UCLouvain

École polytechnique de Louvain

Physics-Informed Neural Networks for Truss FEM Calibration Using Synthetic Data

A Pratt Bridge Case Study

Author: **Alexandre Flamant**
Supervisors: **João Saraiva Esteves Pacheco de Almeida, Ihsan Engin Bal**
Readers: **Pierre Dupont, Pierre Latteur, Basile Payen**
Academic year 2024--2025
Master [120] in Computer Science and Engineering

This page intentionally left blank.

UCLouvain

Hanze

Physics-Informed Neural Networks for Truss FEM Calibration Using Synthetic Data

A Pratt Bridge Case Study

Alexandre Flamant

Supervisor: João Saraiva Esteves Pacheco de Almeida
Professor, Université Catholique de Louvain

Co-supervisor: Ihsan Engin Bal
Professor, Hanze University of Applied Sciences

Eleni Smyrou
Associate Professor, Hanze University of Applied Sciences

Master [120] in Computer Science and Engineering

This page intentionally left blank.

Preparation of this work was facilitated by the use of the *IPLeiria-Thesis* template.

This page intentionally left blank.

Acknowledgements

I am profoundly grateful to my supervisors, Prof. João Saraiva Esteves Pacheco de Almeida, Prof. Ihsan Engin Bal, and Prof. Eleni Smyrou, for their generous availability, insightful guidance, and rigorous feedback. Their expertise has been invaluable in refining both the theoretical foundations and the pedagogical clarity of this thesis.

I also extend my sincere thanks to the organizers and participants of the 1st Interdisciplinary Workshop on Computer Vision Technologies for the Built Environment in Groningen. That invitation not only invigorated my motivation but also sparked stimulating discussions that greatly enriched this work.

Special appreciation goes to François, Thomas, and Leslie for their meticulous comments, which significantly improved the accessibility of these results for practicing structural engineers.

To my parents, siblings, and wider family: thank you for your unwavering encouragement and patience, both during the past year and throughout my academic journey.

Finally, I wish to thank every reader who takes the time to engage with this thesis. Your attention breathes purpose into this research.

This page intentionally left blank.

Abstract

This thesis addresses finite element model (FEM) calibration for truss structures under noisy and sparse measurements by framing the task as a supervised inverse problem: recovering member axial rigidities EA from nodal displacements and applied loads. Focusing on an 8-panel, $60\,\mathrm{m}$ Pratt truss bridge, the work targets non-destructive assessment scenarios with limited high-quality data. A fully synthetic training pipeline was developed using OpenSeesPy to generate large, standardised datasets with controlled parameter ranges and noise models using Sobol sequences to generate space-filling sampling. The case-study geometry, loading, and material bounds reflect typical bridge practice and Eurocode-motivated limits, ensuring physical realism.

Two approaches are compared: a purely data-driven, MSE-trained multilayer perceptron and a physics-informed neural network (PINN). On the member-wise prediction task, the PINN consistently outperforms the MSE model but displays unsatisfactory results. Using a category-based formulation, that is, predicting axial rigidity EA for groups of members instead of each member, drastically improves prediction accuracy, showing the PINN retaining superior performance and robustness. Overall, embedding physics as a stabilising prior and adopting design-informed targets yields more reliable, data-efficient calibration. The contributions are (i) an inverse-problem formulation for truss FEM calibration and (ii) a modular synthetic-data generation framework enabling rigorous, reproducible evaluation of learning-based calibration methods.

Keywords: Machine learning, Physics-Informed Neural Networks (PINN), Finite Element Method ,Calibration, Inverse problem, Truss, Synthetic data

This page intentionally left blank.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 3
- 1.3 Novelty and Contributions . . . 4
- 1.4 Thesis Structure . . . 5

2 State of the Art **6**
- 2.1 A Description of Machine Learning . . . 6
- 2.2 The Multilayer Perceptron . . . 7
  - 2.2.1 Anatomy of a multilayer perceptron . . . 7
  - 2.2.2 The Hyperparameters . . . 10
  - 2.2.3 The training procedure . . . 11
- 2.3 Integrating Physics Knowledge in Neural Networks . . . 14
  - 2.3.1 Physics-Informed Neural Networks . . . 14
  - 2.3.2 Finite Element-Informed Neural Networks . . . 16
- 2.4 Synthetic Data for Neural Network Training . . . 17
- 2.5 Hybrid Finite Element Modeling . . . 18
  - 2.5.1 Data-Driven Surrogate Models for Structural Analysis . . . 18
  - 2.5.2 Hybrid Machine Learning–FEM Approaches . . . 19

3 Solving the Inverse Problem of a Simple Truss **21**
- 3.1 The Linear Truss Structure . . . 21
  - 3.1.1 Definition . . . 21
  - 3.1.2 Representation . . . 22
  - 3.1.3 Properties . . . 22
- 3.2 Solving a Truss Structure . . . 23
  - 3.2.1 Computing the members' stiffness matrix . . . 25
  - 3.2.2 Assembling the structure stiffness matrix . . . 27
  - 3.2.3 Solving for Displacements and Support Reactions . . . 29
  - 3.2.4 Recovering Member Forces . . . 30
- 3.3 The inverse problem of structural analysis . . . 31
- 3.4 Analytical Solution to the Inverse Problem . . . 32
- 3.5 Traditional Machine Learning Techniques . . . 36

3.5.1 Dataset and preprocessing . . . 36
3.5.2 Overview of the Models . . . 39
3.5.3 Training and Comparison of the Models . . . 43
3.6 Neural Network approach . . . 47
3.6.1 Hyperparameter Tuning . . . 48
3.6.2 Model Comparison . . . 53
3.7 Physics-Informed Neural Networks . . . 57
3.7.1 Physics-Based Loss Functions . . . 58
3.7.2 Hyperparameter Tuning . . . 70
3.7.3 Model Comparison . . . 75
3.8 Conclusion . . . 81

4 Dataset Generation **82**
4.1 Dataset Sources . . . 82
4.1.1 Experimental Data . . . 83
4.1.2 Synthetic Dataset . . . 84
4.2 Case Study . . . 85
4.3 Generation Procedure . . . 87
4.3.1 Sampling the Parameters . . . 87
4.3.2 Modeling the Structure . . . 91
4.3.3 Solving and Extracting the Results . . . 91
4.3.4 Storing the Results . . . 91
4.3.5 Generation Framework . . . 92
4.4 Overview of the Synthetic Dataset . . . 93
4.4.1 Dataset with Member-Specific Axial Rigidity . . . 93
4.4.2 Dataset with Category-Based Axial Rigidity . . . 94

5 Application to the Pratt Truss **96**
5.1 Predicting Axial Rigidity for Each Member . . . 97
5.1.1 MSE-Based Model . . . 97
5.1.2 Physics-Informed Model . . . 101
5.1.3 Models Comparison . . . 104
5.2 Predicting category-based EA . . . 106
5.2.1 MSE-Based Model . . . 107
5.2.2 Physics-Informed Model . . . 111
5.2.3 Model Comparison . . . 114

6 Conclusion and Outlook **117**
6.1 Conclusion . . . 117
6.2 Outlook . . . 118

Bibliography 122

This page intentionally left blank.

1 Introduction

1.1 Motivation

Over the past two decades, there has been a marked increase in the frequency and severity of natural disasters [1]. These events often result in the devastation of residential buildings, healthcare facilities, and public infrastructure. When structures remain standing, it becomes critical to assess their structural integrity, as such events can induce deformations, material fatigue, and cracking, among other forms of damage. Consequently, prior knowledge about the structural system may become obsolete, making post-event usability assessments challenging.

A comparable issue arises, albeit under less extreme circumstances, in the context of building renovation, a practice that is becoming increasingly prevalent. In light of growing ecological awareness and environmental responsibility, there is a noticeable shift away from new construction towards the renovation of existing buildings. Similar to post-disaster situations, information about the structure's original condition may be outdated or unavailable at the time of renovation.

To evaluate the structural integrity and performance, engineers rely on computational models. Modeling involves formulating a representation of a system or process to facilitate analysis and prediction. In structural engineering, models are used to estimate internal forces and predict structural responses.

These models are built upon physical and mathematical principles. However, a comprehensive representation of structural systems often entails complex interactions between forces, materials, and geometry, which can be computationally demanding. To address this, engineers employ simplified models that reduce computational load while retaining sufficient accuracy. The choice of modeling approach depends on the specific characteristics of the structure, such as material properties, which can be modeled according to different hypotheses, as displayed in Figure 1.1.

Depending on structural complexity and analytical goals, different modeling techniques are employed. Among these, finite element method (FEM) models are widely

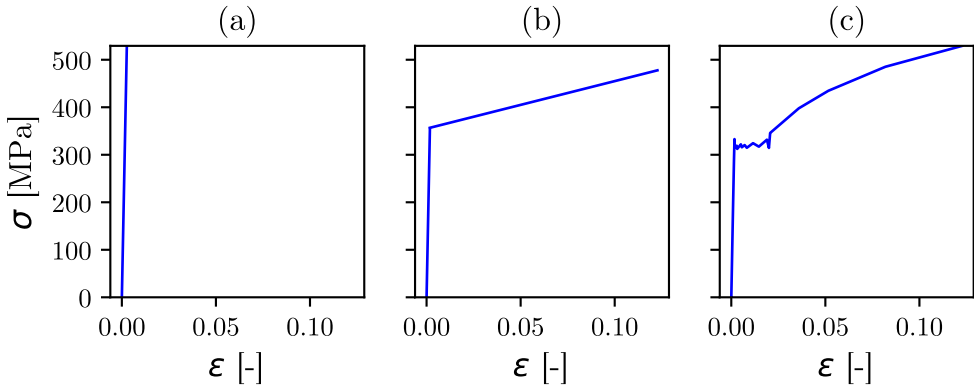


Figure 1.1: *Comparison of the strain-stress relationship of S355 steel for (a) linear hypothesis, (b) bilinear hypothesis, and (c) high-fidelity hypothesis* [2].

recognized for their versatility and prevalence in engineering practice. One of the most common finite element analysis methods in structural engineering is linear analysis. This approach assumes a linear relationship between structural parameters. It is computationally efficient and suitable for most structures and loading scenarios.

However, FEM requires calibration, which involves defining its initial configuration to obtain accurate results. This configuration contains parameters such as meshing and structural properties, including member cross-sections, material properties, and support conditions of the structure.

This process is primarily conducted through the engineer's experience and reference values, as designers typically possess comprehensive knowledge of the materials and manufacturing processes. In the event of an incident that alters the physical condition of the structure, it's of the utmost importance that the structure's integrity is re-evaluated to its current state. In this case, a standard method is often the use of trial and error, combined with engineering knowledge, to match measurements taken on-site.

Unfortunately, for large-scale structures such as multistory buildings or bridges, this often involves cost-, time-consuming, and potentially destructive measurements that can be prohibitive or incompatible with the speed required to perform the assessment. Another option is to perform fewer measurements or measurements that involve a higher tolerance for measurement errors. This implies noisy data, making it more challenging to apply the trial-and-error method.

Statistical inference offers tools to address these issues. It involves drawing conclusions about a population based on sample observations [3]. In recent years, machine learning (ML) has emerged as a powerful subset of statistical inference. Notably, neural networks have demonstrated significant success in modeling complex, nonlinear phenomena. These models are tailored to extract patterns from noisy data and thus permit better robustness to it.

This thesis presents an exploratory study on the application of machine learning techniques for structural analysis, with a specific focus on enhancing the calibration of finite element models. In essence, the objective is to estimate unknown model param-

eters using indirect measurements.

1.2 Objectives

Figure 1.2: *Illustration of the reference truss bridge in a location where access is difficult.*

This research focuses on the study of a specific 8-panel 60m long Pratt Truss Bridge as illustrated in Figure 1.2. The objective of the project is to predict properties of the structure from non-destructive measures on the structure. In this very case, the goal is to predict the axial rigidity EA of each element of the structure. This quantity, specific to each member, is essential to the structure definition, as a change in rigidity will impact the deformation of each member and thus the repartition of stresses within the structure. This model will estimate the structure's parameters from measurements of deflections and loading. The motivation behind this is that these measurements can be gathered with accessible non-destructive methods, such as using drone photogrammetry to measure deformations of the structure.

Multiple hypotheses will be assumed true during this work. It is supposed that the structure behaves as a linear truss. Meaning, the structure is not subject to bending forces and the stress-strain relationship is considered linear as shown in Figure 1.1 (a).

This structure will be the medium to explore and propose solutions to several problems:

- **Addressing Data Sparsity:** It is often said that "*there is no such thing as too much data*". In the context of machine learning, data quality and quantity are crucial factors in determining model performance. However, in structural analysis, datasets are often limited and heterogeneous in quality. Enhancing data availability and reliability is a key goal of this work.

- **Training and Comparing Multiple Models:** A wide array of machine learning algorithms exists. This thesis will implement and evaluate several models to identify those that yield the most accurate and reliable predictions.
- **Incorporating Physical Knowledge into Model Training:** Traditional training of machine learning models relies on statistical loss functions that overlook the domain-specific characteristics of the problem. This research investigates the integration of physics-informed loss functions to improve model performance in structural analysis.
- **Proposing a FEM Model Calibration Procedure:** The ultimate objective is to develop a systematic procedure for calibrating FEM models using the insights and techniques explored throughout the thesis. This includes guidelines on model training, optimization, and application for specific structural systems.

The integration of machine learning into structural analysis remains an emerging area, with relatively few existing applications. As discussed in Chapter 2, most studies focus on structural health monitoring tasks, such as crack detection, rather than analytical modeling. When structural analysis is addressed, the emphasis often lies on auxiliary aspects, such as mesh generation or material modeling, with few works directly tackling structural response prediction.

This thesis pursues a different objective. Machine learning and statistical methods are inherently predictive and well-suited for estimating specific quantities. However, in structural engineering, a detailed understanding of system behavior is often essential, something typically achieved through physics-based simulations, such as FEM.

The aim here is not to replace these simulations but to augment them to improve their robustness to noise in initial measurements. Machine learning is employed as an interface between empirical data and high-precision models, enhancing the effectiveness and usability of traditional analytical methods.

1.3 Novelty and Contributions

Although machine learning techniques have been widely adopted in structural engineering for detection tasks, such as crack detection in concrete, they are recently gaining traction in the context of hybrid FEM, where machine learning has been used to improve mesh generation or prediction. Their application in FEM calibration for structural analysis, particularly in the context of noisy measurements, remains largely unexplored.

The present work distinguishes itself through three main contributions:

1. **Inverse-problem formulation for FEM calibration:** We recast the estimation of member axial rigidities as a supervised inverse problem, mapping measured nodal displacements and applied loads directly to material parameters. This contrasts with the forward-analysis focus of most prior studies.

2. **Development of novel physics-informed losses tailored for linear truss PINNs:** Existing physics-based loss formulations, originally designed for predicting structural responses under loading, are adapted to address the specific requirements of the present problem. In addition, two new loss components are introduced to explicitly embed the principles of local equilibrium and energy conservation into the model, thereby enhancing its physical consistency and predictive robustness.
3. **Development of a fully synthetic dataset-based training:** A generation method to produce large, high-quality datasets by computing a range of structures from FEM-based simulation coupled with stochastic sampling techniques to generate datasets representative of the problem.

These contributions collectively advance the state of the art by bridging empirical ML techniques with the rigorous requirements of structural analysis.

1.4 Thesis Structure

The document is organised as follows:

1. **Chapter 1 — Introduction:** Motivation, objectives, novelty, and an overview of the thesis.
2. **Chapter 2 — State of the Art:** A critical review of machine-learning applications in structural engineering, highlighting gaps addressed by this research.
3. **Chapter 3 — Solving the Inverse Problem of a Simple Truss:** Step-by-step application of the proposed inverse-calibration concept on a small, fully-manipulable truss, establishing fundamental ideas used later.
4. **Chapter 4 — Dataset Generation:** Detailed presentation of the synthetic-data pipeline: parameter sampling, FE modelling, simulation, and preprocessing.
5. **Chapter 5 — Pratt Truss Bridge Case Study:** End-to-end calibration on the reference bridge, model comparison, and validation against classical techniques.
6. **Chapter 6 — Conclusion and Outlook:** Summary of findings, limitations, and directions for future research.

> **Example**
>
> To facilitate reader understanding of key concepts, this manuscript includes highlights and key examples through these text boxes to promote intuitive comprehension of the subject.

2

State of the Art

2.1 A Description of Machine Learning

Machine learning (ML) is the study of computer algorithms and models that automatically improve through experience. It is a multidisciplinary field combining theories from mathematics, probability, statistics, and information theory, among others [3].

Data represents this experience. It consists of measured observations of events or systems under study. These data points, referred to as *samples*, are typically sparse compared to the theoretically infinite possible observations [4].

From the gathered dataset, the machine-learning model generalizes the underlying rules of the system, allowing it to perform tasks such as classification or regression. In supervised ML, datasets consist of n measurements containing d features $\mathbf{X}$ and corresponding p targets $\mathbf{Y}$ (expected outputs), forming the basis to train a predictive model $\mathcal{M}(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^p$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1^T} \\ \vdots \\ \mathbf{x_n^T} \end{bmatrix} \in \mathbb{R}^{n \times d} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y_1^T} \\ \vdots \\ \mathbf{y_n^T} \end{bmatrix} \in \mathbb{R}^{n \times p} \tag{2.1}$$

Once the model $\mathcal{M}$ is trained, feeding it new, unseen, features $\mathbf{x}^*$ predicts $\widehat{\mathbf{y}}^*$, an approximation of the actual unknown target $\mathbf{y}$:

$$\mathcal{M}(\mathbf{x}^*) = \widehat{\mathbf{y}}^* \approx \mathbf{y} \tag{2.2}$$

Figure 2.1 illustrates the process of machine learning from experimentation to prediction based on a trained model.

To ensure consistency, the following notation is adopted throughout this manuscript when referring to data:

- Bold uppercase letters denote matrices, e.g., $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \end{bmatrix}^\top$;
- Bold lowercase letters denote column vectors, e.g., $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}^\top$;

- Regular lowercase letters denote scalar values, e.g., x;
- A superscript asterisk denotes unseen data and values derived from unseen data (i.e., not included in the training set), e.g., $\mathbf{x}^*$;
- A circumflex accent denotes an estimated or predicted value, e.g., $\widehat{\mathbf{x}}$.

Any notation violating this convention will be explicitly mentioned.

