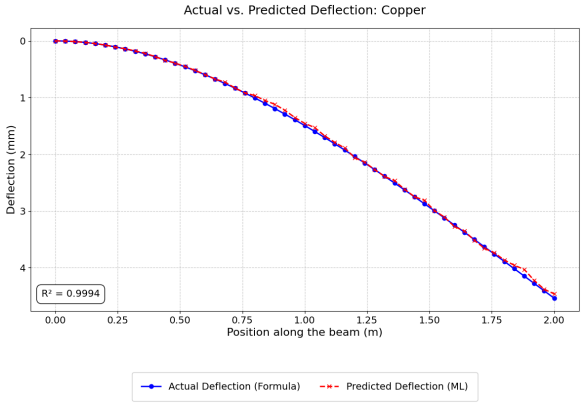

Figure 7: Comparison of analytical vs. predicted (ML) deflection for Copper.

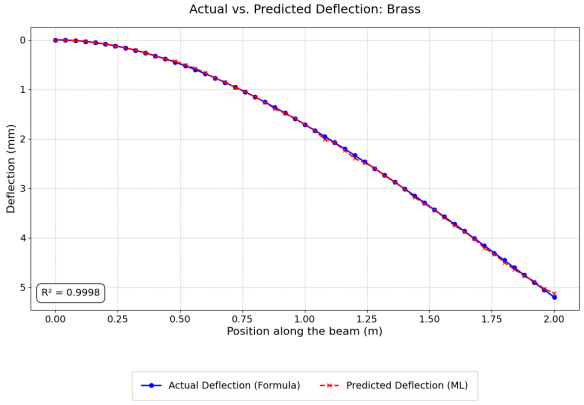

Figure 8: Comparison of analytical vs. predicted (ML) deflection for Brass.

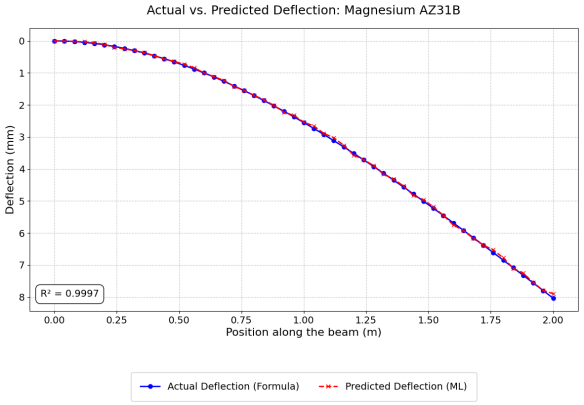

Figure 9: Comparison of analytical vs. predicted (ML) deflection for Magnesium AZ31B.

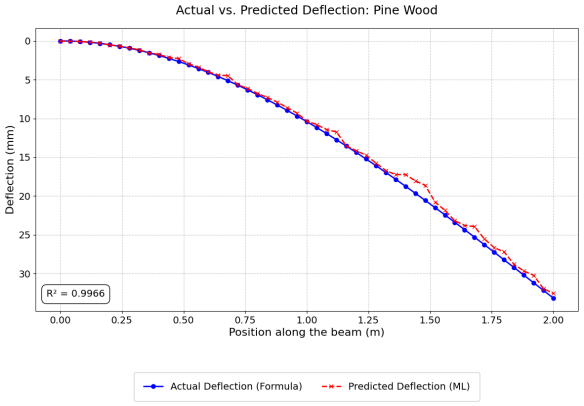

Figure 10: Comparison of analytical vs. predicted (ML) deflection for Pine Wood.

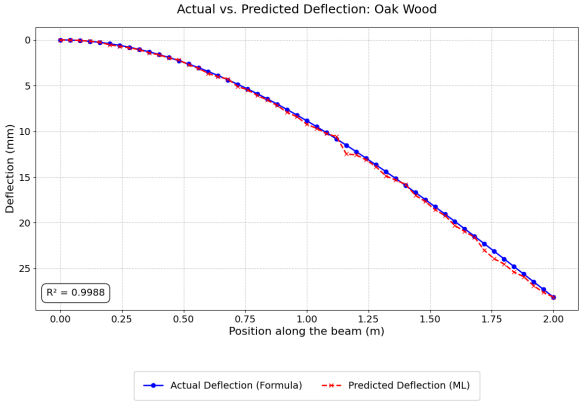

Figure 11: Comparison of analytical vs. predicted (ML) deflection for Oak Wood.

