Optimization of Structural Analysis through Reduced Order Models and Non-Intrusive Machine Learning

Author: Matías Zieleniec

Academic Supervisor: Franz Chouly

Industrial Supervisor: Francisco Pons

Affiliation: Ingenium - Ingeniería Estructural

Academic Institution: Centro de Matemática, FCIEN-UdelaR

Date: June 16, 2025

Optimization of Structural Analysis through Reduced Order Models and Non-Intrusive Machine Learning

Abstract

This research focuses on the temporal optimization of structural analysis using the finite element method (FEM). The problem of high computational cost in high-fidelity simulations, particularly in parametric analyses, is addressed by employing non-intrusive Reduced Order Methods (ROMs). These methods are advantageous as they do not require access to the governing differential equations, but only to the solutions (snapshots) generated by commercial software. Two machine learning-based approaches are proposed and evaluated. The first method uses a neural network to predict the coefficients of a reduced basis, obtained through Proper Orthogonal Decomposition (POD), which approximates the system's solution. The second approach focuses on predicting the inverse of the reduced stiffness matrix using tree-based ensemble methods (Random Forest and Gradient Boosting). Results are presented for two case studies: an elastic block and a Kirchhoff-Love plate, evaluating the effectiveness and accuracy of each methodology. The findings indicate that both approaches are promising, although their accuracy varies depending on the complexity of the structural problem and the machine learning method used

Keywords: Finite Element Method (FEM), Reduced Order Models (ROM), Reduced Basis Method, Machine Learning, Non-Intrusive, Proper Orthogonal Decomposition (POD), Neural Networks, Gradient Boosting, Random Forest, Structural Engineering.

1. Introduction

This project was developed within the framework of an internship at Ingenium - Ingeniería Estructural for the Bachelor 's degree in Mathematics at FCIEN-UdelaR.

The finite element method (FEM), whose theoretical foundations were laid in the mid-20th century and popularized with the advent of digital computers, is now a fundamental numerical tool for simulating the behavior of structures in engineering. Despite its widespread use and maturity, detailed FEM models can be computationally prohibitive, particularly in applications requiring multiple evaluations, such as parametric analyses or optimization processes. To mitigate this cost, Reduced Order Methods (ROMs) have emerged. These techniques, which aim to drastically reduce the dimensionality of the original problem while maintaining acceptable accuracy, are often based on constructing a reduced basis from pre-computed solutions. A cornerstone for this purpose is Proper Orthogonal Decomposition (POD), a method with roots in principal component analysis that was introduced in the field of fluid mechanics by Lumley in 1967 to identify coherent structures in turbulent flows, and has since been adapted to a wide range of physical problems.

Initially, the application of these methods focused on intrusive approaches, which require explicit knowledge of the differential equations and their weak formulations. While these methods work for simple geometries, their applicability is limited in scenarios with a combination of different structural elements or complex boundary conditions. Given these limitations, the research focus has shifted in the last decade towards non-intrusive methods, a change largely driven by advances in Machine Learning. These methods build the reduced model directly from data generated by the high-fidelity model (FEM), without needing to access or modify the underlying equations of the system. This is particularly relevant since in most commercial simulation software, the user only has access to the outputs (solutions or snapshots) and not to the internal matrices of the system.

In this work, two non-intrusive methodologies are proposed and compared:

1. An approach that uses Proper Orthogonal Decomposition (POD) to construct a reduced basis from snapshots and trains a neural network to map the input parameters of the problem to the coefficients of said basis. The theoretical foundation for this mapping is supported by the Universal Approximation Theorem (Cybenko, 1989, 1).
2. A second approach that seeks to directly predict the inverse of the reduced stiffness matrix. For this, POD is applied to a set of these inverse matrices, and a tree-based ensemble method is trained to predict their coefficients from the problem parameters.

---

2. Methodology

The procedure is divided into a computationally intensive offline stage and a rapid online evaluation stage. The two proposed methods are detailed below.