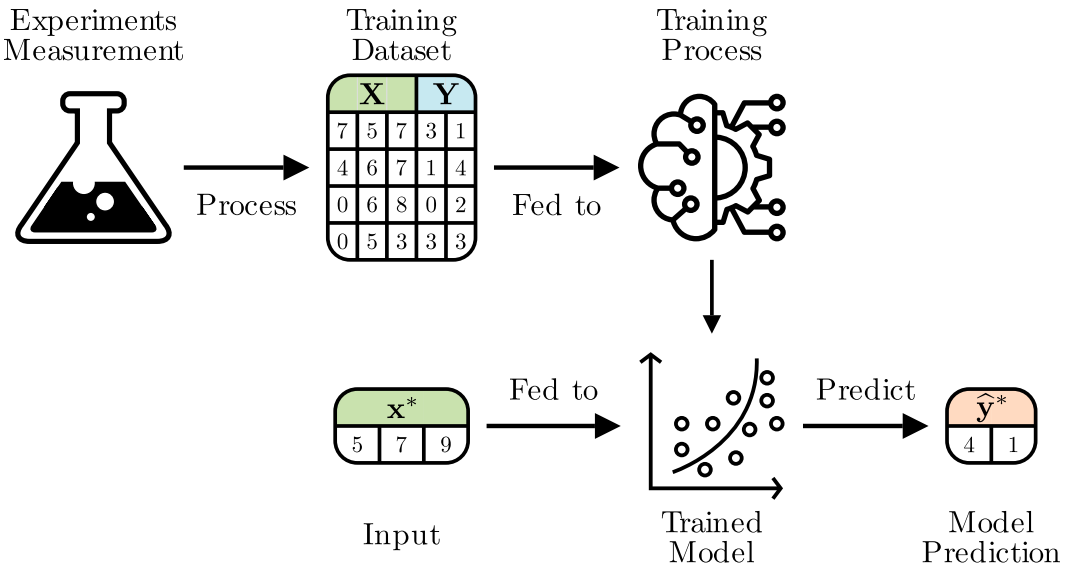


Figure 2.1: *Illustration of the machine learning procedure*

2.2 The Multilayer Perceptron

A Multilayer Perceptron (MLP) is a fundamental class of model architecture among artificial neural network-based models. It consists of sequential layers that apply affine transformations and nonlinearities to approximate a mapping from input features to target outputs [5].

During this section, the notation $a_i^{(L)}$ will be used extensively, where the superscript refers to the layer, and the subscript refers to the location on the layer.

2.2.1 Anatomy of a multilayer perceptron

An MLP comprises three main components, illustrated in Figure 2.2:

- **Input Layer:** Contains d neurons, where d corresponds to the dimensionality of the input feature space (i.e., the number of features). Each neuron receives one feature from the input vector;
- **Hidden Layers:** A series of one or more intermediate layers of neurons that apply nonlinear transformations. These layers enable the network to learn complex patterns. Although it is a common design choice, the hidden layers do not need to have the same number of neurons;

- **Output Layer:** Contains p neurons, where p corresponds to the dimensionality of the prediction target. The output layer produces a vector $\widehat{\mathbf{y}}$, which forms the model's prediction.

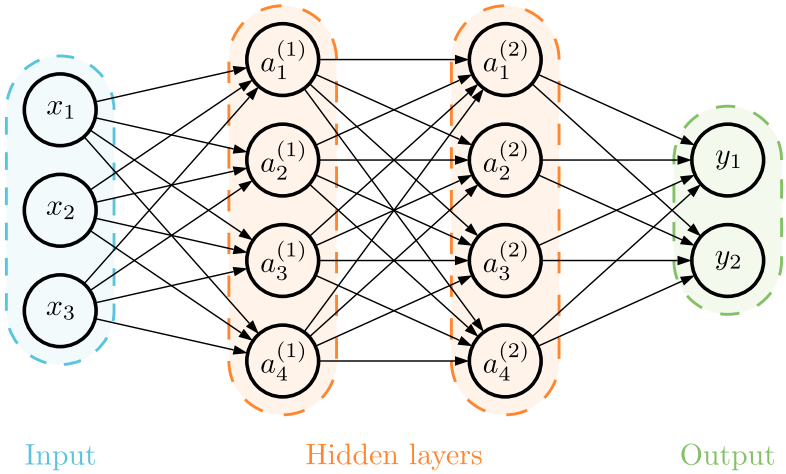


Figure 2.2: *Architecture of an MLP with an input layer containing three neurons, two hidden layers containing four neurons each, and an output layer containing two neurons.*

Each neuron in a layer is fully connected to the neurons in the adjacent layer through weighted connections. As such, every neuron in the layer L receives the activation $\mathbf{a}^{(L-1)}$ of the layers from the previous layer.

To better understand the internal computations, consider a single neuron i in the layer L represented in Figure 2.3. This neuron outputs a value that is a transformation of the output of all the neurons from the previous layer.

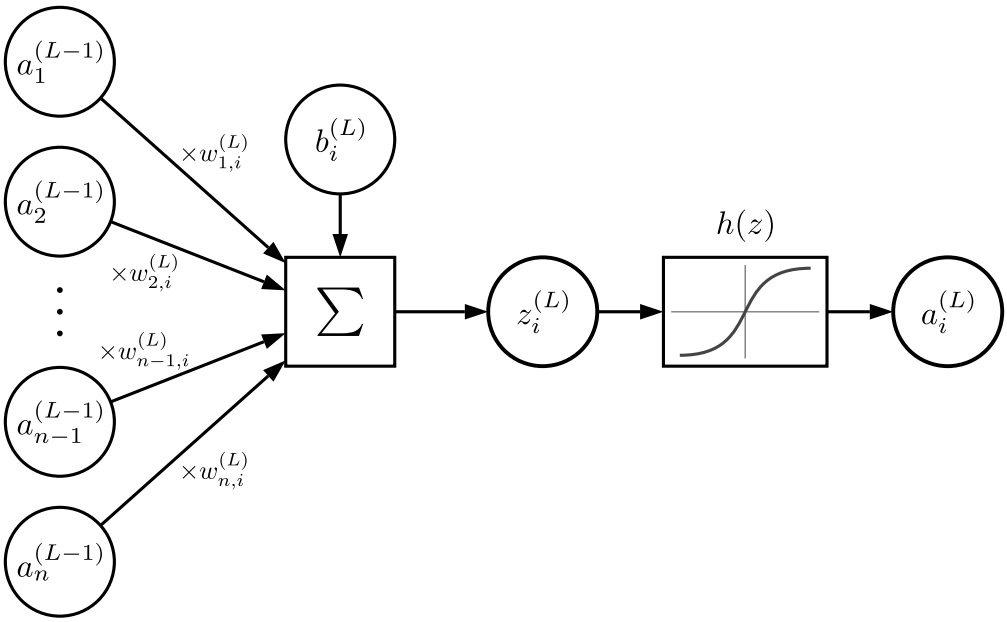


Figure 2.3: *Diagram of the operations in the neural network's neuron i of the layer L to compute its activation $a_i^{(L)}$*

To perform this transformation, each neuron stores a collection of parameters:

- **The weights:** Each neuron i from the layer L stores a weight $w_{j,i}^{(L)}$ that weights its connection from the activation of the neuron j from the previous layer $L-1$. The weights of the n neurons in layer $L-1$ are stored in $\mathbf{w}_i^{(L)}$:

$$\mathbf{w}_i^{(L)} = \begin{bmatrix} w_{1,i}^{(L)} & w_{2,i}^{(L)} & \cdots & w_{n-1,i}^{(L)} & w_{n,i}^{(L)} \end{bmatrix}^T \in \mathbb{R}^{|L-1|} \tag{2.3}$$

- **The bias:** The bias $b_i^{(L)}$ is a scalar value that is added to the neuron and acts as the intercept of the weighted sum.

The set of all these parameters is referred to as the model's parameters, denoted as θ. These parameters condition the model; as such, two models with the same architecture but with different parameters θ won't produce the same output. When referring to the model with an emphasis on its parameters, the notation $\mathcal{M}_\theta(\mathbf{x})$ will be used.

The output of neuron i in this layer, referred to as the activation $a_i^{(L)}$, is computed by computing $z_i^{(L)}$, the weighted sum of the activation of the previous layers and the bias. The activation of the neuron is then calculated by applying a differentiable nonlinear activation function $h(z)$ to $z_i^{(L)}$:

$$z_i^{(L)} = b_i^{(L)} + \sum_{j=1}^{n} w_{ji}^{(L)} a_j^{(L-1)} = b_i^{(L)} + \mathbf{w}_i^{(L)^T} \cdot \mathbf{a}^{(L-1)} \tag{2.4}$$

$$a_i^{(L)} = h(z_i^{(L)}) \tag{2.5}$$

With the understanding of how a neuron behaves in a neural network, we can generalize to the entire layer. Consider $\mathbf{b}^{(L)}$ the vector containing all the bias and $\mathbf{W}^{(L)}$ the matrix containing all the weights stored in each m neuron layer in layer$L$:

$$\mathbf{b}^{(L)} = \begin{bmatrix} b_1 & b_2 & \cdots & b_{m-1} & b_m \end{bmatrix}^T \in \mathbb{R}^{|L|} \tag{2.6}$$

$$\mathbf{W}^{(L)} = \begin{bmatrix} \mathbf{w}_1^{(L)} & \mathbf{w}_2^{(L)} & \cdots & \mathbf{w}_{m-1}^{(L)} & \mathbf{w}_m^{(L)} \end{bmatrix}^T \in \mathbb{R}^{|L| \times |L-1|} \tag{2.7}$$

We can generalize the equations 2.4 and 2.5 to the entire layer using matrix notation.

$$\mathbf{z}^{(L)} = \mathbf{b}^{(L)} + \mathbf{W}^{(L)^T} \mathbf{a}^{(L-1)} \tag{2.8}$$

$$\mathbf{a}^{(L)} = h\left(\mathbf{z}^{(L)}\right) \tag{2.9}$$

The nonlinear function $h(z)$ is essential. Without its application, the model is equivalent to a linear model, which drastically reduces its capability to capture and model complex systems. Applying it allows for the *universal approximation theorem* [6], which states that A feedforward neural network with a single hidden layer and a nonlinear activation function can approximate any continuous function on a compact subset of $\mathbb{R}^p$, to arbitrary precision, given a sufficient number of neurons.

2.2.2 The Hyperparameters

The parameters of the models correspond to the weights and biases of the model for a set model architecture. By analogy, the hyperparameters are the parameters that govern the MLP architecture itself, controlling its capacity, expressiveness, and learning dynamics. These hyperparameters must be carefully selected to strike a balance between model complexity and performance.

The three primary hyperparameters of MLPs are:

- **Number of hidden layers and neurons:** These determine the depth and width of the network. While a single hidden layer is theoretically sufficient for the universal approximation theorem [6], deeper architectures (i.e., architectures with more hidden layers) are generally more efficient in practice, requiring fewer neurons and exhibiting better convergence and performance.
  Together, these values define the number of trainable parameters in the model, also referred to as the *model capacity*. While this metric alone is not enough to capture the MLP's ability to capture complex relationships, it is helpful to quantify how much the model can learn.
- **Activation function:** The choice of activation function has a significant impact on training dynamics and model expressiveness. These functions should be nonlinear and differentiable everywhere because their gradient is used to optimize the parameters θ as developed in section 2.2.3.

Different functions offer distinct behaviors concerning gradient flow and nonlinearity. As some perform better than others based on the specifics of the task, we will consider the following activation functions illustrated in Figure 2.4:

- **Sigmoid:**
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.10}$$
  Maps input to the $(0, 1)$ interval. Prone to vanishing gradients, particularly in deep networks.
- **Tanh:**
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.11}$$
  Outputs values in $(-1, 1)$ with zero-centered symmetry. It offers a smoother gradient flow than sigmoid.
- **ReLU (Rectified Linear Unit):**
$$\text{ReLU}(x) = \max(0, x) \tag{2.12}$$
  Computationally efficient and promotes sparse activations because many neurons won't activate as:
$$a_i^{(L)} = \text{ReLU}(z_i^{(L)}) = 0 \quad \text{if } z_i^{(L)} \leq 0$$

It presents two main drawbacks: the first is that the derivative is discontinuous at 0, and the second is its susceptibility to the "dying neuron" problem, where gradients become zero for negative inputs, meaning neuron weights can't be updated anymore [7].

- **GELU (Gaussian Error Linear Unit):**

$$\text{GELU}(x) = x\,\Phi(x) \tag{2.13}$$

Where $\Phi(x)$ is the standard Gaussian cumulative distribution function. This activation function aims at solving the ReLU differentiability issue.

- **Leaky ReLU (Leaky Rectified Linear Unit):**

$$\text{LeakyReLU}(x, \alpha) = \max(x, 0) + \alpha \min(x, 0) \tag{2.14}$$

Introduces a slight slope α for negative inputs to address the dying neuron issue with standard ReLU.

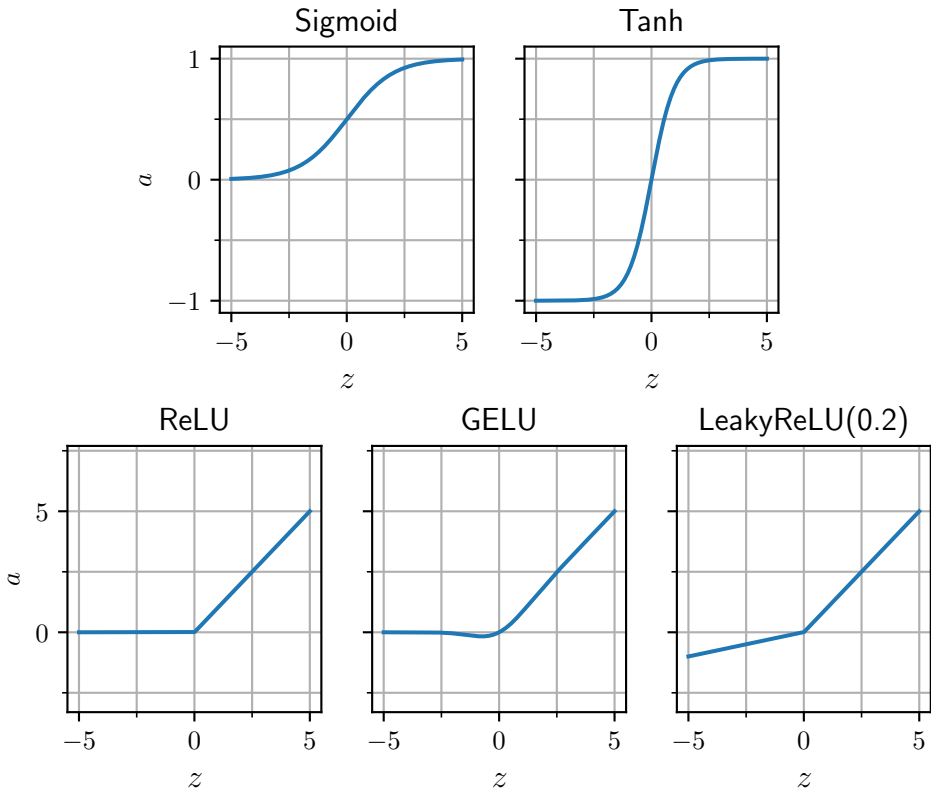


Figure 2.4: *Comparison of the activation functions*

2.2.3 The training procedure

Training a model involves learning patterns and relationships between features and output to produce accurate results from unseen data. This is done by finding the correct values for all weights and biases stored in each neuron.

During training, the model is fed examples of data from which it predicts $\widehat{\mathbf{y}}$, estimating the correct target value $\mathbf{y}$. Initially, the answer is random as the weights are initialized randomly.

To assess the quality of the answer, we use a differentiable metric that quantifies how far off the prediction $\hat{y}$ is from the actual value y. Such a function is called a loss function $\mathcal{L}(\widehat{\mathbf{y}}, \mathbf{y})$. For a perfect model, this loss is equal to zero, meaning that the model perfectly predicts the data and finds a set of valid underlying rules in the data.

A standard loss function for a regression task is the Mean Squared Error (MSE), which measures the error between the predicted value and the actual value. This error is squared, allowing for penalizing large errors more than minor errors. Intuitively, this loss improves learning by first limiting significant inconsistencies in prediction.

$$\text{MSE}(\widehat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{y}_i - y_i\right)^2 \tag{2.15}$$

Feeding the network with a batch of m training features $\mathbf{X}$ and training target $\mathbf{Y}$, we can compute a batch loss:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\widehat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\mathcal{M}_\theta(\mathbf{x}_i), \mathbf{y}_i) \tag{2.16}$$

Which is a function of all the models' parameters θ, the weights and biases of each neuron.

Now, the task is to perform an optimization to minimize the loss of the model:

$$\min_\theta J(\theta) \tag{2.17}$$

The learning process aims to find parameters θ that minimize $J(\theta)$. This can be done using the gradient descent algorithm (GDA) [8] by iteratively updating parameters in the direction of the steepest descent of the cost function as illustrated in Figure 2.5:

$$\theta \leftarrow \theta - \eta\nabla_\theta J(\theta) \tag{2.18}$$

Where η is the *learning rate*, a hyperparameter controlling the step size, and $\nabla_\theta J(\theta)$ represents the gradient of the cost function with respect to the parameters.

The update of each layer is done using the backpropagation algorithm. It essentially involves computing the chain rule of differentiation to determine the contribution of each layer to the batch loss and updating the layer's parameters accordingly.

$$\mathbf{W}^{(L)} \leftarrow \mathbf{W}^{(L)} - \eta\frac{\partial J}{\partial \mathbf{W}^{(L)}} \tag{2.19}$$

$$\mathbf{b}^{(L)} \leftarrow \mathbf{b}^{(L)} - \eta\frac{\partial J}{\partial \mathbf{b}^{(L)}} \tag{2.20}$$

While basic gradient descent provides a concrete mechanism for minimizing the

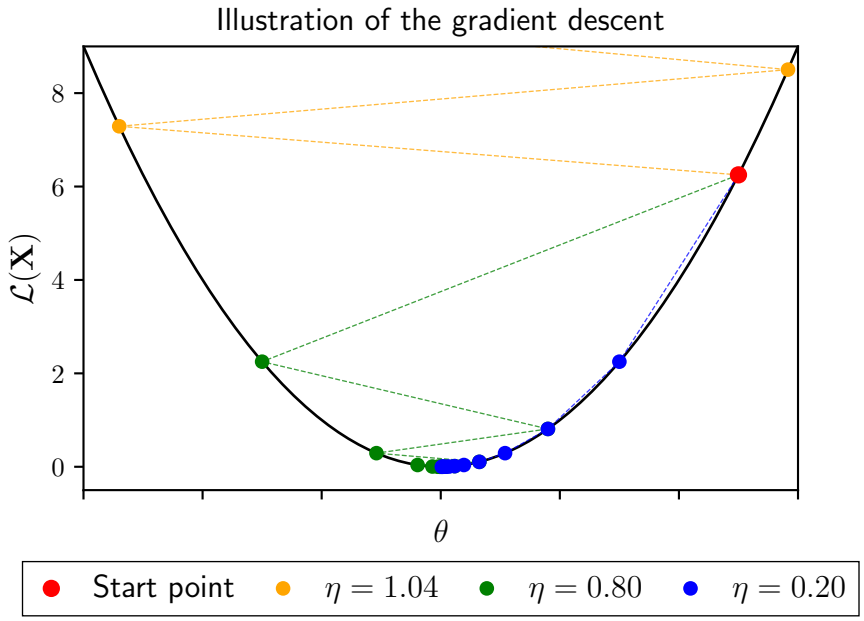


Figure 2.5: *Illustration of the behavior of the gradient descent algorithm for multiple learning rates, where each dashed line indicates the transition to the next value after an iteration*

cost function, it also has notable drawbacks. In particular, the algorithm can become trapped in local minima or saddle points, and its convergence speed is highly sensitive to the choice of the learning rate η as shown in Figure 2.5. To address these issues, a variety of enhanced optimizers have been proposed, including:

- **Fast Gradient Descent** [9], which incorporates a look-ahead step to improve convergence rate;
- **Momentum-based gradient descent** [10], which accumulates a velocity term to damp oscillations;
- **Adam** [11], which adaptively scales per-parameter learning rates using first and second moments of the gradients;
- **L-BFGS** [12], a quasi-Newton method that approximates second-order curvature information.