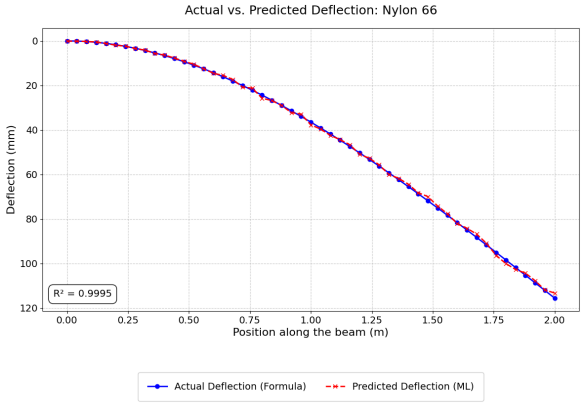

Figure 12: Comparison of analytical vs. predicted (ML) deflection for Nylon 66.

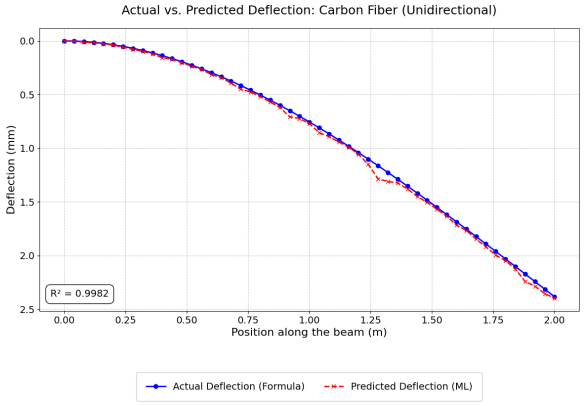

Figure 13: Comparison of analytical vs. predicted (ML) deflection for Unidirectional Carbon Fiber.

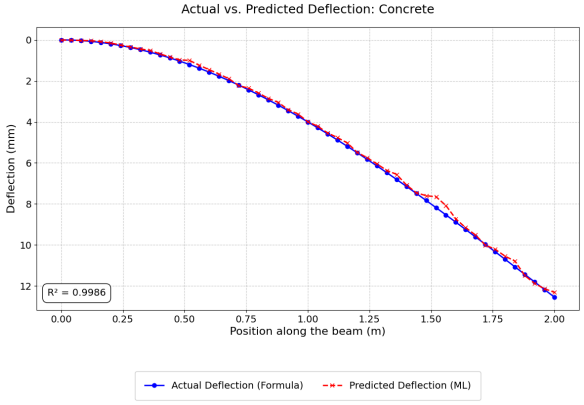

Figure 14: Comparison of analytical vs. predicted (ML) deflection for Concrete.

tions. This metric quantifies the contribution of each variable to reducing impurity (or variance, in regression) at the nodes of the trees composing the forest. Figure 15 presents the calculated importance (normalized, where the total sum is 1.0) for the five features used in this study.

The analysis of Figure 15 reveals that the position along the beam ('x_m') was identified by the model as the most influential feature, with a relative importance of approximately 0.49. This is in complete agreement with the physics of the problem, as the deflection in a cantilever beam exhibits strong spatial dependence, varying from zero at the fixed end to its maximum value at the free end, as described by the polynomial components in x in (1) and (2).

Poisson's ratio ('Poisson_ratio') emerged as the second most important feature (approximately 0.33), followed by Young's Modulus ('E_Pa') with about 0.15 importance. Although Young's Modulus is fundamental in Euler-Bernoulli theory (appearing in the denominator of the deflection equations), the high importance attributed to Poisson's ratio by the specific model trained in this study is an unexpected result, since ν does not explicitly appear in the simplified equations used to generate the training data. This finding might indicate that the Random Forest model utilized ν, which varies among materials, in conjunction with other features to make splits at the tree nodes efficiently for this specific dataset, or it might reflect the sensitivity of feature importance calculation to correlations in the input data.