2.1 Method 1: Solution Approximation with Neural Networks

2.1.1 Offline Stage

1. **Choose the parameter space:** The first step is to define which parameters will be varied and in what ranges (the smaller the range, the better the approximation will be, and the more parameters there are, the larger the training set will need to be due to the curse of dimensionality).
2. **Snapshot Generation:** A finite element software (FEniCS in this research) is used to calculate a set of high-fidelity solutions (snapshots) for different parametric configurations of the structural problem. These snapshots form the training and testing set.
3. **Reduced Basis Construction (POD):** Proper Orthogonal Decomposition (POD) is applied to the set of training snapshots using the UQpy library. POD allows for the identification of the most energetically significant modes, which will form the reduced basis. The choice of the number of modes in the basis is crucial, as a larger number of modes can make the approximation by the neural network more difficult. The singular values of the snapshot matrix can be useful for this choice.
4. **Reference Coefficient Calculation:** With the reduced basis obtained, the optimal coefficients (expected outputs for the network) are calculated for each snapshot in the training set through a least-squares projection. This projection ensures the best possible approximation of the high-fidelity solutions in the subspace defined by the reduced basis.
5. **Neural Network Training:** A neural network is trained to learn the function that maps the input parameters of the problem to the reduced basis coefficients calculated in the previous step. The goal is for the network to be able to predict these coefficients for new combinations of parameters. Different network architectures are explored, varying the number of hidden layers, neurons per layer, and activation functions

2.1.2 Online Stage

In the online stage, given a new set of input parameters for which the solution is desired, the trained neural network is used to predict the coefficients of the reduced basis. The approximate solution is then reconstructed as a linear combination of the reduced basis vectors using the predicted coefficients. This stage is computationally very efficient, as it only involves the evaluation of the trained neural network.

2.2 Method 2: Prediction of the Inverse Stiffness Matrix with Tree-Based Ensemble Methods

2.2.1 Offline Stage

1. **Choose the parameter space:** The first step is to define which parameters will be varied and in what ranges (the smaller the range, the better the approximation will be, and the more parameters there are, the larger the training set will need to be due to the curse of dimensionality).
2. **Data Generation:** Solution snapshots are generated, and the stiffness matrices A and load vectors f are extracted for each parametric configuration. These depend on the chosen parameter.
3. **Reduced Basis Construction (POD):** POD is applied to the snapshots to obtain the reduced basis matrix V.
4. **Reduction of the stiffness matrices:** The stiffness matrices are reduced with the reduced basis matrix. If V is the reduced basis matrix with n modes and A is the stiffness matrix, then $V^T AV$ is the reduced stiffness matrix of size $n \times n$
5. **POD on the inverse of the reduced stiffness matrix:** The reduced stiffness matrix is inverted and vectorized (flattened), storing them in a matrix. POD is applied to this matrix to obtain a reduced basis to express these matrices.
6. **Training of tree-based ensemble methods:** A tree-based ensemble model (we work with Random Forest and Gradient Boosting) is trained to predict the relationship between a parameter and the multipliers of the reduced basis in order to approximate the inverse of the reduced stiffness matrix.

2.2.2 Online Stage

In this stage, the idea is to find the displacement field for a new parameter. The tree-based ensemble method predicts the coefficients to reconstruct the inverse reduced stiffness matrix (A_r^{-1}). Then we can solve the reduced system $u_r = A_r^{-1} f_r$. Where f_r is the reduced load vector, meaning $f_r = V^T f$ (this is the projection onto the reduced space). This reduced vector can be obtained for a new parameter or a model can be trained to predict it (in general, the load vectors vary linearly). Finally, it is multiplied by the reduced basis matrix to bring it to the original size (project to the original space).

---

3. Case Studies and Software

All codes were developed in Python. The main libraries used include:

- **FEniCS** (2019 version, not FEniCSx) for finite element simulation and snapshot generation.[1]
- **UQpy** for the implementation of POD.
- **Pytorch** for the development and training of neural networks.
- **Scikit-learn (Sklearn)** to test other regression models.
- **LightGBM** to train the gradient boosting.