In practice, training proceeds over multiple *epochs*: each epoch consists of applying the update rules across the entire training set once. Since many real-world datasets are too large to process in a single pass, we split the data into smaller *batches*. At each batch step, the network computes the loss on only that subset of examples and updates its parameters accordingly. For instance, with a dataset of 1,000 samples and a batch size of 250, one epoch consists of four parameter updates. Iterating over many epochs allows the model to gradually refine its parameters and move closer to the global minimum. The figure 2.6 illustrates the training process with batches.

By tuning both the optimizer (e.g., Adam vs. momentum) and the epoch/batch schedule, we ensure efficient, stable convergence of the network.

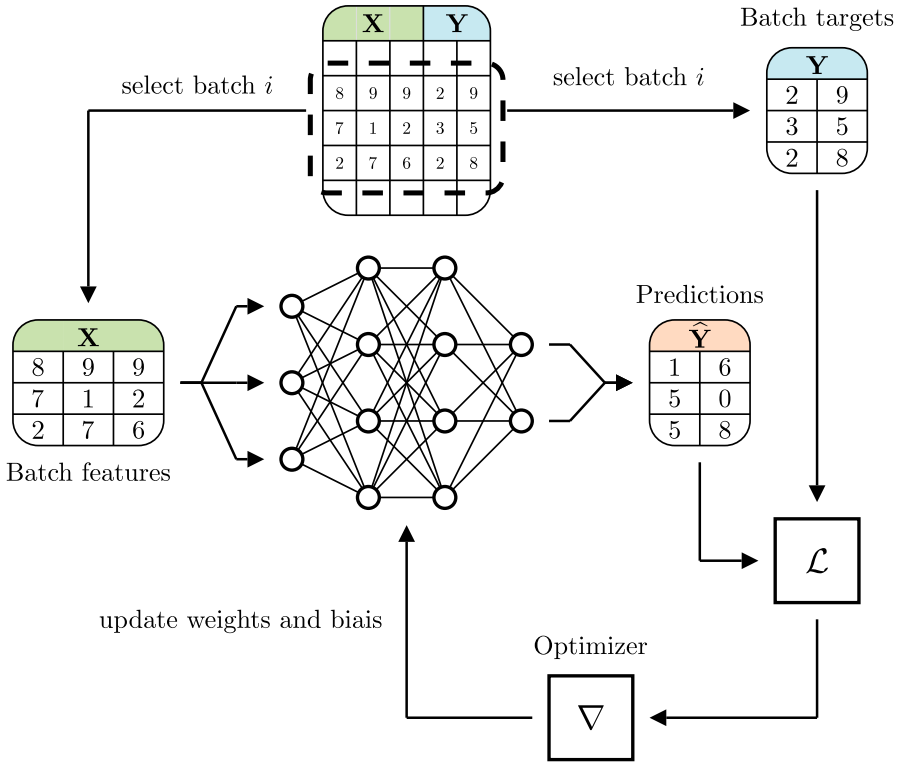


Figure 2.6: *Schematic of MLP training: the dataset is divided into mini-batches, each of which generates a loss and a parameter update; one full pass over all batches constitutes an epoch.*

2.3 Integrating Physics Knowledge in Neural Networks

2.3.1 Physics-Informed Neural Networks

The neural networks are commonly data-driven. This means that the learning pattern is done by leveraging purely statistical relationships between data. This approach is the most common one and has applications in state-of-the-art models such as GPT-3 the natural language processing model from which originate the well known ChatGPT [13], AlphaGo a model trained to play the game of Go [14] or YOLOv5, a model for the detection task [15], and more.

However, some tasks can leverage more than what appears in the data. Some problems involve physics and, as such, relate to known physical laws that can be leveraged to improve learning. This is the intuition behind the Physics-Informed Neural Networks (PINN) proposed by Raissi et al. [16].

In his framework, Raissi suggests that knowledge of partial differential equations can be leveraged to improve the model. Consider the problem of a one-dimensional mass-spring oscillator:

$$m\frac{d^2u}{dt^2} + ku = 0 \tag{2.21}$$

We know that this equation is the ground-truth, as such we know that if we define a model $\mathcal{M}(t) = \widehat{u}(t)$, we can introduce a residual based on the differential equation

that is equal to 0 only if the model is perfect, that is if $\mathcal{M}(t) = \widehat{u}(t) = u(t)$:

$$R_\theta\left(\widehat{u}(t)\right) = m\frac{d^2\widehat{u}(t)}{dt^2} + k\widehat{u}(t) \tag{2.22}$$

As explained in section 2.2.3, neural networks are differentiable with respect to their input, in this case, t, meaning that the residual is calculable. Thus, we can compute $\frac{d^2u}{dt^2}$; in practice, this is done using automatic gradient calculation through the libraries dedicated to neural networks [17].

The framework introduced by Raissi et al. uses this residual to augment the data-based loss:

$$\mathcal{L}\left(\widehat{\mathbf{u}}, \mathbf{u}\right) = \mathcal{L}_{data}\left(\widehat{\mathbf{u}}, \mathbf{u}\right) + \lambda\mathcal{L}_{physics}\left(\widehat{\mathbf{u}}\right) \quad , \lambda \in \mathbb{R} \tag{2.23}$$

Where the data-based loss is the MSE loss between $\widehat{u}$ and u and the physics-based loss is the MSE loss between R and 0

$$\mathcal{L}_{data}\left(\widehat{\mathbf{u}}, \mathbf{u}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{u}_i - u_i\right)^2 \tag{2.24}$$

$$\mathcal{L}_{physics}\left(\widehat{\mathbf{u}}\right) = \frac{1}{n}\sum_{i=1}^{n}\left[R_\theta\left(\widehat{u}\left(t\right)\right)\right]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[m\frac{d^2\widehat{u}(t)}{dt^2} + k\widehat{u}(t)\right]^2 \tag{2.25}$$

Using this loss, let the model learn through two means during training: the first is through data likelihood, which determines whether the prediction fits the expected value. The second incorporates physics knowledge by enabling the model to learn a specific derivative pattern linking input and output values. The Figure 2.7 illustrates the adapted training process to train PINN.

Raissi et al. demonstrated the PINN approach on several forward and inverse problems involving PDEs, including fluid dynamics and quantum mechanics, showing that neural networks could accurately recover solutions (and even unknown parameters in the equations) by minimizing the physics-based loss [16]. The key advantage is that the training data can be augmented or entirely replaced by physical knowledge. Instead of requiring a large dataset of input–output pairs generated by experiments or high-fidelity simulations, the PINN uses the equations of motion or equilibrium as a teacher. This paradigm has since been extended and applied to diverse fields. For instance, subsequent studies have applied PINNs to solve Navier–Stokes flow fields [18] and structural mechanics equations [19], often achieving good agreement with classical solutions while bypassing the need for exhaustive datasets. In essence, PINNs blur the line between data-driven learning and numerical simulation, using neural networks to approximate physics-governed mappings with the benefit of built-in physical correctness. Unfortunately, PINNs also have multiple drawbacks. Firstly, they are notoriously hard to train to obtain good results. [20] It is also worth mentioning that the physics losses are often computationally expensive and thus require more training time and resources.

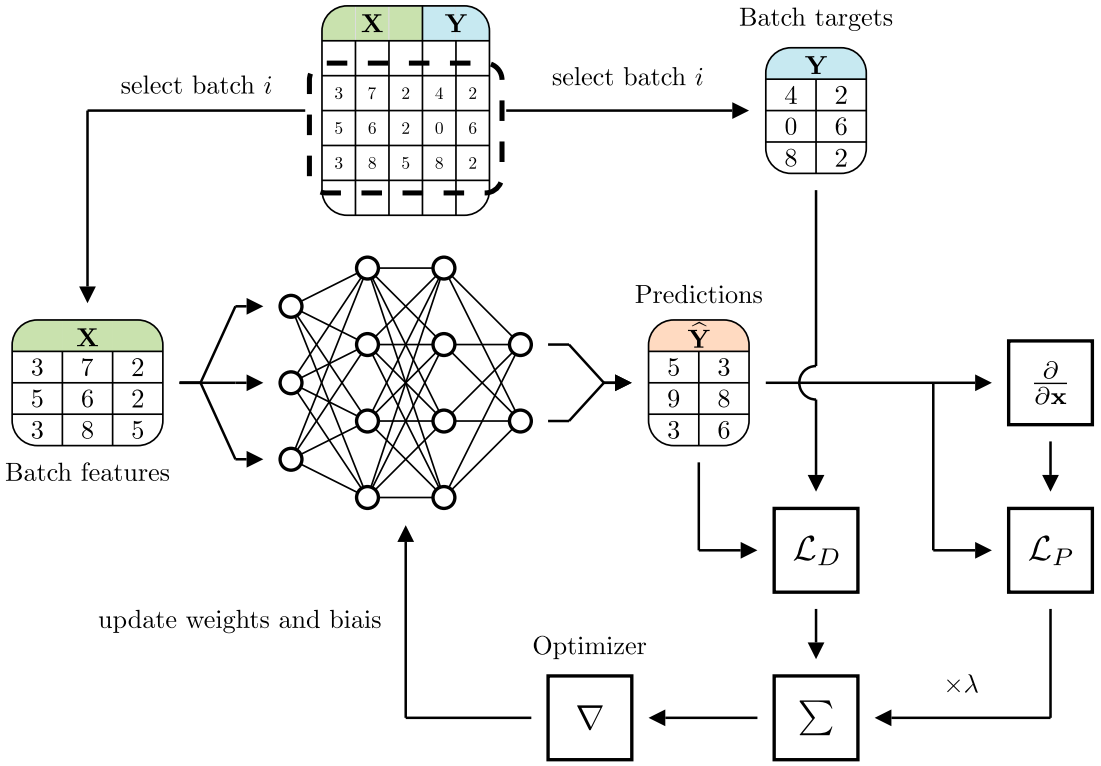


Figure 2.7: *Schematic of MLP training with PINN: the dataset is divided into mini-batches, each of which generates a loss and a parameter update; one full pass over all batches constitutes an epoch. The $\mathcal{L}_{data}$ has been abeviated to $\mathcal{L}_D$ and $\mathcal{L}_{physics}$ has been abbreviated to $\mathcal{L}_P$.*

2.3.2 Finite Element-Informed Neural Networks

Neural networks are well-suited for continuous and dynamic systems that are described by partial differential equations throughout the domain. Unfortunately, the application to discrete and/or non-dynamic systems is not straightforward.

This is such a case for linear structural analysis FEM, where the application of the PINN framework is limited. To solve this issue, Le-Duc et al. proposed the Finite Element Informed Neural Networks (FEI-NNs), an adaptation of PINN to finite element models [21]. They trained an MLP to predict the deformation of linear trusses and beams. The proposed method significantly improved the model's prediction compared to the classical MLP.

The adaptation involves the computation of residuals derived from quantities included in the finite element model and the model's output. Embedding physical knowledge from the finite element simulation. In their paper, Le-Duc et al. used the constitutive equation of structural systems:

$$\mathbf{Ku} = \mathbf{q} \tag{2.26}$$

Where $\mathbf{K}$ is the stiffness matrix of the structure, $\mathbf{u}$ is the nodal deformations of the structure, and $\mathbf{q}$ is the load applied to the structure.

Using an MLP to approximate $\widehat{\mathbf{u}}(\mathbf{x}) \approx \mathbf{u}$, they created a residual and crafted a finite-

element informed loss from it:

$$R_\theta\left(\widehat{\mathbf{u}}\right) = \mathbf{K}\widehat{\mathbf{u}} - \mathbf{q} \tag{2.27}$$

$$\mathcal{L}_{FEM}(\widehat{\mathbf{u}}) = \frac{1}{n}\sum_{i=1}^{n}\left[R_\theta\left(\widehat{\mathbf{u}}\right)\right]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{K}\widehat{\mathbf{u}} - \mathbf{q}\right]^2 \tag{2.28}$$

They used the same strategy as PINN and developed a two-part loss:

$$\mathcal{L}\left(\widehat{\mathbf{u}}, \mathbf{u}\right) = \mathcal{L}_{data}\left(\widehat{\mathbf{u}}, \mathbf{u}\right) + \lambda\mathcal{L}_{FEM}\left(\widehat{\mathbf{u}}\right) \quad , \lambda \in \mathbb{R} \tag{2.29}$$

In another paper, Meethal et al. developed a similar approach, using only the loss defined in equation 2.28 to train an MLP to predict the deformation of trusses and beams. They achieved good results with 99% of accuracy [22].

2.4 Synthetic Data for Neural Network Training

As we have seen, training an ML model requires a dataset, ideally with a large number of examples representing a diverse range of scenarios. Unfortunately, acquiring large volumes of real-world structural data, whether from laboratory experiments or in-service measurements, can be prohibitively expensive, time-consuming, or even destructive. It is particularly complex for structures that exhibit extreme behaviors, such as large deformations, which are unlikely to be built in reality. As a practical alternative, *synthetic data* generated by numerical simulations offers a scalable route to assemble datasets covering wide variations in material properties, geometric configurations, and damage scenarios. In structural engineering, several recent studies have demonstrated that neural networks trained solely on synthetic datasets can generalize effectively to real-world tests, validating the viability of an entirely synthetic training pipeline.

For example, Seventekidis et al. trained a convolutional neural network on over 10 000 simulated vibration signatures from a carbon-fiber–reinforced-polymer truss, each labeled by damage location and severity. Without ever seeing experimental data during training, their network detected and localized multiple damage cases in a physical prototype with nearly 95% accuracy, matching laboratory measurements and illustrating robust sim-to-real transfer [23]. Similarly, Lee et al. first updated a steel-frame finite-element "digital twin" to match its healthy-state modal properties, then synthesized 5 000 damage configurations via simulation. A deep classifier trained on these scenarios correctly identified damage patterns in the real structure over 90% of the time, confirming that synthetic-only training can yield practical structural health monitoring tools [24]. For concrete structures, Jayawickrema et al. used a nonlinear finite element model to generate 8 000 beam health states under four-point bending, varying rebar yield stress and bond conditions; an ANN trained purely on that FE dataset then classified the true physical beams' reinforcement yielding with 98% accuracy [25].

These successes rest on two key advantages of synthetic-data training. First, numerical pipelines allow exhaustive coverage of parameter spaces as rare damage modes or extreme loads can be simulated at will, ensuring networks are exposed to all relevant scenarios. Second, fully labeled data is produced automatically, removing the need for manual annotation or destructive testing. As a result, synthetic training can deliver arbitrarily large, balanced, and fault-free datasets without logistical constraints.

In summary, structural engineering studies now demonstrate that neural networks can learn damage detection, parameter identification, and health state classification directly from synthetic finite element-generated data, and then transfer that knowledge to real structures with high fidelity. This paradigm underpins modern data-driven frameworks, where simulation-based surrogate modeling augments or replaces experimental campaigns, significantly accelerating the development of robust ML tools for structural analysis and monitoring.

2.5 Hybrid Finite Element Modeling

Machine learning applications in civil engineering have been used so far to cover a wide range of problems. Many early successes have been in areas like structural health monitoring and image-based assessments rather than direct analytical modeling of structures. For example, computer vision algorithms powered by deep learning have been used for automatic crack detection and segmentation in infrastructure images, assisting inspectors in identifying damage on bridges and buildings [26]. Other studies leverage sensor data and time series to detect structural anomalies or predict failures [27] , using pattern recognition capabilities of ML.

These data-driven tasks treat the civil structure as a source of data to be classified or regressed upon and have shown significant promise in reducing manual effort and improving safety.

In contrast, using ML to perform or accelerate structural analysis (i.e., to compute structural response given loads, geometry, and material properties) is a more recent development. The finite element method (FEM) is the cornerstone of structural analysis; however, high-fidelity FEM simulations can be time-consuming for large or complex structures. There is growing interest in employing ML models either as surrogates that replace FEM calculations or as part of hybrid frameworks that integrate ML with FEM. The ultimate goal is to enhance computational efficiency and tackle problems where purely data-driven or purely physics-based methods alone may fall short.

2.5.1 Data-Driven Surrogate Models for Structural Analysis

Surrogate modeling refers to using a predictive ML model to emulate the input-output behavior of a physics-based simulator. In structural engineering, researchers have trained neural networks to approximate the results of FEM analyses, effectively creating fast-running approximators of structural response. These surrogates are typi-

cally trained on a database of simulations: a range of structures or loading scenarios is first analyzed with FEM to produce a training set, then a network learns the mapping from input parameters to outputs of interest. Once trained, the surrogate can predict results almost instantaneously, offering huge speed-ups for tasks like design space exploration, real-time monitoring, or uncertainty quantification.

Numerous examples of FEM surrogates have been reported. Liang et al. [28] presented a deep neural network that directly outputs the stress distribution in an aortic wall given the geometry and loading, bypassing the conventional FE solve. Their model achieved very high accuracy, with an average error of under 1% in peak stress prediction, compared to the FEM baseline, while computing results in a fraction of a second. Similarly, other studies have used multilayer perceptrons or convolutional neural networks to predict structural responses such as displacements, accelerations, or buckling loads, after training on simulation data specific to problem domains (e.g., buildings or shells under various load patterns) [29]. Chou et al. [30] recently introduced StructGNN: a graph neural network that can predict the static responses of frame buildings of varying heights. By representing the building as a graph (with nodes for joints and additional "story" nodes for floors) and training on numerous synthetic structures, their GNN surrogate achieved over 99% accuracy in predicting internal forces and deflections, generalizing better to taller structures than traditional neural networks.

These works demonstrate that, with sufficient training data encompassing the design space, an ML surrogate can closely replicate FEM calculations while being orders of magnitude faster at runtime.