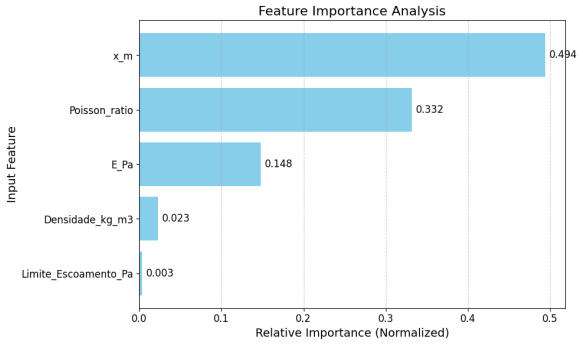

Figure 15: Relative importance of input features calculated by the Random Forest model for predicting beam deflection.

Density showed a lower relative importance (approximately 0.02), despite directly influencing the self-weight w and, therefore, the $y_w(x)$ component of the deflection. This suggests that, for the studied load and geometry configuration, the deflection due to the concentrated load F was dominant over the deflection due to self-weight, reducing the relative impact of density

on the prediction of total deflection.

Finally, as expected, the Yield Strength ('Yield_Strength_Pa') demonstrated almost negligible importance (approximately 0.003). This result is consistent with the fact that the analysis and generated data were based on linear elasticity theory, where deflection is not influenced by the material's yield limit.

Although the importance ranking between Poisson's ratio and Young's Modulus was unexpected, the model correctly attributed the highest relevance to position x and Young's Modulus E, and correctly identified the low influence of Yield Strength for predicting elastic deflection, demonstrating a general alignment with the fundamental physical understanding of the problem.

4.4 General Discussion and Contextualization

The rigorous validation of the ML model against the known analytical solution, evidenced by the excellent quantitative metrics (Table 3) and the visual agreement of the deflection curves (Figures 2-14), demonstrates the capability of the proposed pipeline to create a high-fidelity surrogate model for this structural problem.

A crucial point is the *computational efficiency* of the approach. While generating the 663 data points via analytical calculation and training the Random Forest model took seconds, the *prediction* phase with the trained model is practically instantaneous (milliseconds for a complete curve). This speed starkly contrasts with the time required by complex numerical simulations (FEM/DEM) [11], making surrogate models extremely valuable tools for analyses requiring multiple evaluations, such as optimization, parametric studies, or uncertainty quantification.

It is relevant to *contextualize* this computational approach with experimental methods for measuring beam deflection. Works like those by Dias et al. [1] (robotic arm), Picoy et al. [6] (DIC), and Braga Jr et al. [7] (PIV/Speckle) demonstrate sophisticated techniques for obtaining experimental data. These studies often report good agreement with simulations or theory but also observe deviations (e.g., $\approx 0.3mm$ in [6]) attributable to experimental uncertainties and idealizations in the comparison models. This ML model, by replicating the analytical solution with MAE $\approx 0.21mm$, shows consistency with idealized theory but does not replace experimental validation. It acts, rather, as a complementary predictive tool, whose high speed and theoretical consistency can aid in planning and interpreting experiments or rapidly exploring virtual scenarios. Future cross-validation, comparing predictions from ML models trained on FEM simulations calibrated with experimental data like those cited, would be an important step.

Finally, the confidence established in the methodology through this validation in a canonical case paves the way for its *application to more complex problems*. The same ML pipeline can be adapted by replacing the analytical data source with results from FEM/DEM simulations (which also often provide results at discrete points, like mesh nodes). It is expected that a similar ML model can learn to map complex input parameters (DEM calibration parameters [11, 14], geometric details, non-linear boundary conditions) to the observed response curves (stress-strain curves, velocity profiles, etc.), significantly accelerating the analysis and design cycle in computational engineering.