Two main structures were investigated:

1. **Elastic Block:** A square block with a side of 1 meter and around 2000 nodes, fixed on its left side, divided into 9 subdomains (quadrants). Traction or compression forces are applied to three sections of the right side (see Appendix A).
2. **Kirchhoff-Love Plate:** A square plate with a side of 5 meters with fixed edges, divided into 4 quadrants. A distributed force (upward or downward) is applied in each quadrant, and the thickness of the plate is varied (see Appendix B). For the discretization, we divided each side into 17 equal parts, generating 578 finite elements (triangles).

Initial tests included the weak formulation of the Kirchhoff-Love plate problem and its solution using FEM in FEniCS, as well as initial tests with the Euler-Bernoulli beam in FEniCS and RBniCS. RBniCS was also used for an example of an elastic block with variable stiffness and for the Kirchhoff-Love plate with reduced bases. However, due to the complexity of working with the differential equations, the non-intrusive alternative based on POD and neural networks was chosen.

---

4. Results

We analyze the results for both methodologies.

4.1 Results of Method 1 (Neural Networks)

The effectiveness of the proposed non-intrusive method was evaluated in the two case studies.

4.1.1 Elastic Block

[1] For Windows, the installation of this software can be complex. It is recommended to use WSL with Linux.

For the elastic block case study, a dataset with 1000 samples for training and 30 for testing was used. From the training snapshots, a reduced basis of 10 elements was constructed using POD. The prediction model consisted of a neural network with 3 hidden layers, each with 48 neurons. The accuracy of the method was evaluated using two error indicators:

1. **Relative error:** Measures the overall discrepancy between the solution of the reduced model and that of the high-fidelity model, calculated as the norm of the difference between the two, normalized by the norm of the real solution. This error quantifies how far off the predicted solution is as a whole.

2. **Relative error to the maximum displacement:** Focuses on the maximum local error. It is calculated as the largest absolute difference between the predicted and the real solution at any degree of freedom, normalized by the absolute maximum value of the real displacement. This indicator allows for the identification of the magnitude of the worst-point prediction error.

$$e_{relative} = \frac{||u_{FEM}(\mu) - u_{ROM}(\mu)||}{||u_{FEM}(\mu)||} \qquad e_{rel\,max} = \frac{max_i(|u_{FEM}(\mu)[i] - u_{ROM}(\mu)[i]|)}{max_i(|u_{FEM}(\mu)[i]|)}$$

The results in terms of relative error and relative error to the maximum displacement are shown in Table 1.

Table 1: Results of the neural network model for the elastic block.

	Relative Error		Relative Error to Maximum Displacement	
	Training	Testing	Training	Testing
Average	0.49%	1.37%	0.55%	1.57%
Variance	3.34e-5	5.23e-5	2.98e-5	5.86e-5
Maximum	4.14%	3.34%	8.90%	3.16%

This approximation is considered very good with both error evaluation methods.

Additionally, polynomial regression was tested. A third-degree polynomial generated the relative errors shown in Table 2.

Table 2: Results of polynomial regression (degree 3) for the elastic block. Only with the usual relative error.

	Training	**Testing**
Average relative error	0.5451%	0.48683%
Maximum relative error	1.8261%	1.08789%

Polynomial regression also yields a very good approximation for this case.

4.1.2 Plate

For the plate case, 4000 training data and 40 testing data were used. The reduced basis consisted of 4 elements and the neural network had 3 layers and 150 neurons per layer. The results are presented in Table 3

Table 3: Results of the neural network model for the plate.

	Relative Error		Relative Error to Maximum Displacement	
	Training	Testing	Training	Testing
Average	13.23%	26.39%	14.31%	27.91%
Variance	0.0259	0.0746	0.0283	0.0859
Maximum	122.43%	133.16%	124.70%	141.55%

If an expected error of the order of 5% is considered, as this can be the difference between two different meshes in the finite element method, according to the analysis, the approximation for the plate was not as good as expected. Other alternatives to neural networks such as random forest and other linear models were also tested, but these did not work.