However, pure data-driven surrogates also face challenges. They may require an extensive and representative simulation dataset, whose generation can be laborious. Moreover, if a query falls outside the distribution of the training data (for example, an unforeseen structural configuration or loading case), a black-box surrogate might extrapolate poorly. This is where incorporating physics (as discussed in Section 2.3) can be beneficial even for surrogate models. Researchers are beginning to blend data-driven and physics-informed ideas to create surrogates that are both fast and reliable within known physical bounds.

2.5.2 Hybrid Machine Learning–FEM Approaches

Beyond full replacements of FEM, another line of research seeks to combine ML algorithms with traditional FEM in a single framework, leveraging the strengths of each. In hybrid approaches, machine learning might handle the parts of the problem that are data-rich or computationally intensive. In contrast, FEM handles the parts best described by physics and for which robust solvers exist. One form of hybridization is the use of ML-generated components within a finite element simulation. For instance, Logarzo et al. [31] introduced smart constitutive laws, where a neural network is trained on high-fidelity microscale simulations to serve as the material model within a macroscale FEM analysis. This allowed them to capture complex inelastic material

behavior without repeatedly solving expensive micro-level models. The resulting hybrid model could then be embedded into a standard FE solver, dramatically speeding up multiscale analysis while maintaining accuracy in the macroscopic structural response.

Another class of hybrid methods involves using ML to assist in finite element model updating and calibration. Finite element model updating is the process of adjusting a simulation model's parameters, such as stiffness properties or boundary conditions, so that its outputs match measured real-world data from the structure. Traditional model calibration can be formulated as an optimization problem and may require many iterations of FEM runs. By introducing machine learning, one can approach this inversely: train a network to learn the mapping from observed structural responses to the model parameters or damage indicators. For example, Lee et al. [27] combine a physics-based model updating step with a deep learning model to detect damage in structures. They first use measured vibration data to update an FE model of the healthy structure, then generate many damage-case simulations with that calibrated model to train a neural network that can identify damage location and severity from new sensor inputs.

This hybrid strategy leverages FEM to generate physics-consistent data for training and utilizes ML to solve the complex inverse problem of damage identification efficiently. More generally, such approaches suggest how ML and FEM can be iteratively coupled: FEM provides physically grounded data or constraints to guide ML, while ML provides speed or flexibility to tackle inverse problems, large solution spaces, and noisy data.

In summary, hybrid ML-FEM models strive to leverage the advantages of both methodologies. They maintain fidelity to physical laws and provide interpretability while exploiting data-driven learning to reduce computation time and handle uncertainties or incomplete information. As the field of civil engineering advances toward digital twinning and the rapid assessment of structural systems, these hybrid methods are becoming increasingly pertinent. They allow engineers to incorporate real-time data and predictive ML into classical analysis workflows, potentially enabling faster post-hazard evaluations, continuous model calibration with sensor feeds, and more robust designs. The developments reviewed in this chapter underscore the thesis motivation: by infusing machine learning with physics domain knowledge and by intelligently coupling it with proven simulation tools, one can tackle structural analysis challenges that are intractable by purely conventional means or purely data-driven means alone.

3

Solving the Inverse Problem of a Simple Truss

3.1 The Linear Truss Structure

3.1.1 Definition

A truss is a structural assembly of straight, slender members connected at their ends to form a stable lattice, typically composed of triangular units. In a planar truss, each member carries only axial force, either tension or compression, and the joints are modeled as frictionless pins, allowing for free rotation.

The geometry and properties of a planar truss are fully specified by:

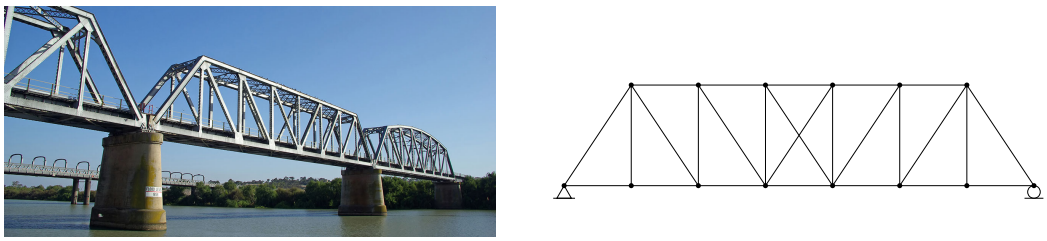
- **Nodes** N: an $n \times 2$ matrix of coordinates, where each row gives the (x, y) position of a node:
$$\mathrm{N} = \begin{bmatrix} c_{x,1} & c_{y,1} \\ \vdots & \\ c_{x,n} & c_{y,n} \end{bmatrix} \in \mathbb{R}^{n \times 2}. \tag{3.1}$$
- **Members** M: an $m \times 2$ matrix listing the start and end node indices of each member:
$$\mathrm{M} = \begin{bmatrix} s_1 & e_1 \\ \vdots & \\ s_m & e_m \end{bmatrix} \in \mathbb{N}^{m \times 2}. \tag{3.2}$$
Here, s_i and e_i denote the indices of the start and end nodes of member i.
- **Cross-sectional areas** $\mathbf{A} = \begin{bmatrix} A_1, \ldots, A_m \end{bmatrix}^\top \in \mathbb{R}^{|\mathrm{M}|}$, where A_i is the area of member i.
- **Young's moduli** $\mathbf{E} = \begin{bmatrix} E_1, \ldots, E_m \end{bmatrix}^\top \in \mathbb{R}^{|\mathrm{M}|}$, where E_i is the modulus of member i.
- **Support conditions** $S = \{\, i : u_i = 0 \}$, the set of fixed degrees of freedom (indices i for which the displacement u_i is zero). Each node has two degrees of freedom:

one translation in the x direction and one translation in the y direction.

Equivalently, a truss can be viewed as an undirected graph $G(\mathrm{N}, \mathrm{M})$, with edge attributes A_i, E_i, and node attributes given by coordinates and support constraints.

3.1.2 Representation

A truss structure is represented by plotting its nodes at the specified coordinates and drawing members as straight lines between the corresponding node pairs (Figure 3.1). Concentrated loads at a node are indicated by arrows emanating from that node.



(**a**) *Photograph of a truss* [32] (**b**) *Schematic of the same truss*

Figure 3.1: *Real versus schematic representation of a planar truss.*

Support conditions follow standard engineering symbols (Figure 3.2): a single fixed degree of freedom is qualified as a rolling support, shown by superposing the node on a circle, while fixing both degrees of freedom is denoted as a pinned support and represented by a triangle.

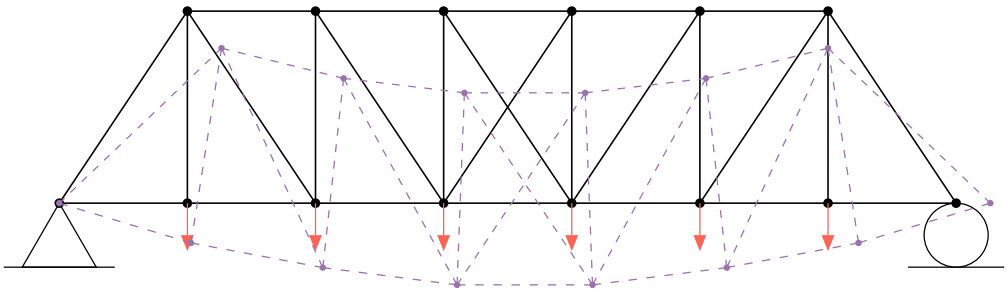


Figure 3.2: *Deformed truss with a pinned support (left) and a rolling support (right).*

Additional properties, such as cross–sectional areas, material parameters, and nodal loads, are typically described in accompanying tables or text. When illustrating deformation under load, the deformed shape is sometimes overlaid using dashed lines.

3.1.3 Properties

In general, a structure's members resist three primary internal forces:

- **Axial force** F**:** Forces which stretches or compresses members along their length;
- **Shear force** V**:** Forces acting perpendicular to the member axis;
- **Bending moment** M**:** Moment causing members to bend or rotate.

A planar truss has exactly two translational degrees of freedom per node (horizontal and vertical) and no rotational stiffness. Consequently, the bending moment vanishes:

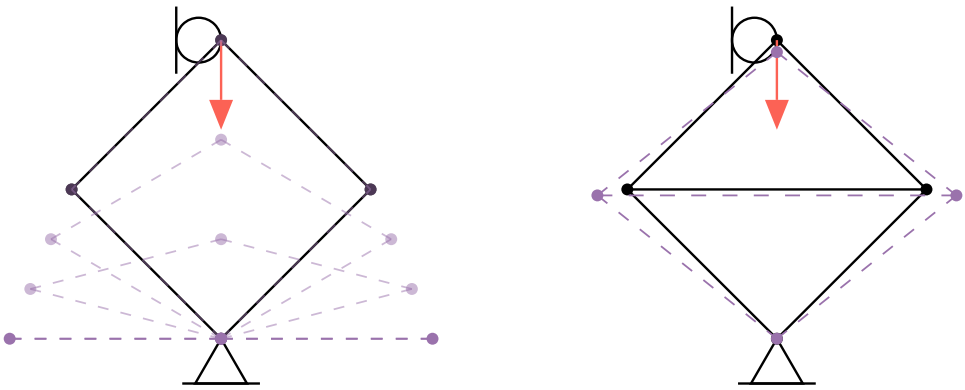
$$M = 0. \tag{3.3}$$

Since shear is the spatial derivative of moment,

$$\frac{dM}{dx} = V \quad \Longrightarrow \quad \frac{d0}{dx} = V = 0, \tag{3.4}$$

All internal equilibrium is thus carried by axial force alone. Therefore, each member is fully characterized by its cross-sectional area A_i and Young modulus E_i.

This axial-only load path imposes a geometric requirement: the truss must be triangulated to remain statically determinate. Without triangulation, specific member configurations become mechanisms that fail to maintain equilibrium under load, as shown in Figure 3.3.



(a) *Untriangulated (unstable) truss* **(b)** *Triangulated (stable) truss*

Figure 3.3: *Stable versus unstable truss configurations under axial loading.*

3.2 Solving a Truss Structure

Solving a truss entails computing the nodal displacements $\mathbf{u}$ and support reactions $\mathbf{r}$ under applied loads $\mathbf{q}_{\text{ext}}$, and subsequently deriving each member's axial force F.

Among the various computational approaches to solve structures [33], we adopt the *direct stiffness method*, a linear analysis technique for general frame structures. Its linearity rests on two core assumptions:

- **Material linearity:** The stress-strain relationship is assumed linear and described by Hooke's law,
$$E\,\varepsilon = \sigma = \frac{F}{A}\,, \tag{3.5}$$
- **Geometric linearity:** Displacements are assumed negligible, the structure geometry is assumed non-deformed through the application of the load.

Under these hypotheses, the equilibrium of the structure is expressed as

$$\mathbf{K}\,\mathbf{u} \;=\; \mathbf{q}_{\text{ext}} + \mathbf{r} \;=\; \mathbf{q}, \tag{3.6}$$

where:

- $\mathbf{K} \in \mathbb{R}^{2|\mathrm{N}|\times 2|\mathrm{N}|}$ is the global stiffness matrix (assembly detailed in Section 3.2.2) a matrix containing information on geometry and stiffness of the structure,
- $\mathbf{u} \in \mathbb{R}^{2|\mathrm{N}|}$ is the nodal displacement vector,
- $\mathbf{q} \in \mathbb{R}^{2|\mathrm{N}|}$ is the vector of applied nodal loads which is the sum of the external load $\mathbf{q}_{\text{ext}}$ and the support reactions $\mathbf{r}$.

Example

To illustrate the concept presented in this chapter, we will apply most procedures to the truss in Figure 3.4, using $EA = 525 \times 10^6$ N:

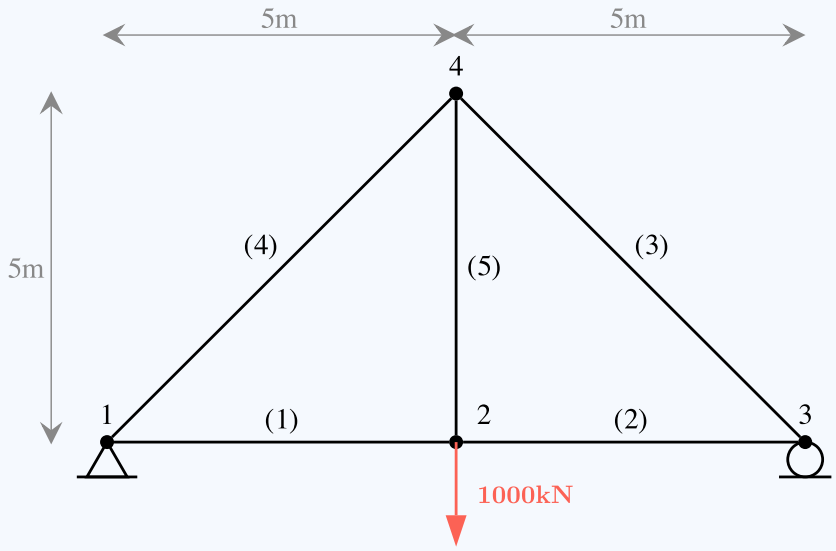


Figure 3.4: *Truss that will be studied through this chapter*

The displacement vector $\mathbf{u}$ stacks the two degrees of freedom per node in a consistent ordering:

$$\mathbf{u} = \begin{bmatrix} u_{1,x} \\ u_{1,y} \\ \vdots \\ u_{n,x} \\ u_{n,y} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{2n-1} \\ u_{2n} \end{bmatrix} \in \mathbb{R}^{2|\mathrm{N}|}, \tag{3.7}$$

and the load vector $\mathbf{q}$ is defined analogously,

$$\mathbf{q} = \begin{bmatrix} q_{1,x} \\ q_{1,y} \\ \vdots \\ q_{n,x} \\ q_{n,y} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{2n-1} \\ q_{2n} \end{bmatrix} \in \mathbb{R}^{2|\mathrm{N}|}. \tag{3.8}$$

3.2.1 Computing the members' stiffness matrix

The global stiffness matrix $\mathbf{K}$ is assembled from each member's contribution to the global stiffness. We begin by calculating the stiffness matrix of a single member aligned with its local axis, see Figure 3.5.

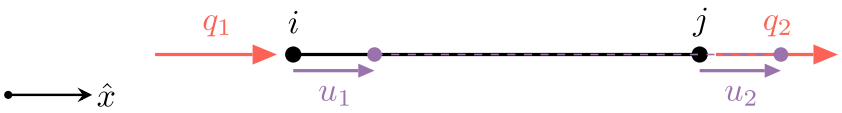


Figure 3.5: *Local element displacements and corresponding forces*

In local coordinates, the element stiffness is

$$\mathbf{k}_{\mathrm{loc}} = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{3.9}$$

To express this in the global coordinate system (i.e., the one from the structure), we map each member and its degrees of freedom from the local to the global coordinate system. Illustrated in Figure 3.6, this transformation is two-fold:

- Expand local (u_1, u_2) to global $(u_{1,x}, u_{1,y}, u_{2,x}, u_{2,y})$ denoted by (u_1, u_2, u_3, u_4);
- Rotate the member by an angle θ.

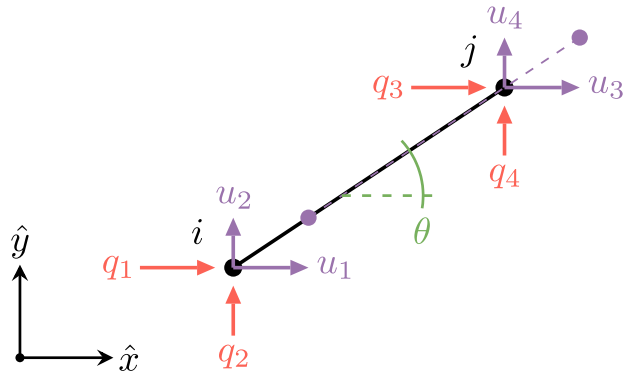


Figure 3.6: *Global element displacements and orientation*

The transformation yields

$$\mathbf{k}_{\text{global}} = \mathbf{R}(\theta)\,\mathbf{k}_{\text{loc}}\,\mathbf{R}(\theta)^{\top}, \quad \mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & \cos\theta \\ 0 & \sin\theta \end{bmatrix}. \tag{3.10}$$

Substituting $\cos\theta = c$ and $\sin\theta = s$ gives the explicit form:

$$\mathbf{k}_{\text{global}} = \frac{EA}{L} \begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix}. \tag{3.11}$$

Example

Applying (3.11) to each member of the example truss (Figure 3.4) yields the global member stiffness matrices summarized in Table 3.1.

Member	Length	Angle	$\mathbf{k}_{\text{global}}$
1	5	0°	$\frac{EA}{5}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
2	5	0°	$\frac{EA}{5}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
3	$5\sqrt{2}$	-45°	$\frac{EA}{5\sqrt{2}}\begin{bmatrix} 0.5 & -0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}$
4	$5\sqrt{2}$	45°	$\frac{EA}{5\sqrt{2}}\begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \end{bmatrix}$
5	5	90°	$\frac{EA}{5}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$

Table 3.1: *Global stiffness matrix of each member*

3.2.2 Assembling the structure stiffness matrix

Each element connecting nodes i and j yields a 4×4 stiffness matrix $\mathbf{k}_{\text{global}}$, which we partition into four 2×2 subblocks $\mathbf{B}_{kl}$

$$\mathbf{k}_{global} = \frac{EA}{L} \left[\begin{array}{cc:cc} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ \hdashline -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{array}\right] = \left[\begin{array}{c:c} \mathbf{B}_{i,i} & \mathbf{B}_{i,j} \\ \hdashline \mathbf{B}_{j,i} & \mathbf{B}_{j,j} \end{array}\right] \tag{3.12}$$

By exploiting the block-matrix abstraction introduced in equation 3.12, we can systematically and efficiently assemble the global stiffness matrix. We begin by allocating an initially zeroed square matrix $\mathbf{K} \in \mathbb{R}^{2|\mathrm{N}| \times 2|\mathrm{N}|}$. Conceptually, $\mathbf{K}$ is partitioned into a grid of 2×2 subblocks, which we denote by $\mathbf{K}_{[k,l]}$, each corresponding in size to the subblocks $\mathbf{B}_{k,l}$ of a single element's $\mathbf{k}_{\text{global}}$.