5 Conclusion

This work presented the design, implementation, and validation of a machine learning-based methodology for predicting the deflection behavior of a cantilever beam subjected to a concentrated tip load and self-weight. A Random Forest Regressor was employed to construct a surrogate model capable of accurately reproducing the complete deflection curves based solely on input features such as material properties and beam position. The main goal was to rigorously assess the modelâs ability to capture a well-established physical phenomenonâgoverned by analytical expressionsâbefore extending this approach to more complex scenarios typically analyzed using Finite Element Method (FEM) or Discrete Element Method (DEM) simulations [11].

The trained model demonstrated exceptional predictive performance on the test dataset, achieving a coefficient of determination (R^2) above 0.999 and mean errors (MAE and RMSE) on the order of tenths of a millimeter. Beyond quantitative metrics, the graphical comparison between predicted and analytical curves showed that the model accurately replicated both the magnitude and spatial profile of the beam's deformation for all tested materials. These results underscore the model's ability not only to interpolate between known cases, but also to generalize its predictions across varying material behaviors.

The analysis of feature importance further supported the physical plausibility of the surrogate model. Although the relative rankings of some variablesâsuch as Poisson's ratio and Young's modulusâdiffered slightly from expectations, the model correctly identified position along the beam and Youngâs modulus as dominant predictors of deflection. Yield strength, a parameter not involved in elastic deformation, was appropriately ranked as minimally influential. These findings reinforce the interpretability of Random Forest and

its capacity to reflect underlying physics when properly trained.

A notable advantage of the proposed approach is its computational efficiency. Once trained, the surrogate model generates full deflection curves within milliseconds, offering significant acceleration compared to traditional simulation-based workflows. This is particularly relevant for applications involving iterative design, optimization, or real-time control systems, where repeated evaluations are computationally costly.

The contributions of this study can be summarized as follows:

(i) A clear demonstration that a Random Forest model can learn and generalize the functional response (deflection curve) of a canonical structural problem from tabular input data;

(ii) A fully validated modeling pipeline, tested against analytical ground truth, which establishes a baseline for future applications to more complex and nonlinear engineering systems;

(iii) A structured and replicable frameworkâfrom synthetic data generation to model training, evaluation, and predictionâthat is adaptable to surrogate modeling of numerical simulations in engineering.

Nevertheless, the scope of this work was intentionally restricted to a linear-elastic, single-load, and geometrically simple system. As such, several limitations are acknowledged. First, the modelâs performance is intrinsically linked to the diversity and representativeness of the training dataset. Generalization to geometries or loading conditions not present in the training data is not guaranteed. Second, while Random Forest was effective here, no systematic comparison was conducted against alternative machine learning algorithms, leaving room for further performance benchmarking.

Future research will extend this methodology to more realistic and computationally intensive problems. These include systems involving geometric and material nonlinearities, multiple concurrent loads, and more intricate boundary conditions. Data for such cases will be generated using high-fidelity FEM and DEM simulations. In parallel, benchmarking against other regression algorithmsâsuch as Support Vector Regression, Gradient Boosting, and neural networksâwill be performed to assess relative strengths and weaknesses. Additionally, the integration of uncertainty quantification techniques and the exploration of data-driven approaches tailored to sequential or spatially correlated outputs will be pursued.

In summary, this study establishes a robust and validated foundation for using machine learning techniques to accelerate the modeling and prediction of structural behavior in engineering. By demonstrating that data-driven models can capture the essence of a well-understood physical system, this work paves the way for applying such techniques to complex, high-dimensional, and computationally demanding problems in structural mechanics and beyond.

References

[1] G. L. Dias, R. R. MagalhÃ£es, F. A. Vitoriano, and D. D. Ferreira, “The use of a robotic arm for displacement measurements in a cantilever beam,” *Int. J. Manuf. Mater. Mech. Eng.*, vol. 6, no. 4, pp. 45–57, 2016, doi: 10.4018/ijmmme.2016100104.