4.2 Results of Method 2 (Tree-based ensemble methods)

For this method, the elastic block was tested in 2 different situations. The first is like the previous problem, and the second is the elastic block divided into 9 subdomains where the stiffness of each subdomain is varied individually. This means that in total we have a parameter space of dimension 12 (3 forces and 9 stiffnesses). This method cannot be tested with the plate because to formulate the plate problem, the mixed formulation had to be used, modifying the stiffness matrix. Also, the value of f was not predicted; it was extracted with the snapshots.

4.2.1 Results for the elastic block (4 parameters)

Using **Random Forest** with a training set of 100 data and a test set of length 25 we obtain in the test set:

Average Relative Error	1.4184%
Maximum Relative Error	6.4480%

We consider this approximation to be very good.

Using **Gradient Boosting** we need a larger amount of data to train the model. With 1000 training data and a test set of length 36 we obtain:

Average Relative Error	1.2630%
Maximum Relative Error	15.7146%

While this approximation is not as good as the previous one, on average it works well.

4.2.2 Results for the complexified elastic block (12 parameters)

Using **Random Forest**, with a dataset of 1000 and a test training set of length 25, the following is obtained:

Average Relative Error	19.7734%
Maximum Relative Error	36.4412%

This approximation does not have the precision we are looking for.

Using **Gradient Boosting**, with a training set of 1000 data and the structure with 2000 trees, the following is achieved in the test set:

Average Relative Error	6.3382%
Maximum Relative Error	12.1351%

While it is not a bad approximation, it does not have the precision we are looking for. So we increased the size of the training set to 4000 data, a test set of 40 data and with 1000 trees we obtain in the test set:

Average Relative Error	4.7392%
Maximum Relative Error	15.9427%

4.3 Temporal Analysis

The objective of this section is to study how much the different methods improve the times compared to the original finite element method.

4.3.1 Finite Element Method

Object	Average response time for one parameter
Plate (5 parameters)	2.35 s
Elastic Block (4 parameters)	4.75 s
Elastic Block (12 parameters)	4.82 s

4.3.2 Method 1: Neural Networks

For the elastic block with 4 parameters:

Stage	Duration in seconds
Obtaining 1000 training data and POD	771.29
Neural Network training	144.07
Online stage	0.00054

Total offline stage: 915.36 s (15.256 minutes)

For the plate with 5 parameters:

Stage	Duration in seconds
Obtaining 1000 training data and POD	358.50
Neural network training	70.0
Online stage	0.00039

Total offline stage: 428.5 s (7.141 minutes)

Now if the training of this method is performed with polynomial regression instead of using neural networks for the elastic block with 4 parameters.

Stage	Duration in seconds
Obtaining 1000 training data and POD	771.29
Regression training	0.1003
Online stage	0.00002

4.3.3 Method 2: Tree-based ensemble methods

<u>For the elastic block with 4 parameters</u>

Gradient Boosting

Stage	Duration in seconds
Obtaining 1000 training data and POD	781.76
Gradient Boosting training	5.76
Online stage	0.00113

Total offline stage: 787.52 s (13.125 minutes)

Random Forest

Stage	Duration in seconds
Obtaining 1000 training data and POD	781.76
Random Forest training	122.15
Online stage	0.1159

Total offline stage: 903.91 s (15.065 minutes)

For the elastic block with 12 parameters

Gradient Boosting

Stage	Duration in seconds
Obtaining 1000 training data and POD	816.79
Gradient Boosting training	34.44
Online stage	0.0509

Total offline stage: 851.23 s (14.187 minutes)

Random Forest

Stage	Duration in seconds
Obtaining 1000 training data and POD	816.79
Random Forest training	56.37
Online stage	0.1073

Total offline stage: 873.16 s (14.553 minutes)

4.3.4 Comparison of Evaluation Times (Online vs. FEM)

Below, the response times to obtain the solution for a new parameter using the high-fidelity model (FEM) and the different reduced models in their online stage are compared.