For every member that connects nodes i and j, we extract its four block contributions $\mathbf{B}_{i,i}$, $\mathbf{B}_{i,j}$, $\mathbf{B}_{j,i}$, and $\mathbf{B}_{j,j}$. These blocks are then added into the appropriate submatrices of $\mathbf{K}$. In practice, this means locating the global block indices associated with node i (rows and columns $2i-1$ and $2i$) and node j (rows and columns $2j-1$ and $2j$), and accumulating the element-level stiffness:

$$\begin{aligned} \mathbf{K}_{[i,i]} &\longleftarrow \mathbf{K}_{[i,i]} + \mathbf{B}_{i,i} \\ \mathbf{K}_{[i,j]} &\longleftarrow \mathbf{K}_{[i,j]} + \mathbf{B}_{i,j} \\ \mathbf{K}_{[j,i]} &\longleftarrow \mathbf{K}_{[j,i]} + \mathbf{B}_{j,i} \\ \mathbf{K}_{[j,j]} &\longleftarrow \mathbf{K}_{[j,j]} + \mathbf{B}_{j,j} \end{aligned} \tag{3.13}$$

For instance, if a member spans nodes 2 and 5, its block matrices $\mathbf{B}_{2,2}$, $\mathbf{B}_{2,5}$, $\mathbf{B}_{5,2}$, and $\mathbf{B}_{5,5}$ are added into the corresponding positions $\mathbf{K}_{[2,2]}$, $\mathbf{K}_{[2,5]}$, $\mathbf{K}_{[5,2]}$, and $\mathbf{K}_{[5,5]}$. Repeating this procedure for every element yields the fully assembled $\mathbf{K}$ that governs the global equilibrium.

Finally, we enforce the support conditions by recognizing that certain degrees of freedom are fixed. We know that $u_i = 0, \forall i \in S$. Accordingly, we partition the fully assembled system

$$\mathbf{K}\mathbf{u} = \left[\begin{array}{c:c} \mathbf{K}_{aa} & \mathbf{K}_{ab} \\ \hdashline \mathbf{K}_{ba} & \mathbf{K}_{bb} \end{array}\right] \left[\begin{array}{c} \mathbf{u}_a \\ \hdashline \mathbf{u}_b \end{array}\right] = \left[\begin{array}{c} \mathbf{q}_a \\ \hdashline \mathbf{q}_b \end{array}\right] = \left[\begin{array}{c} \mathbf{q}_{\text{ext},a} \\ \hdashline \mathbf{q}_{\text{ext},b} \end{array}\right] + \left[\begin{array}{c} \mathbf{r}_a \\ \hdashline \mathbf{r}_b \end{array}\right] \tag{3.14}$$

where the subscript a indicates the fixed degrees of freedom and b the remaining unknowns. Imposing $\mathbf{u}_a = \mathbf{0}$ and noting that reactions appear only in $\mathbf{q}_a$, since there are no supports in b, reduces the problem to two coupled subsystems:

$$\mathbf{K}_{bb}\mathbf{u}_b = \mathbf{q}_{\text{ext},b} \tag{3.15}$$

$$\mathbf{K}_{ab}\mathbf{u}_b = \mathbf{q}_{\text{ext},a} + \mathbf{r_b} \tag{3.16}$$

Equation (3.15) is solved first to obtain the unknown displacements $\mathbf{u}_b$. Once $\mathbf{u}_b$ is known, the support reactions $\mathbf{r}_a$ follow directly from (3.16).

Example

To demonstrate the assembly procedure, we return to the four-node truss of Figure 3.4. First, we allocate an 8×8 zero matrix for $\mathbf{K}$. Next, using each member's global stiffness blocks from Table 3.1, we accumulate their contributions into $\mathbf{K}$ according to the node pairs:

$$\mathbf{K} = \begin{bmatrix} \left(\mathbf{B}^{(1)} + \mathbf{B}^{(4)}\right)_{1,1} & \mathbf{B}^{(1)}_{1,2} & \mathbf{0} & \mathbf{B}^{(4)}_{1,4} \\ \mathbf{B}^{(1)}_{2,1} & \left(\mathbf{B}^{(1)} + \mathbf{B}^{(2)} + \mathbf{B}^{(5)}\right)_{2,2} & \mathbf{B}^{(2)}_{2,3} & \mathbf{B}^{(5)}_{2,4} \\ \mathbf{0} & \mathbf{B}^{(2)}_{3,2} & \left(\mathbf{B}^{(2)} + \mathbf{B}^{(3)}\right)_{3,3} & \mathbf{B}^{(3)}_{3,4} \\ \mathbf{B}^{(4)}_{4,1} & \mathbf{B}^{(5)}_{4,2} & \mathbf{B}^{(3)}_{4,3} & \left(\mathbf{B}^{(3)} + \mathbf{B}^{(4)} + \mathbf{B}^{(5)}\right)_{4,4} \end{bmatrix} \tag{3.17}$$

With the notation $\mathbf{B}^{(i)}$ indicating the block matrix form of the stiffness matrix of member i.

Expanding this result yields the full stiffness matrix. Observing that degrees of freedom 1, 2, and 6 are restrained, we partition $\mathbf{K}$ into $\mathbf{K}_{aa}$ (in red), $\mathbf{K}_{ab}$ (in blue), $\mathbf{K}_{ba}$ (in green), and $\mathbf{K}_{bb}$ (the remaining of the matrix):

$$K = EA \begin{bmatrix} 0.2 + \frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & -0.2 & 0 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ \frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & 0 & 0 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ -0.2 & 0 & 0.4 & 0 & -0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & -0.2 \\ 0 & 0 & -0.2 & 0 & 0.2 + \frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} & 0 & 0 & -\frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & \frac{0.2}{\sqrt{2}} & 0 \\ -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} & 0 & -0.2 & \frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} & 0 & 0.2 + \frac{0.2}{\sqrt{2}} \end{bmatrix} \tag{3.18}$$

From equations 3.15 and 3.16 we obtain two systems of equations :

$$EA \begin{bmatrix} 0.4 & 0 & -0.2 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & -0.2 \\ -0.2 & 0 & 0.2 + \frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} \\ 0 & 0 & -\frac{0.1}{\sqrt{2}} & \frac{0.2}{\sqrt{2}} & 0 \\ 0 & -0.2 & \frac{0.1}{\sqrt{2}} & 0 & 0.2 + \frac{0.2}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \\ u_5 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} 0 \\ -1000 \times 10^3 \\ 0 \\ 0 \\ 0 \end{bmatrix} [N] \tag{3.19}$$

$$EA \begin{bmatrix} -0.2 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ 0 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ 0 & 0 & -\frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \\ u_5 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r_1 \\ r_2 \\ r_6 \end{bmatrix} [N] \tag{3.20}$$

3.2.3 Solving for Displacements and Support Reactions

In its unconstrained form, the global stiffness matrix is only positive semi-definite. Introducing support conditions yields the reduced matrix $\mathbf{K}_{bb}$, which is positive definite and therefore nonsingular [34]. Consequently, the unknown displacement vector $\mathbf{u}_b$ is uniquely determined by

$$\mathbf{u}_b \;=\; \mathbf{K}_{bb}^{-1}\,\mathbf{q}_b\,. \tag{3.21}$$

In practice, direct inversion is avoided; instead, specialized solvers are used that exploit symmetry and sparsity to achieve efficiency and numerical stability.

With $\mathbf{u}_b$ in hand, the reaction forces at the supported degrees of freedom follow from the equation 3.16:

$$\mathbf{r}_a \;=\; \mathbf{K}_{ab}\,\mathbf{u}_b \;-\; \mathbf{q}_{\text{ext},a}\,. \tag{3.22}$$

Example

By applying equation 3.21, the deformation vector $\mathbf{u}_b$ can be explicitly determined. For illustrative purposes, the inverse of the stiffness matrix is computed in this example:

$$\mathbf{u}_b = \frac{10^{-6}}{525} \begin{bmatrix} 5. & -2.5 & 5. & 2.5 & -2.5 \\ -2.5 & 14.571 & -5. & -2.5 & 9.571 \\ 5. & -5. & 10. & 5. & -5. \\ 2.5 & -2.5 & 5. & 9.571 & -2.5 \\ -2.5 & 9.571 & -5. & -2.5 & 9.571 \end{bmatrix} \begin{bmatrix} 0 \\ -1000\text{e}3 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 4.761 \\ -27.754 \\ 9.524 \\ 4.762 \\ -18.231 \end{bmatrix} \times 10^{-3} \tag{3.23}$$

Subsequently, the reaction forces at the supports $\mathbf{r_a}$ are calculated using equation 3.22:

$$\mathbf{r_a} = 525 \times 10^6 \begin{bmatrix} -0.2 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ 0 & 0 & 0 & -\frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \\ 0 & 0 & -\frac{0.1}{\sqrt{2}} & \frac{0.1}{\sqrt{2}} & -\frac{0.1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4.761 \\ -27.754 \\ 9.524 \\ 4.762 \\ -18.231 \end{bmatrix} \times 10^{-3} = \begin{bmatrix} 0 \\ 500 \\ 500 \end{bmatrix} \times 10^3 \tag{3.24}$$

Figure 3.7 visualises the displacement and reaction obtained from the structural analysis. The deformations have been scaled by a factor of 50 for clarity.

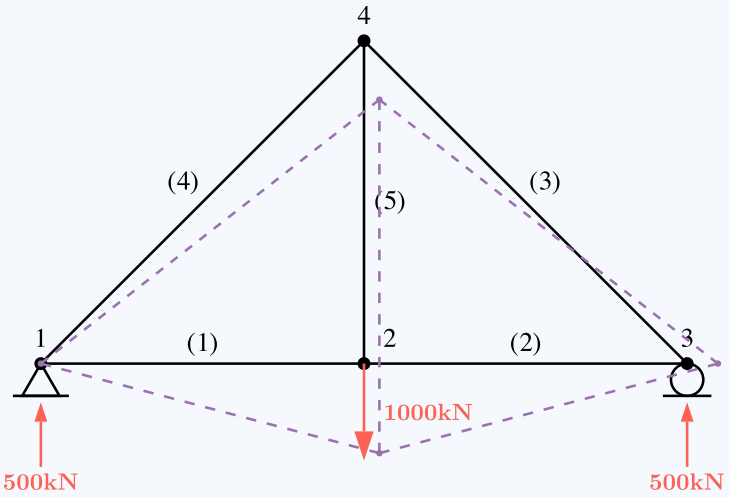


Figure 3.7: *Resulting displacements on the structure scaled by 50.*

3.2.4 Recovering Member Forces

After obtaining the nodal displacements $\mathbf{u}$, we compute each member's axial force F via Hooke's law:

$$\sigma = E\,\varepsilon = \frac{F}{A} \quad \Longrightarrow \quad F = E\,A\,\varepsilon, \tag{3.25}$$

where σ is the axial stress and ε the axial strain.

The remaining task is to express the strain in terms of the known displacements at the element's end-nodes. Although one could use the exact definition

$$\varepsilon = \frac{\Delta L}{L} \tag{3.26}$$

Our linearized (small-deformation) framework employs the projected displacement difference, which does not consider the variation in the geometry under loading:

$$\varepsilon = \frac{(\mathbf{u}_j - \mathbf{u}_i)\cdot \widehat{\mathbf{l}}}{L}, \tag{3.27}$$

where $\widehat{\mathbf{l}}$ is the unit vector along the member from node i to node j, and L its undeformed length. Substituting (3.27) into (3.25) yields

$$F \;=\; E\,A\,\frac{(\mathbf{u}_j - \mathbf{u}_i)\cdot \widehat{\mathbf{l}}}{L} \tag{3.28}$$

.

Example

The internal force in each structural member is computed using equation 3.28, with the results presented in Table 3.2. A negative value of F indicates that the member is in compression, whereas a positive value signifies traction.

Member	EA	L	$\widehat{\mathbf{l}}$	$\mathbf{u}_i$	$\mathbf{u}_j$	F
1	525×10^6	5	$\begin{bmatrix} 1.000 \\ 0.000 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \end{bmatrix}$	$\begin{bmatrix} 0.005 \\ -0.028 \end{bmatrix}$	500
2	525×10^6	5	$\begin{bmatrix} 1.000 \\ 0.000 \end{bmatrix}$	$\begin{bmatrix} 0.005 \\ -0.028 \end{bmatrix}$	$\begin{bmatrix} 0.010 \\ 0.000 \end{bmatrix}$	500
3	525×10^6	7.071	$\begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}$	$\begin{bmatrix} 0.010 \\ 0.000 \end{bmatrix}$	$\begin{bmatrix} 0.005 \\ -0.018 \end{bmatrix}$	−707.11
4	525×10^6	7.071	$\begin{bmatrix} -0.707 \\ -0.707 \end{bmatrix}$	$\begin{bmatrix} 0.005 \\ -0.018 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \end{bmatrix}$	−707.11
5	525×10^6	5	$\begin{bmatrix} 0.000 \\ -1.000 \end{bmatrix}$	$\begin{bmatrix} 0.005 \\ -0.018 \end{bmatrix}$	$\begin{bmatrix} 0.005 \\ -0.028 \end{bmatrix}$	1000

Table 3.2: *Axial force in each structural member.*

Figure 3.8 illustrates the distribution of forces within the structure. By convention, members in compression are depicted in red, while those in traction are shown in blue.

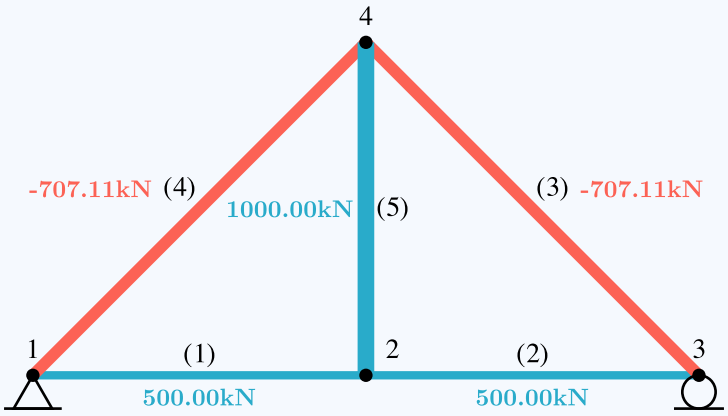


Figure 3.8: *Distribution of axial forces in the truss structure. Red indicates compression, and blue indicates traction.*

3.3 The inverse problem of structural analysis

In structural engineering, the most common scenario involves solving the forward problem: computing the response of a structure under a prescribed set of loads and boundary conditions. This process assumes that all structural parameters, such as ge-

ometry and material properties, are known a priori, either specified by the design team or mandated by regulatory standards such as the Eurocodes [35].

However, in the context of structural assessment, particularly during renovation works or after unexpected damage, engineers often face the inverse problem. In this case, structural parameters are unknown and must be inferred from observed data. Specifically, given a known set of loads and the resulting measured deformations, the objective is to estimate the stiffness matrix of the structure.

For truss structures, the geometry is typically known. Therefore, the primary unknowns are the axial rigidities EA of individual members. This section presents methods for solving the inverse problem.

3.4 Analytical Solution to the Inverse Problem

Consider the standard constitutive equation:

$$\mathbf{K}\mathbf{u} = \mathbf{q} \tag{3.29}$$

This system contains $2|\mathrm{N}|$ equations, where $|\mathrm{N}|$ denotes the number of nodes in the structure. As established in Section 3.2.2, the application of support conditions reduces the system to:

$$\mathbf{K}_{bb}\mathbf{u}_b = \mathbf{q}_{\mathrm{ext},b} \tag{3.30}$$

This reduced system contains $2|\mathrm{N}| - |S|$ equations, where $|S|$ is the number of fixed degrees of freedom.

The stiffness matrix $\mathbf{K}$ is determined solely by the structural geometry and material properties, and is therefore considered constant. By applying multiple distinct loading configurations $\mathbf{q}_i$ and recording the corresponding displacements $\mathbf{u}_i$, the following set of systems can be constructed:

$$\begin{cases} \mathbf{K}_{bb}\mathbf{u}_{b,1} = \mathbf{q}_{b,1} \\ \vdots \\ \mathbf{K}_{bb}\mathbf{u}_{b,n} = \mathbf{q}_{b,n} \end{cases} \tag{3.31}$$

Provided the total number of independent equations exceeds $|\mathrm{M}|$, the number of unknown axial rigidities (one per member), the system is theoretically solvable.

Although feasible for small structures, solving this system manually becomes impractical as the number of members increases. Symbolic computation techniques can be employed to obtain exact solutions in such cases.

Example

Building on the previously analysed truss, the reduced stiffness matrix $\mathbf{K}_{bb}$ can

be constructed under the assumption that the axial rigidity EA_i of each member is unknown:

$$\mathbf{K}_{bb} = \begin{bmatrix} \frac{EA_0+EA_1}{5} & 0 & -\frac{EA_1}{5} & 0 & 0 \\ 0 & \frac{EA_4}{5} & 0 & 0 & -\frac{EA_4}{5} \\ -\frac{EA_1}{5} & 0 & \frac{EA_1}{5} + \frac{EA_2}{\sqrt{2}} & -\frac{EA_2}{\sqrt{2}} & \frac{EA_2}{\sqrt{2}} \\ 0 & 0 & -\frac{EA_2}{\sqrt{2}} & \frac{EA_2+EA_3}{\sqrt{2}} & \frac{EA_3-EA_2}{\sqrt{2}} \\ 0 & -\frac{EA_4}{5} & \frac{EA_2}{\sqrt{2}} & \frac{EA_3-EA_2}{\sqrt{2}} & \frac{EA_2+EA_3}{\sqrt{2}} + \frac{EA_4}{5} \end{bmatrix} \tag{3.32}$$

This matrix formulation yields a linear system of five equations. Given that the truss contains five members, there are five unknowns, one per axial rigidity. Hence, a single measurement of displacement $\mathbf{u}$ under a known load $\mathbf{q}$ suffices to determine the system completely.

Suppose the structure is subjected to a vertical point load of $1000\,\mathrm{kN}$ at node (2), as in the preceding example. On-site measurement of the resulting displacements provides the following data:

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 4.761 \\ -27.754 \\ 9.524 \\ 0 \\ 4.762 \\ -18.231 \end{bmatrix} \times 10^{-3}[m] \tag{3.33}$$

This leads to a linear system of the form $A\mathbf{x} = \mathbf{b}$, where the coefficients depend linearly on the unknown axial rigidities EA_i. Substituting the known displacements into the stiffness relation results in:

$$\begin{aligned} \mathbf{K}_{bb}\mathbf{u}_b = 10^{-4} \begin{bmatrix} 9.524EA_0 - 9.524EA_1 \\ -19.05EA_4 \\ 9.524EA_1 - 9.524EA_2 \\ 9.524EA_2 - 9.524EA_3 \\ -9.524EA_2 - 9.524EA_3 + 19.05EA_4 \end{bmatrix} &= \begin{bmatrix} 0 \\ -1000 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times 10^3 \\ \Leftrightarrow 10^{-4} \begin{bmatrix} 9.524 & -9.524 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -19.05 \\ 0 & 9.524 & -9.524 & 0 & 0 \\ 0 & 0 & 9.524 & -9.524 & 0 \\ 0 & 0 & -9.524 & -9.524 & 19.05 \end{bmatrix} \begin{bmatrix} EA_0 \\ EA_1 \\ EA_2 \\ EA_3 \\ EA_4 \end{bmatrix} &= \begin{bmatrix} 0 \\ -1000 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times 10^3 \end{aligned} \tag{3.34}$$

Solving this system yields the unique solution $EA_i = 525 \times 10^6\,\mathrm{N}$ for all i, consistent with the true member properties.