[2] S. Mishra, A. Kumar, and R. Singh, “Transforming Civil Engineering with AI and Machine Learning: Innovations, Applications, and Future Directions,” *Journal of Innovations in Engineering and Science*, vol. 8, no. 2, pp. 1-15, 2025.

[3] K. Mo, D. L. J. M. van der Heijden, J. C. Vergeest, and A. Khademi, “Visual design intuition: predicting dynamic properties of beams from raw cross-section images,” *J. R. Soc. Interface*, vol. 18, no. 184, p. 20210493, 2021.

[4] T. Sahin, D. Wolff, M. von Danwitz, and A. Popp, “Towards a Hybrid Digital Twin: Physics-Informed Neural Networks as Surrogate Model of a Reinforced Concrete Beam,” in *2024 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, Bonn, Germany, 2024, pp. 1-8.

[5] Y. Sun, B. Li, and X. Sun, “Deep Learning in Earthquake Engineering: A Comprehensive Review,” *arXiv preprint arXiv:2405.08882*, 2024.

[6] Y. S. M. Picoy, R. R. MagalhÃ£es, E. T. Andrade, M. C. Campos, and G. H. Costa, “Digital image correlation analysis for displacement measurements in cantilever beams,” *Theor. Appl. Eng.*, vol. 1, no. 1, pp. 1–7, 2017. [Online]. Available: `http://www.aeletters.com/pdf/ae-2017-v01-n01-a01.pdf`

[7] R. A. Braga Jr, R. R. MagalhÃ£es, R. P. Melo, and J. V. S. Gomes, “Maps of deformations in a cantilever beam using particle image velocimetry (PIV) and speckle patterns,” *Rem: Rev. Esc. Minas*, vol. 68, no. 3, pp. 273–278, 2015, doi: 10.1590/0370-44672014680059.

[8] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 6th ed. Amsterdam, Netherlands: Elsevier, 2005.

[9] P. A. Cundall and O. D. L. Strack, "A discrete numerical model for granular assemblies," *Géotechnique*, vol. 29, no. 1, pp. 47–65, 1979, doi: 10.1680/geot.1979.29.1.47.

[10] C. J. Coetzee, "Review: Calibration of the discrete element method," *Powder Technol.*, vol. 310, pp. 104–142, 2017, doi: 10.1016/j.powtec.2016.12.065.

[11] J. IrazÃ¡bal, F. Salazar, and D. J. Vicente, "A methodology for calibrating parameters in discrete element models based on machine learning surrogates," *Comput. Particle Mech.*, vol. 10, no. 6, pp. 1031–1047, 2023, doi: 10.1007/s40571-022-00534-8.

[12] A. I. Forrester, A. Sobester, and A. J. Keane, *Engineering design via surrogate modelling: a practical guide*. Chichester, UK: John Wiley & Sons, 2008, doi: 10.1002/9780470770801.

[13] L. Benvenuti, C. Kloss, and S. Pirker, "Identification of DEM simulation parameters by artificial neural networks and bulk experiments," *Powder Technol.*, vol. 291, pp. 456–465, 2016, doi: 10.1016/j.powtec.2016.01.015.

[14] H. Cheng, T. Shuku, K. Thoeni, R. Tempone, S. Luding, and V. Magnanimo, "An iterative Bayesian filtering framework for fast and automated calibration of DEM models," *Comput. Methods Appl. Mech. Eng.*, vol. 350, pp. 268–294, 2019, doi: 10.1016/j.cma.2019.03.004.

[15] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[16] J. M. Gere and B. J. Goodno, *Mechanics of Materials*, 7th ed. Boston, MA, USA: Cengage Learning, 2009.

[17] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: `http://jmlr.org/papers/v12/pedregosa11a.html`

[18] C. R. Harris *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[19] W. McKinney, "Data structures for statistical computing in python," in *Proc. 9th Python Sci. Conf.*, 2010, pp. 51–56, doi: 10.25080/Majora-92bf1922-00a.

[20] Plotly Technologies Inc., "Collaborative data science," Montréal, QC, 2015. [Online]. Available: `https://plot.ly`