Case Study	**Method**	**Evaluation Time (s)**	**Acceleration Factor (vs. FEM)**
Elastic block (4 parameters)	**FEM (High-Fidelity)**	**4.75**	**1x**

	Neural Network	0.00054	~ 8,800x
	Polynomial Regression	0.00002	~ 237,500x
	Gradient Boosting	0.00113	~ 4,200x
	Random Forest	0.1159	~ 41x
Elastic block (12 parameters)	**FEM (High-Fidelity)**	**4.82**	**1x**
	Gradient Boosting	0.0509	~ 95x
	Random Forest	0.1073	~ 45x
Plate (5 parameters)	**FEM (High-Fidelity)**	**2.35**	**1x**
	Neural Network	0.00039	~ 6,000x

---

5. General Discussion

The temporal analysis (Section 4.3) explicitly quantifies this advantage: the online stage shows an acceleration of up to four orders of magnitude (e.g., 4.75 s for FEM vs. 0.00054 s for the neural network in the elastic block), validating the enormous potential of these methods for real-time applications. This benefit justifies the computational investment of the offline stage, which in the studied cases remained in the order of 15 minutes.

The efficiency difference between the ML models themselves is also noteworthy. While the neural network offers the fastest online evaluation, polynomial regression proves to be an extremely efficient alternative in both training (0.10 s) and evaluation (0.00002 s) for simpler problems. Among the ensemble methods, Gradient Boosting was not only faster to train than Random Forest in these cases, but its online stage was also significantly faster.

However, there are disadvantages. For neural networks, finding the optimal architecture is not trivial and must be done for each problem. For tree-based ensemble methods, although it seems to be more robust in terms of architecture and require less data, its online stage is slightly more costly due to the additional matrix operations. Furthermore, the accuracy of this second method may depend on the ability to predict the reduced load vector, if it cannot be extracted directly. A common limitation to both approaches is that the problem's geometry cannot be changed without repeating the entire costly offline stage.

6. Conclusion

This research has demonstrated the feasibility of using non-intrusive machine learning-based methods to optimize structural analysis. Two approaches were compared: one based on neural networks to predict the solution's coefficients and another based on tree-based ensemble methods to predict the inverse of the reduced stiffness matrix.

A key finding of this research is the quantification of the computational savings: the reduced models managed to accelerate predictions by several orders of magnitude compared to FEM, with an online stage that runs in milliseconds versus the seconds required by the high-fidelity simulation. This result confirms that the time investment in an offline preparation stage is highly profitable for massive parametric analyses.

The results show that both methods can achieve high accuracy, as seen in the case of the elastic block. However, their performance can degrade in more complex problems or with insufficient data, as occurred with the neural network in the plate case. The

Gradient Boosting method appears to be more generalizable and require less data, but at the cost of a slightly slower online stage.

The non-intrusive approach is a promising alternative, especially when working with "black-box" software. Its efficiency in the online stage makes it attractive for optimization tasks and parametric analysis. Future lines of research could focus on improving the robustness of the methods, exploring more advanced deep learning architectures, and developing techniques to handle changes in the problem's geometry more efficiently.

Acknowledgments

I would like to express special thanks to Franz Chouly, who was my guiding professor during the internship. I also extend my gratitude to Francisco Pons and Fabricio Guido, whose contribution was fundamental for the successful completion of this project. I also thank Ernesto Mordecki for creating the contact with the company and, thus, making this internship possible. Finally, a recognition to Gianluigi Rozza, whose work served as an important source of inspiration. Without the valuable contribution of all of them, this project would not have been possible.

References

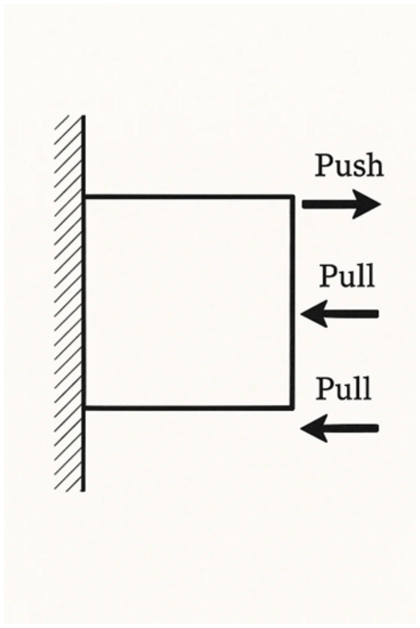
- "Material de Apoyo de unidad curricular Resistencia de Materiales 2" FING
- Langtangen, H. P., & Logg, A. (2017). *Solving PDEs in Python: The FEniCS Tutorial I* (Vol. 1). Springer. https://doi.org/10.1007/978-3-319-52462-7
- Rozza, G., Ballarin, F., Scandurra, L., & Pichi, F. (2024). *Real Time Reduced Order Computational Mechanics: Parametric PDEs Worked Out Problems*. Springer. https://doi.org/10.1007/978-3-031-38791-4
- Tannous, M., Ghnatios, C., Fonn, E., Kvamsdal, T., & Chinesta, F. (2025). *Machine Learning (ML) based Reduced Order Modelling (ROM) for linear and non-linear solid and structural mechanics* (arXiv:2504.06860v1). arXiv. https://arxiv.org/abs/2504.06860
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303-314. 10.1007/BF02551274