While the analytical approach offers a direct and interpretable method for solving the inverse problem, it is not without limitations. The first significant issue arises when the system of equations is ill-posed, rendering the solution indeterminate. This occurs, for instance, when certain members experience zero strain under loading, thereby providing no information about their stiffness.

Example

Consider a modification of the previous structure in which two pinned supports are introduced instead of one, as depicted in Figure 3.9. This alteration results in an indeterminate system that remains unsolvable, regardless of the number of independent measurements $\mathbf{u}^{(i)}$ under known loads $\mathbf{q}^{(i)}$.

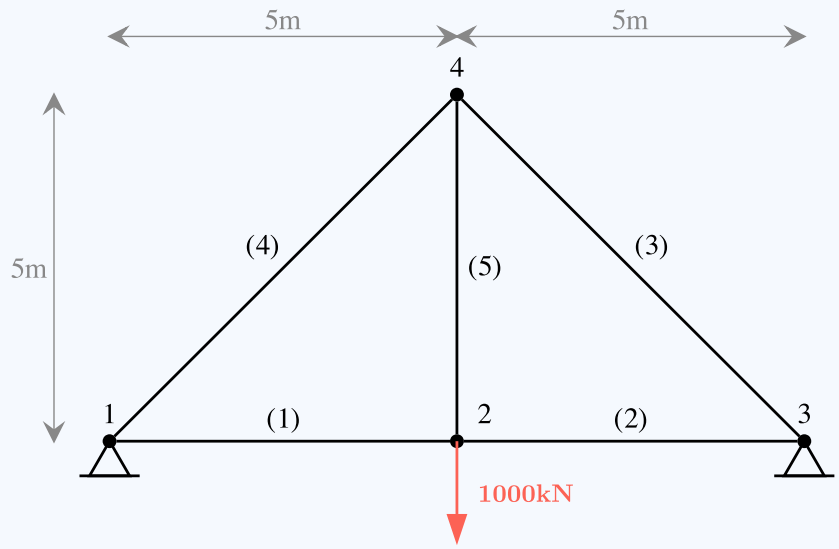


Figure 3.9: *Modified truss with two pinned supports, leading to an indeterminate system.*

In this configuration, members (1) and (2) are colinear and connected by node (2). A rightward displacement of node (2) induces extension in member (1) and compression in member (2). However, the displacements alone are insufficient to distinguish between the two effects. The average axial rigidity governs the overall movement, $0.5(EA_1 + EA_2)$, rendering the individual values of EA_1 and EA_2 unidentifiable from the available data.

The second limitation of the analytical method lies in its sensitivity to noise, which is a direct consequence of the precision demands embedded in the underlying mathematical and physical formulations.

In practice, particularly when assessing damaged or aging structures, obtaining highly accurate displacement measurements is often infeasible. Cost constraints, accessibility challenges, and the complexity of instrumentation installation all contribute to this difficulty. For instance, the use of unmanned aerial vehicles (UAVs) as a cost-effective method for monitoring deflections in concrete bridges typically results in measurement errors in the range of $\pm 1\%$ to $\pm 5\%$ [36, 37]. According to technical guidance, more traditional measurement approaches, such as laser-based measurement, can achieve tolerances below 2%, although deviations up to 5% are reported in complex structural environments [38].

Given this inherent uncertainty, the analytical method is unable to predict axial rigidities within acceptable error margins. Even small perturbations in the input data can lead to significant deviations in the computed axial rigidity values, thereby undermining the reliability of the inverse analysis in real-world applications.

To demonstrate the sensitivity of the analytical method to measurement noise, we will study the example truss in Figure 3.4. Consider the vectors $\widetilde{\mathbf{q}}$ and $\widetilde{\mathbf{u}}$, representing the applied load and the measured structural displacements, respectively, both perturbed by a multiplicative noise factor α:

$$\widetilde{\mathbf{u}} = \boldsymbol{\alpha}_u \odot \mathbf{u} \tag{3.35}$$

$$\widetilde{\mathbf{q}} = \boldsymbol{\alpha}_q \odot \mathbf{q} \tag{3.36}$$

With $\odot$, the elementwise product operator and the noise α modelled as a random variable following a normal distribution, parameterised by a noise level ϵ:

$$\alpha_i \sim \mathcal{N}(\mu = 1, \sigma = \frac{\epsilon}{2}) \tag{3.37}$$

The parameter ϵ corresponds to the total noise amplitude. Under this model, the 95% confidence interval for α is given by $\mu \pm \frac{\epsilon}{2}$. For instance, with $\epsilon = 0.1$, the noise factor α lies within the range $[0.9, 1.1]$ with 95% confidence, representing a $\pm 10\%$ error margin.

Applying this noise model to the previous example yields:

$$\widetilde{\mathbf{q}} = \boldsymbol{\alpha}_q \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1000 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times 10^3\,[\mathrm{N}], \quad \widetilde{\mathbf{u}} = \boldsymbol{\alpha}_u \odot \begin{bmatrix} 0 \\ 0 \\ 4.761 \\ -27.754 \\ 9.524 \\ 0 \\ 4.762 \\ -18.231 \end{bmatrix} \times 10^{-3}\,[\mathrm{m}] \tag{3.38}$$

Using these perturbed values, the inverse problem is solved to estimate the axial

rigidity $\widehat{EA}_i$ of each member. The accuracy of these estimates is quantified using the Mean Absolute Percentage Error (MAPE), defined as:

$$\text{MAPE}(\widehat{\mathbf{x}}, \mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\widehat{x}_i - x_i}{x_i} \right| \tag{3.39}$$

To ensure statistical robustness, this process is repeated 100 times with different noise per level ϵ. The resulting average MAPE values are shown in Figure 3.10.

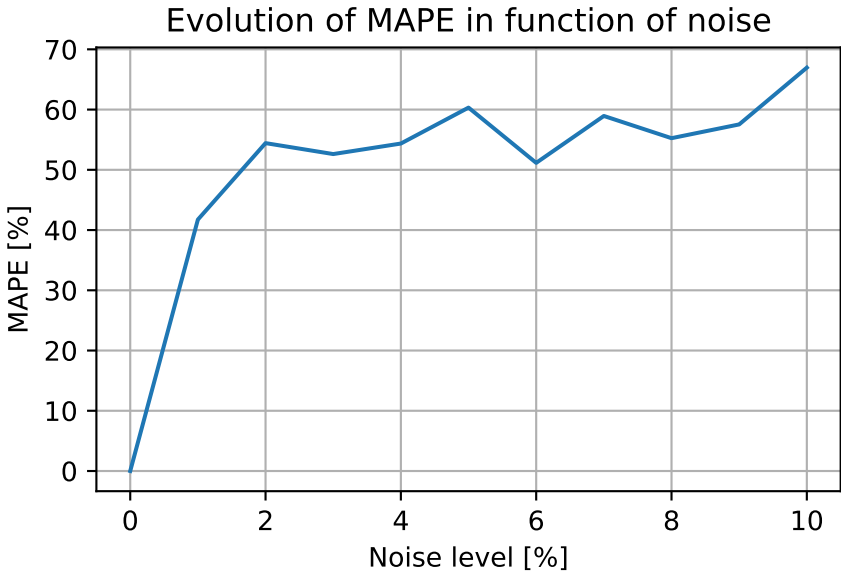


Figure 3.10: *Evolution of MAPE of the analytical method as a function of noise level, with 100 trials per level.*

The results clearly indicate that even low levels of measurement noise can lead to significant errors in the estimated axial rigidities. This confirms that the analytical method is highly sensitive to perturbations in input data and is therefore unsuitable in contexts where noise cannot be rigorously controlled.

3.5 Traditional Machine Learning Techniques

3.5.1 Dataset and preprocessing

To enhance the robustness of inverse predictions, this section introduces several classical machine learning methods that are, by design, more tolerant to measurement noise [8]. We begin by assuming the availability of a *noiseless* dataset consisting of structural data for a specific class of structures. This dataset contains 4096 various examples of the reference structures with different loads and axial rigidities. The composition of this dataset is illustrated in Figure 3.11, and comprises:

- node displacements, $\mathbf{u}$,
- applied loads, $\mathbf{q}$,
- member strains, $\boldsymbol{\varepsilon}$,

- axial rigidities of the members, EA.

The noiseless property of our training dataset ensures that the model is trained on exact data and thus captures the actual underlying rules of structural analysis. The model will then be used to predict axial rigidities from noisy input.

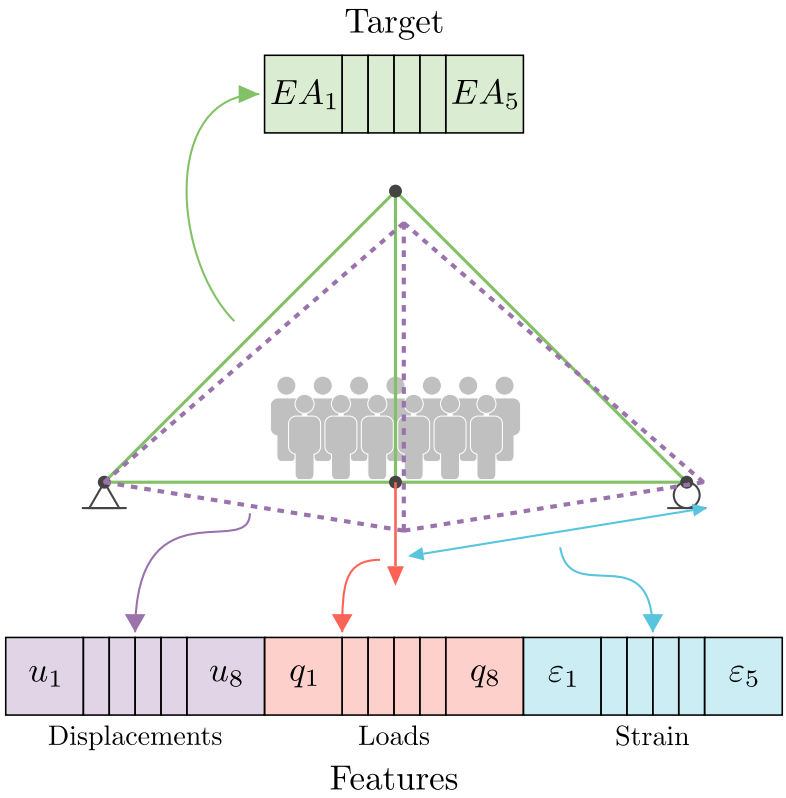


Figure 3.11: *Illustration of the dataset composition.*

The origin and generation methodology of this dataset are described in detail in Chapter 4. For the remainder of this chapter, we treat $\mathbf{u}$, $\mathbf{q}$, and $\boldsymbol{\varepsilon}$ as the input features, denoted $\mathbf{X}$, used to predict the target variable $\mathbf{Y}$, which corresponds to the member axial rigidities EA.

To optimise model learning performance, the input features undergo a preprocessing pipeline comprising three primary steps: feature filtering, feature standardisation, and dimensionality reduction.

Feature Filtering

In the context of supervised learning, not all features contribute meaningful information to the learning process. In particular, features that exhibit no variation across the dataset (i.e., those with constant values) can be safely excluded. A typical example in structural mechanics involves the displacements associated with fixed degrees of freedom, which, by definition, are identically zero throughout all samples.

Such features are uninformative, as they do not correlate with the target variable and offer no discriminative power for the learning algorithm. Including them may

hinder model performance by introducing unnecessary complexity and diluting the impact of relevant features.

Therefore, as a preprocessing step, constant features are systematically removed from the feature set. This filtering improves the model's efficiency and learning capacity by reducing dimensionality and focusing on informative patterns in the data.

Feature Standardisation

Under the assumption that input features follow a normal distribution with mean μ and standard deviation σ, we transform each feature to a standard normal distribution with zero mean and unit variance. This standardisation ensures that all features share a common scale, mitigating issues arising from large differences in magnitude. For example, nodal loads expressed in $[N]$ are typically on the order of 10^5, while displacements in $[m]$ are generally of the order 10^{-3}.

Without standardisation, the model may overemphasise large-magnitude features, thereby distorting the learning process. The transformation is defined as follows:

$$Z = \frac{X - \bar{X}}{\sqrt{\text{Var}(X)}} \tag{3.40}$$

with $\bar{X}$ the mean of X and $\text{Var}(X)$ the variance of X.

Dimensionality Reduction via Principal Component Analysis (PCA)

The second preprocessing step involves applying PCA. This linear transformation technique rotates the feature space to align with the directions of maximum covariance with the target. PCA identifies the principal components that explain the greatest variance in the data. By retaining only the most informative components, PCA reduces the dataset's dimensionality, thereby improving generalization and reducing computational cost. This approach is especially beneficial in high-dimensional settings where many features are either redundant or irrelevant.

> **Example**
>
> To illustrate the concept of PCA, consider Figure 3.12, which presents a simplified example using a subset of the Iris dataset [39]. The original dataset includes two morphological features used to predict the species of flowers. After applying PCA, the data is rotated such that the first principal component captures the majority of the variance related to species classification. In this case, even a single component is sufficient to enable a reasonably accurate prediction.

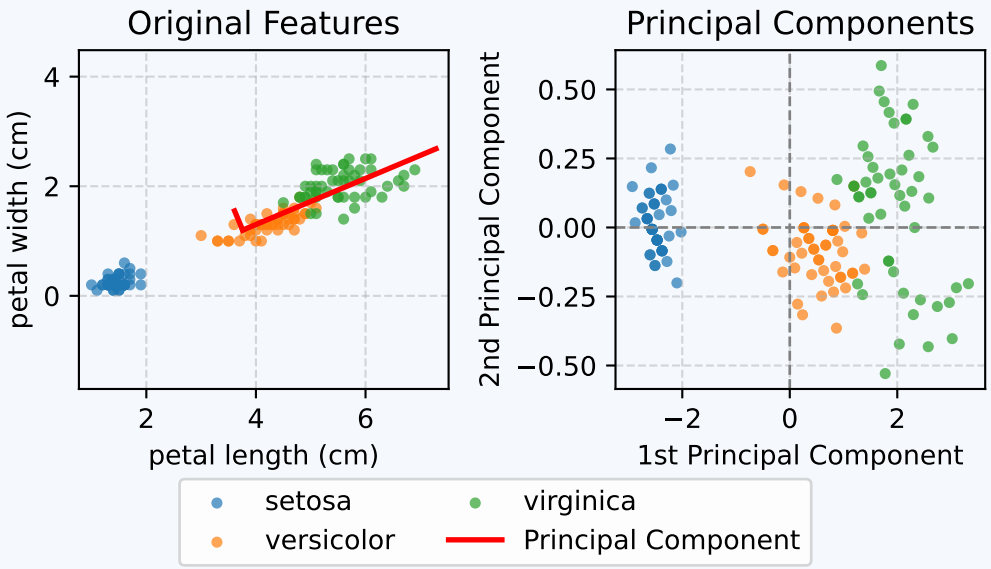


Figure 3.12: *Application of PCA on the Iris dataset.*

3.5.2 Overview of the Models

Linear Regression

Linear regression is a foundational and interpretable machine learning model that assumes a linear relationship between the input features $\mathbf{X}$ and the target $\mathbf{Y}$. The model seeks to fit a linear function of the form:

$$\widehat{\mathbf{y}} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{3.41}$$

where β_0 is the intercept and β_i are the coefficients associated with each feature.

The optimal parameters $\boldsymbol{\beta}$ are determined by minimising the mean squared error (MSE) between the predicted and true values:

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \|(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n) - y\|^2 \tag{3.42}$$

Ridge Regression

Ridge regression extends linear regression by introducing a regularisation term to mitigate overfitting. Overfitting occurs when a model learns to perfectly fit the training data at the expense of generalisability to unseen data. Figure 3.13 illustrates this behaviour in the context of polynomial regression: a degree-nine polynomial fits the training data exactly but fails to capture the underlying function in the test data.

The form of the ridge regression model remains:

$$\widehat{\mathbf{y}} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{3.43}$$

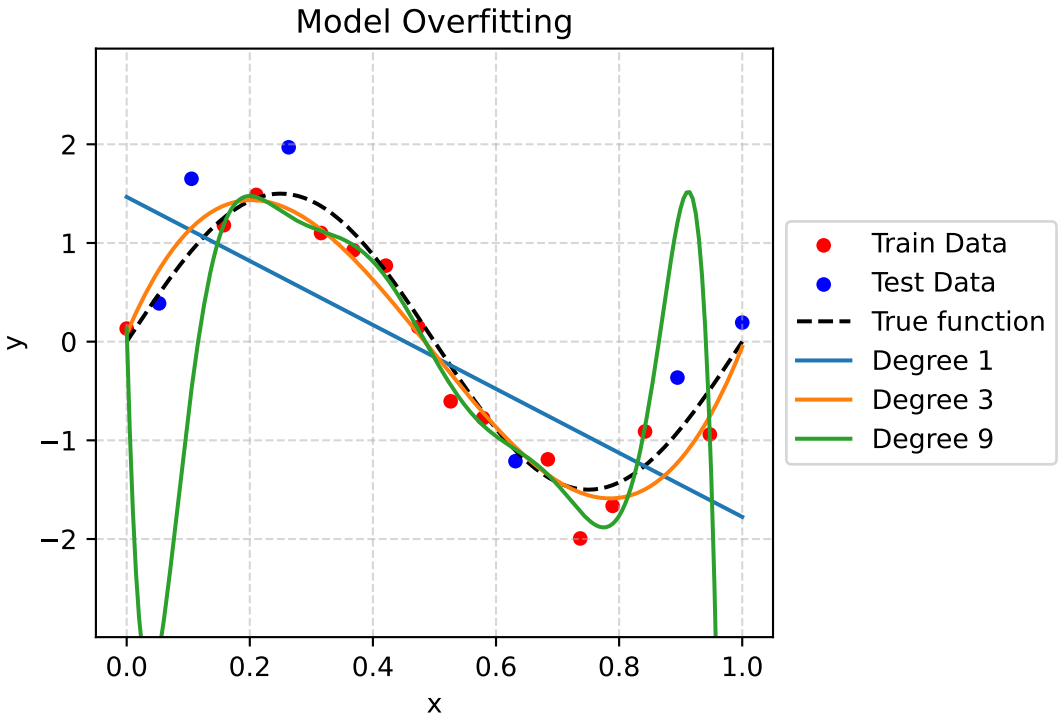


Figure 3.13: *Illustration of overfitting in polynomial regression.*

However, the optimisation objective is modified to include the ℓ_2 norm of the weights:

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \|(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n) - y\|^2 + \lambda \left\|\boldsymbol{\beta}\right\|_2 \tag{3.44}$$

where the ℓ_2 norm is defined as:

$$\ell_2(\mathbf{x}) = \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{3.45}$$

The inclusion of the ℓ_2 penalty discourages large parameter magnitudes, which are often symptomatic of overfitting. This constraint promotes smoother and more generalisable models.

Lasso Regression

Lasso regression is another regularised variant of linear regression. It also models the output as a weighted sum of the inputs:

$$\widehat{\mathbf{y}} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{3.46}$$

The key distinction from ridge regression lies in the regularisation term. Lasso

employs the ℓ_1 norm of the weights:

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \|(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n) - y\|^2 + \lambda \left\|\boldsymbol{\beta}\right\|_1 \tag{3.47}$$

with the ℓ_1 norm defined as:

$$\ell_1(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{3.48}$$

Unlike the ℓ_2 norm, the ℓ_1 norm encourages sparsity in the model coefficients. This means that Lasso tends to set many coefficients exactly to zero, effectively performing feature selection. The underlying intuition is that, whereas the squared-error loss heavily penalises large weights, the absolute-error loss applies a linear penalty, thereby facilitating simpler and more interpretable models.

Random Forest Regression

The random forest regression is an *ensemble method*. These types of models combine multiple submodels to make a final prediction, hence the forest. When trained, the random forest regressor creates multiple decision trees that are trained on a sampling with replacement of the training set, a technique called bootstrapping.

After training, when predicting, each of these decision trees produces a prediction; the final prediction is the average of the individual predictions of the trees. Figure 3.14 illustrates the prediction logic behind the model.

A notable property of the random forest regressor is that it can capture and model nonlinear relationships, unlike linear regression.

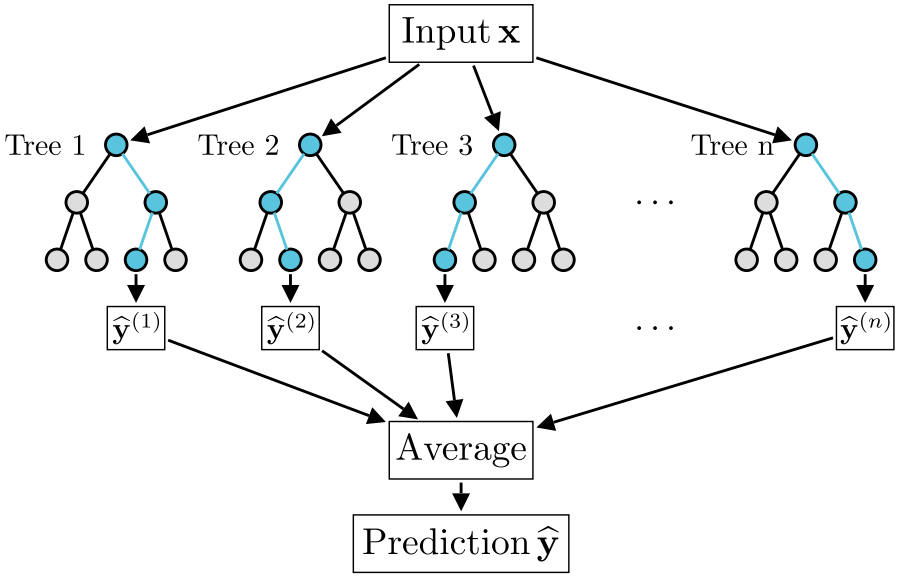


Figure 3.14: *Illustration of the Random Forest Regressor*

k-nearest neighbors regression

The K-nearest neighbors (KNN) regressor is a model that makes predictions by assuming similarity between examples learned during training and the unseen input.

During the training, the KNN regressor stores all training features in $\mathbf{X}_{train}$ and training target $\mathbf{Y}_{train}$. Afterward, when an input is fed to the model, a distance function D, for example, the Euclidean distance, is used to measure similarity between the input and every example stored from training

$$d_i = D(\mathbf{x}, \mathbf{x}_{train,i}) \tag{3.49}$$

Then the prediction is assumed to be the average of the targets of the k closest neighbors

$$\widehat{\mathbf{y}} = \frac{1}{k} \sum_{i=0}^{k} \mathbf{y}_{train,i} \tag{3.50}$$

Like the Random Forest Regressor, the KNN regressor can model a nonlinear system.

Example

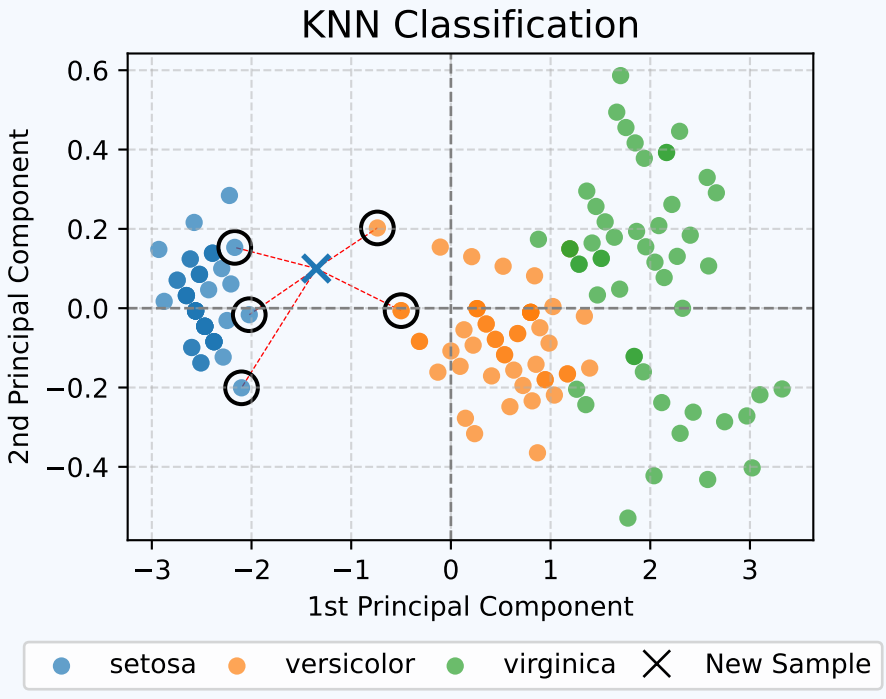


Figure 3.15: *Application of KNN on the Iris dataset*

To illustrate this concept, we can apply KNN to predict the flower species from the Iris dataset as in the previous example. The extension of the model to classifi-

cation is natural as the predicted class is the average class (i.e., the majority class in the neighborhood). The Figure 3.15 illustrates a KNN model parametrized to use the five closest neighbors. We observe that a point in between setosa and versicolor classes is classified based on its neighbor to the setosa species.

3.5.3 Training and Comparison of the Models

To assess the performance of each machine learning model against the analytical baseline, we adopt a structured training and evaluation procedure. The process involves both the optimisation of model hyperparameters and a principled splitting of the dataset to avoid overfitting and ensure fair comparison.

Dataset Splitting Strategy

A critical aspect of supervised learning is the division of the dataset into distinct subsets to fulfil different roles in the modelling process. Specifically, we partition the full dataset into three disjoint sets:

1. **Training set:** Used to learn the model parameters by fitting patterns in the input–output data.
2. **Validation set:** Used to tune the model's hyperparameters—those that govern model behaviour but are not learned during training (e.g., the regularisation parameter λ in ridge regression or the number of neighbours k in KNN).
3. **Test set:** Used exclusively for final evaluation. This set must remain unseen throughout both the training and hyperparameter tuning phases to provide an unbiased assessment of generalisation performance.

This three-way split is crucial. If the test set were used during the hyperparameter tuning stage, the model would become indirectly exposed to the test data, leading to overfitting and overestimated performance. By reserving the test set until the final evaluation, we ensure that the reported metrics reflect the model's true performance on unseen data.

However, when working with smaller datasets, allocating large portions of the data to validation and test sets reduces the number of examples available for training. To mitigate this issue, we employ *K-fold cross-validation*, a robust technique for performance estimation and hyperparameter tuning.

In K-fold cross-validation, the dataset is divided into k equally sized folds. The model is trained k times, each time using $k-1$ folds for training and the remaining fold for validation. The performance metrics are then averaged across all k runs to obtain a more reliable estimate of the model's generalisation capability. This procedure reduces the variance associated with the train–validation split, ensuring that every example contributes to both training and validation.

By combining K-fold cross-validation with a final, separate test set, we achieve a balance between reliable hyperparameter selection and unbiased evaluation.

In our experiments, we employ a training set comprising 4096 examples and a separate test set of 512 examples. The training set is further partitioned using a 5-fold cross-validation.

Hyperparameter Optimisation

To optimise model hyperparameters, we employ Bayesian optimisation using the `Optuna` framework [40]. Bayesian optimisation builds a probabilistic model of the objective function (in this case, model performance as a function of hyperparameters) and iteratively refines it to select hyperparameter configurations likely to improve performance [41].

The workflow proceeds as follows:

1. A statistical surrogate model estimates performance over the hyperparameter space.
2. Candidate configurations are evaluated using the training and validation sets.
3. The surrogate model is updated based on the observed results.
4. This process continues iteratively to converge on the optimal configuration.

Once the best hyperparameters are identified, the final model is retrained using both the training and validation data and then evaluated on the test set. This ensures that the final comparison between models reflects true generalisation performance rather than artefacts of tuning.

Comparison of the Models

To evaluate the robustness of traditional machine learning models, we compare their performance against the analytical method under identical conditions. Specifically, we assess their ability to predict axial rigidities from input features corrupted by measurement noise. As previously discussed in Section 3.4, realistic noise levels encountered during on-site measurements typically range between 0% and 5%.

To simulate noisy conditions, the input features $\mathbf{x}$ are perturbed using multiplicative noise as follows:

$$\widetilde{\mathbf{x}} = \boldsymbol{\alpha} \odot \mathbf{x} \tag{3.51}$$

where the noise vector $\boldsymbol{\alpha}$ is sampled element-wise from a normal distribution:

$$\alpha_i \sim \mathcal{N}(\mu = 1, \sigma = \frac{\epsilon}{2}) \tag{3.52}$$

The prediction performance of each model is quantified using the Mean Absolute Percentage Error (MAPE), which provides a relative error metric between the predicted and true values.

Several insights can be drawn from Figure 3.16:

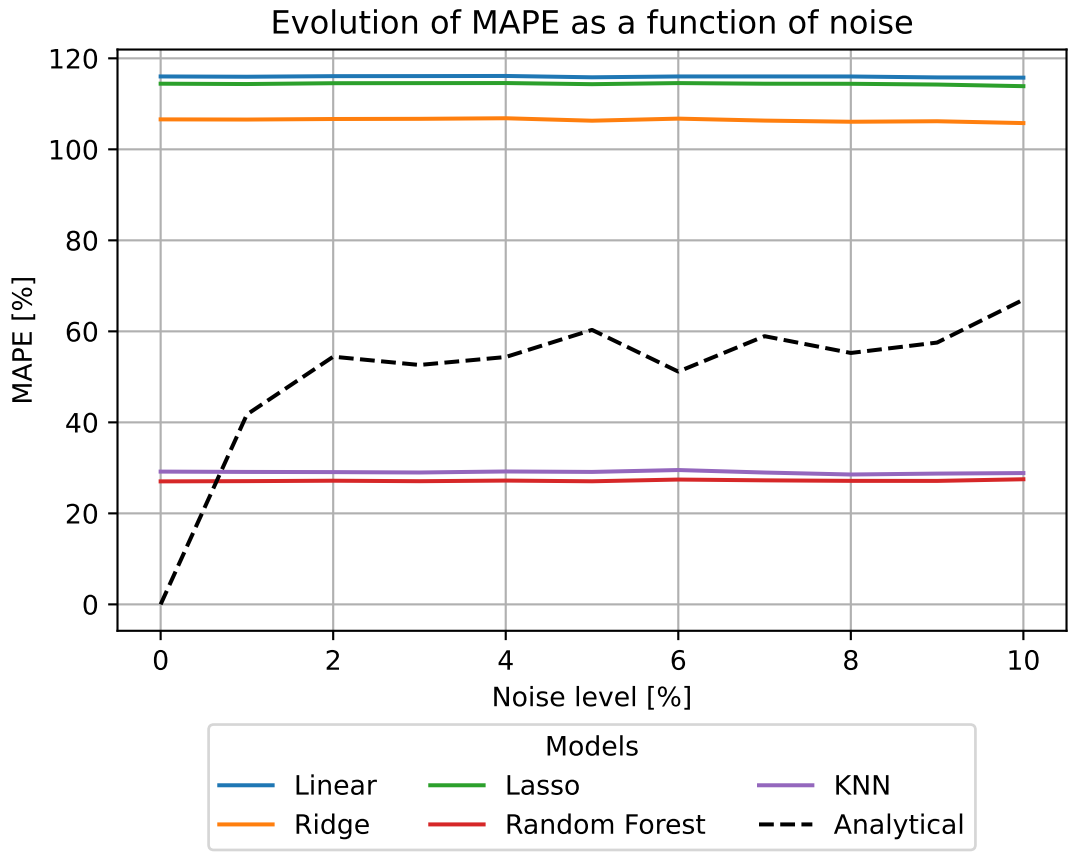


Figure 3.16: *Comparison of model performance under varying noise levels. All models were trained and tuned using 5-fold cross-validation on a noiseless training dataset containing 4096 samples, and evaluated on 512 previously unseen noisy samples.*

First, the regression-based models (linear, ridge, and lasso) exhibit significantly higher prediction errors compared to the analytical solution, the K-nearest neighbours (KNN) regressor, and the random forest model. This outcome suggests that the inverse problem under consideration is fundamentally nonlinear. While the forward structural model adheres to linear theory under assumptions of material and geometric linearity, the inverse mapping from displacements and loads to stiffness properties is inherently more complex.

To understand this, consider the constitutive relation:

$$\mathbf{Ku} = \mathbf{q} \tag{3.53}$$

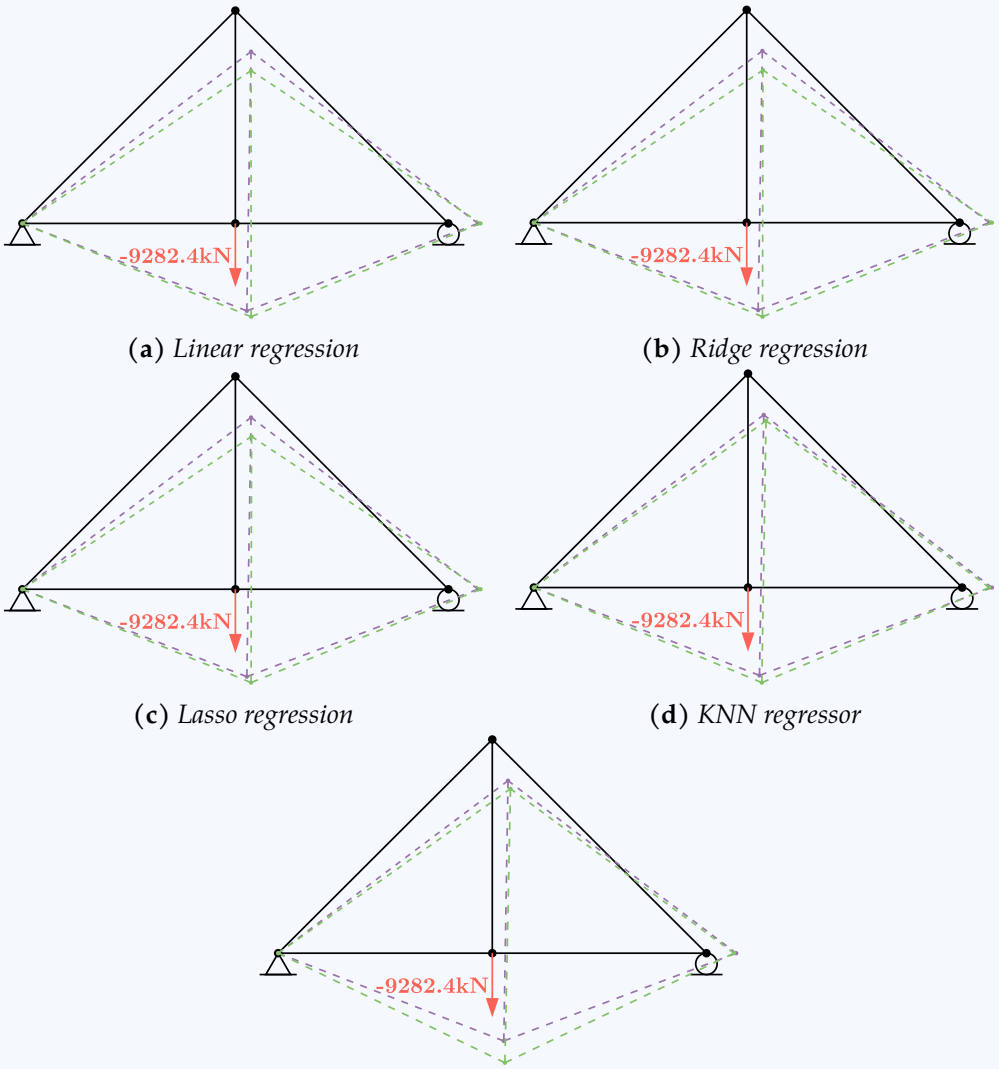
Although this equation is linear with respect to the unknown displacements when stiffness $\mathbf{K}$ is known, the inverse problem reverses this logic: the aim is to determine the stiffness from known displacements and loads. In this setting, the stiffness matrix $\mathbf{K}$ encodes both material and geometric information. Crucially, the geometric component, such as member orientation, introduces nonlinear interdependencies among elements, making the mapping from input features to stiffness values nonlinear.

Second, the results demonstrate that machine learning models exhibit greater robustness to noise than the analytical approach. Even under relatively high noise levels, with errors approaching 23%, the predictions remain stable. While the overall accuracy of the traditional models may not yet be sufficient for practical deployment, their stability under noisy conditions suggests that more advanced machine learning techniques have the potential to outperform purely analytical solutions in real-world scenarios involving imperfect data.

Example

To further illustrate the predictive capability of the models, we visualise their performance on a randomly selected example from the test dataset. For each model, the axial rigidities EA are predicted based on noisy input features with 2% added noise. These predicted values are then used to recompute the structural displacements under the original applied load. The goal is to assess how closely the reconstructed structure's displacements align with the ground truth obtained from the actual rigidities.

Figure 3.17 presents this comparison across several models. The displacements computed from the true axial rigidities are shown in purple, while those based on the predicted values are shown in green. All displacements are scaled by a factor of 100 for visual clarity.