- The FEniCS Project. (n.d.). *Mixed Poisson equation*. FEniCS Project Documentation. Recuperado el 18 de junio de 2025, de https://olddocs.fenicsproject.org/dolfin/latest/python/demos/mixed-poisson/demo_mixed-poisson.py.html
- Baratta, I. A., & Bower, F. A. L. (2023). *Biharmonic equation*. FEniCSx v0.7.2 Documentation. Recuperado el 18 de junio de 2025, de https://docs.fenicsproject.org/dolfinx/v0.7.2/python/demos/demo_biharmonic.html
- Guzman, J., & Neilan, M. (2013). *A family of non-conforming and stable finite elements for the biharmonic equation*. [Prepublicación]. Division of Applied Mathematics, Brown University. Recuperado de https://www.dam.brown.edu/people/jguzman/documents/biharmonic_sys.pdf
- Gander, W., Gander, M. J., & Kwok, F. (2014). *Scientific Computing: An Introduction using Maple and MATLAB*. Springer.
- González Olmedo, M. (2019). *Introducción al aprendizaje automático: Apuntes para el minicurso del 7mo Coloquio Uruguayo de Matemática*. Departamento de Matemática del Litoral, Universidad de la República.

The main software and libraries used are: FEniCS, UQpy, Pytorch, Scikit-learn (Sklearn) and LightGBM.

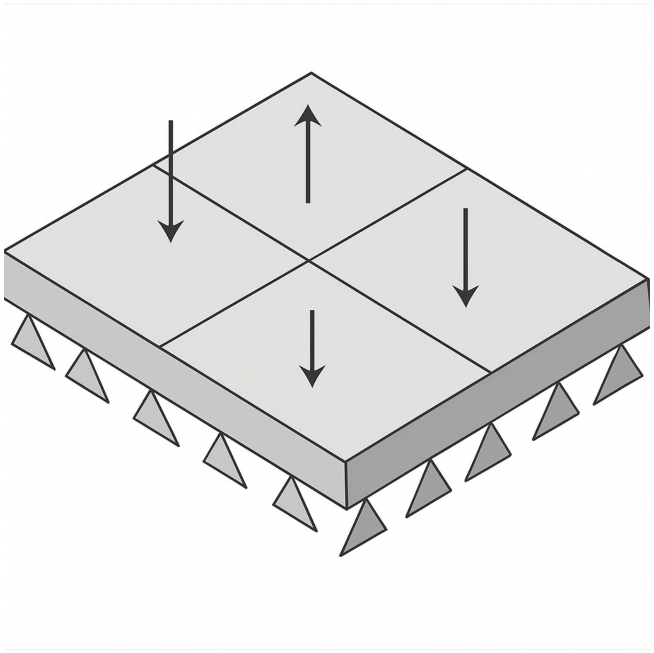
Appendices

Appendix A: Properties of the Elastic Block



- Forces between -1 to 1 N
- Stiffness between 1 and 20 Pa
- Modulus of elasticity of 1.0 Pa
- Poisson's ratio of 0.3

Appendix B: Properties of the Plate



- Loads vary from -10 to 10 kN
- Thickness varies between 4 to 60 centimeters
- Modulus of elasticity 30 GPa
- Poisson's ratio of 0.3