(**a**) *Linear regression*

(**b**) *Ridge regression*

(**c**) *Lasso regression*

(**d**) *KNN regressor*

(**e**) *Random forest regressor*

Figure 3.17: *Comparison of structural deformations computed using true axial rigidities (purple) and predicted axial rigidities from features with 2% noise (green)under identical loading conditions. Deformations are scaled by a factor of 100 for visibility.*

Although the MAPE values for some models are relatively high, the visual discrepancies in displacement remain modest in this simple example. Nevertheless, the impact of prediction error becomes more pronounced in more complex structural configurations, as further explored in Chapter 5.

3.6 Neural Network approach

In this section, we will apply the multilayer perceptron model, introduced in Section 2.2, to solve the inverse problem. The first part will describe the hyperparameter selection process used for this model. Then, we will examine the model's performance. This model will be trained using the Adam optimizer and the MSE loss.

3.6.1 Hyperparameter Tuning

Selecting the appropriate hyperparameters is crucial for achieving high model performance. In this section, we explore the impact of key hyperparameters introduced in Section 2.2.2, which govern the structure and learning dynamics of a multilayer perceptron (MLP). The hyperparameters considered are:

1. The number of hidden layers and the number of neurons per layer,
2. The activation function,
3. The learning rate.

The tuning process is conducted using a standard procedure: we evaluate a wide range of hyperparameter configurations using 5-fold cross-validation on the previously described training dataset, which consists of 4096 examples. Model performance is then assessed on a separate test set comprising 512 samples, which remains untouched during training and validation. The model is trained using batches of size 512.

As a starting point, we adopt a commonly used baseline configuration for the initial training round:

- **Activation function:** Rectified Linear Unit (ReLU),
- **Learning rate:** 10^{-3}.

These initial values serve as a reference for evaluating the performance gains achieved during the hyperparameter optimisation process. The configuration is iteratively refined to identify the combination that yields the best predictive accuracy and generalisation.

Number of Hidden Layers and Neurons per Layer

The number of hidden layers and the number of neurons per layer play a pivotal role in determining the model's learning capacity. A model with too few layers may lack the complexity required to capture nonlinear relationships, while an overly large model may suffer from overfitting and increased computational cost.

To systematically investigate the influence of model capacity, we evaluate a range of configurations using a 5-fold cross-validation approach. All models use the Rectified Linear Unit (ReLU) activation function and a learning rate of 10^{-3}. The eight best-performing configurations are compared in Figure 3.18, which reports averaged training and validation metrics across folds.

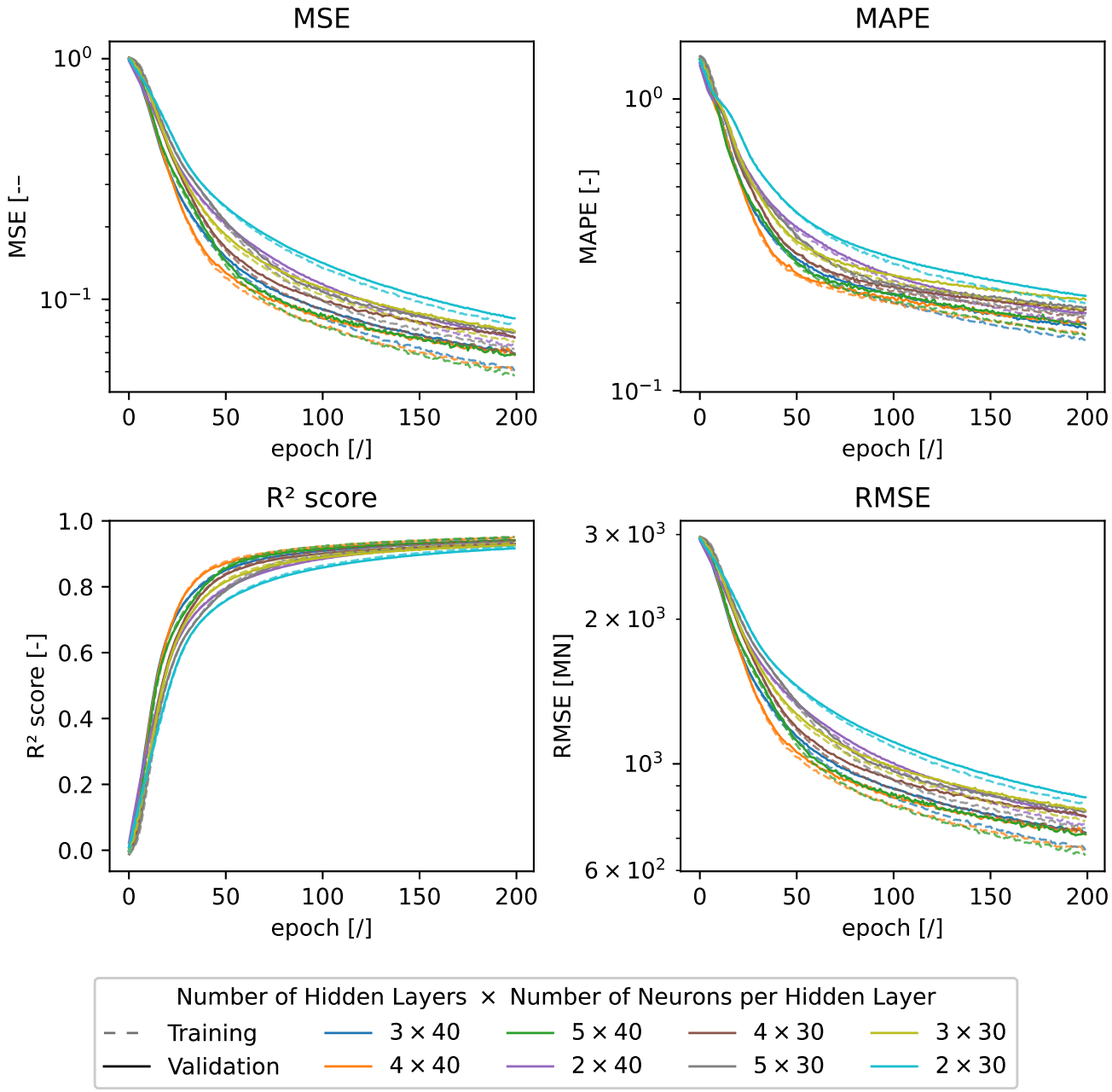


Figure 3.18: *Average metrics across 5-fold training with an MLP using ReLU activation and learning rate of* 10^{-3} *for multiple numbers of hidden layers and layer sizes.*

Model performance is evaluated using the following metrics:

- **Mean Squared Error (MSE)**: Captures the average squared deviation between predicted and actual values. It heavily penalises larger errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.54}$$

- **Root Mean Squared Error (RMSE)**: The square root of MSE. It provides a more

interpretable error in the same units as the target variable.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.55}$$

- **Coefficient of Determination** (R^2): Measures the proportion of variance in the target explained by the model. An R^2 of 1 corresponds to a perfect model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.56}$$

- **Mean Absolute Percentage Error** (**MAPE**): Expresses prediction error as a percentage, facilitating interpretability. However, it is sensitive to small true values.

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3.57}$$

It is worth noting that MSE is computed on standardised data and therefore lacks physical interpretability; it is useful primarily for relative comparisons. In contrast, RMSE is computed on de-standardised predictions, and its magnitude reflects expected physical error in the same units as the target.

Although the performance of the configurations varies across metrics, selecting the optimal model requires a principled approach to mitigate subjective bias. We therefore apply a multi-criteria selection strategy:

1. Identify the model with the lowest MAPE.
2. Retain all configurations with MAPE within 2% of this minimum.
3. Within the remaining models, select the one with the lowest MSE.
4. Retain all configurations with MSE within 2% of this value.
5. If multiple models remain, select the one with the highest R^2 score.

This approach ensures that the selected model not only exhibits strong average predictive accuracy, as measured by MAPE, but also minimizes large errors using MSE and explains the maximum possible variance in the target variable via the R^2 score.

Following this strategy, the selected configuration consists of three hidden layers, each with 40 neurons.

Activation Function

The choice of activation function has a profound impact on a neural network's ability to model nonlinear relationships.

Figure 3.19 compares several activation functions for an MLP with three hidden layers of 40 neurons each and a learning rate of 10^{-3}. Among the tested options, the hyperbolic tangent ($\tanh$) activation function consistently yields the best metrics across

all evaluated metrics. Notably, the MAPE improves substantially from approximately 18% with the previously used activation function to around 10% with $\tanh$.

Based on these findings, the $\tanh$ activation function is adopted for the final model configuration.

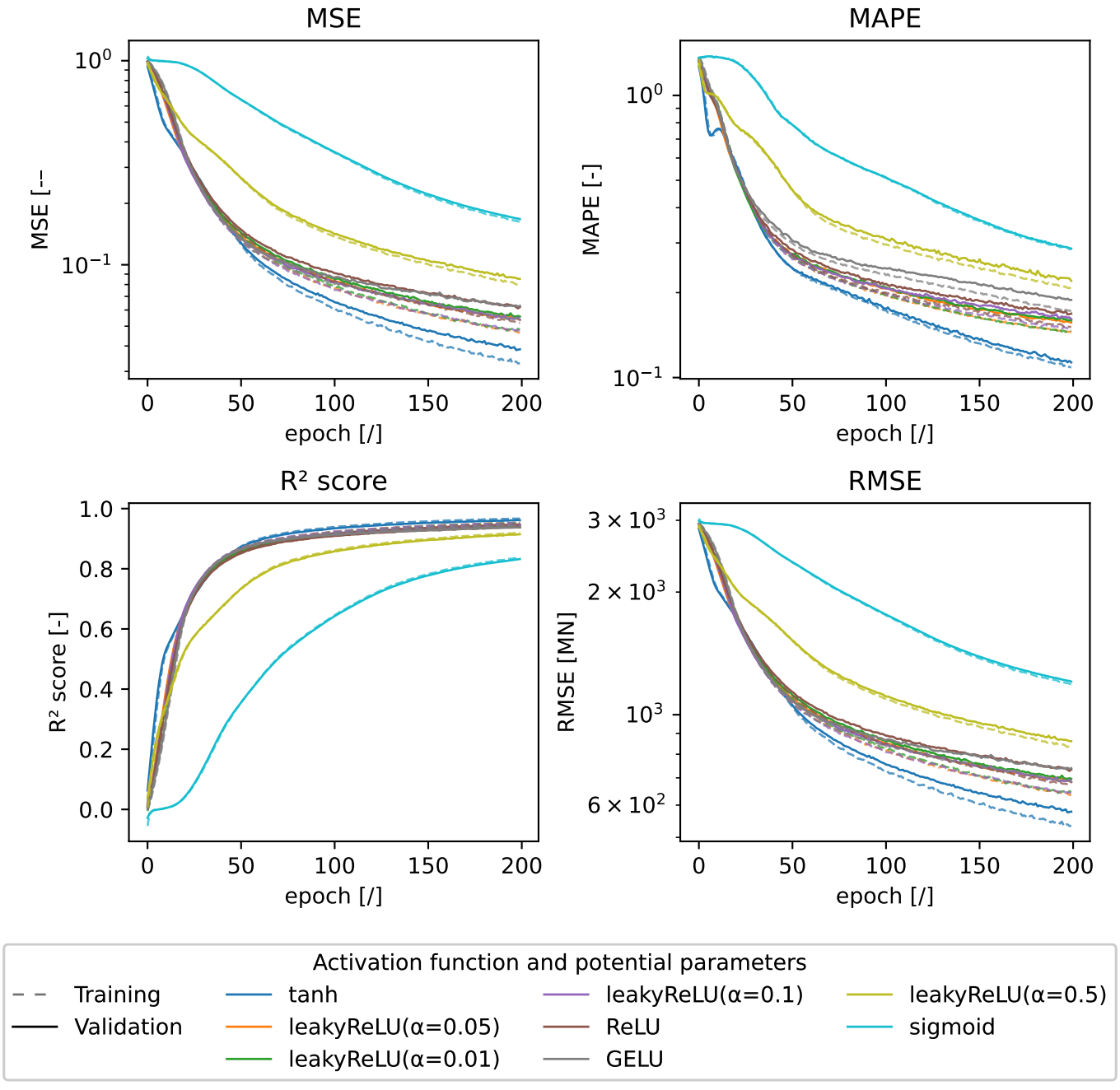


Figure 3.19: *Average metrics across 5-fold training with an MLP with three hidden layers of 40 neurons each and a learning rate of 10^{-3} for multiple activation functions.*

Learning Rate

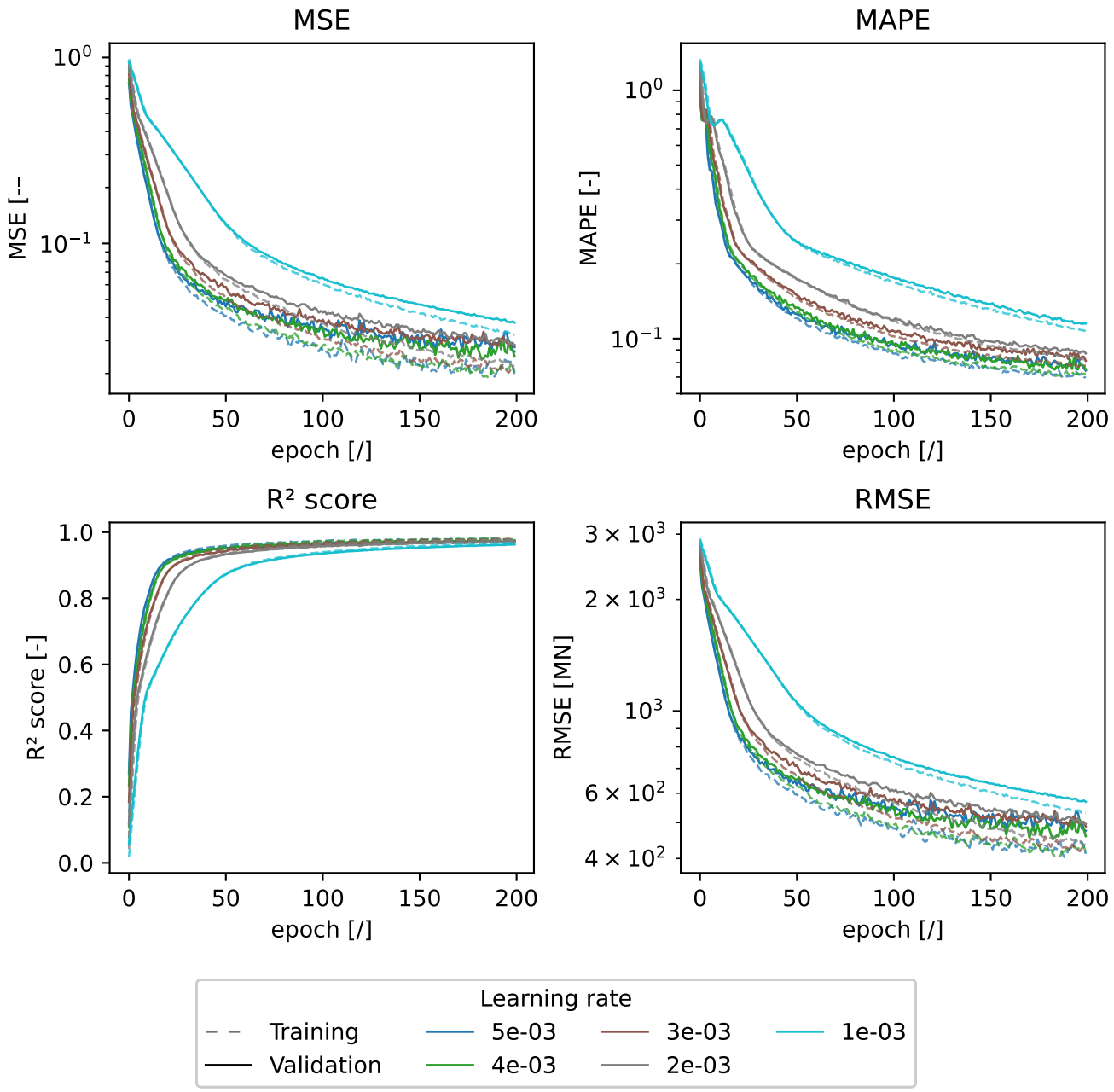


Figure 3.20: *Average metrics across 5-fold training with an MLP with three hidden layers using* tanh *activation function for multiple learning rates.*

The learning rate is a critical hyperparameter that controls the step size during gradient descent optimisation. Selecting an appropriate value is non-trivial, as it involves balancing convergence speed and training stability. A large learning rate may result in oscillations or divergence, whereas an overly low rate can lead to slow or stalled convergence.

Figure 3.20 displays the evolution of several performance metrics for different learning rates. Notably, a learning rate of 5×10^{-3} leads to rapid initial learning but intro-

duces high variance in the optimisation process, as evidenced by oscillations in the MSE curve—particularly noticeable after approximately 100 epochs. This instability is indicative of noisy updates and may hinder convergence to a well-generalised solution.

To mitigate this effect, a more conservative learning rate of 3×10^{-3} is preferred. Although this results in a slower convergence rate, the increased stability allows for more consistent learning. The performance gap can be addressed through extended training over a greater number of epochs, making this learning rate a more reliable choice for subsequent model development.

3.6.2 Model Comparison

With the optimal hyperparameters selected, the multilayer perceptron (MLP) is trained using the full training dataset. The resulting model is then evaluated against previously explored methods to assess improvements in performance.

Figure 3.21 illustrates the evolution of training and test metrics across training epochs. As is commonly observed in neural network optimisation, metric curves tend to exhibit increased fluctuations near convergence, reflecting the model's oscillations around a local minimum. The use of a logarithmic scale accentuates this phenomenon. To enhance interpretability, the curves have been smoothed.

As expected, the training metrics are consistently better than those on the test dataset, as the model is explicitly optimised on the training data. However, the close alignment between training and test metrics suggests that the model generalises well to previously unseen data.

According to the selection criteria established earlier for model capacity, the model achieved its optimal performance at epoch 1894. The corresponding metrics at that epoch are presented in Table 3.3.

Dataset	MSE [-]	MAPE [%]	R^2 Score	RMSE [MN]
Training	5.81e-03	3.79	0.994	223
Test	9.67e-03	3.85	0.990	290

Table 3.3: *Performance metrics at epoch 1894.*

The model demonstrates strong predictive performance, achieving an average relative error (MAPE) below 4% on the test set. Furthermore, the high R^2 score indicates that the model captures the vast majority of variance in the target data.

To enable a fair comparison with other approaches, the model is also evaluated under varying levels of noise, replicating the testing conditions applied in Section 3.4. The results are presented in Figure 3.22, alongside those of the analytical method, KNN regressor, and random forest regressor.

The MLP exhibits a clear advantage in predictive accuracy and robustness to noise. Notably, its prediction error remains below the added noise level for noise intensities exceeding approximately 7%, demonstrating the model's capacity to extract meaningful patterns even in the presence of significant perturbations.

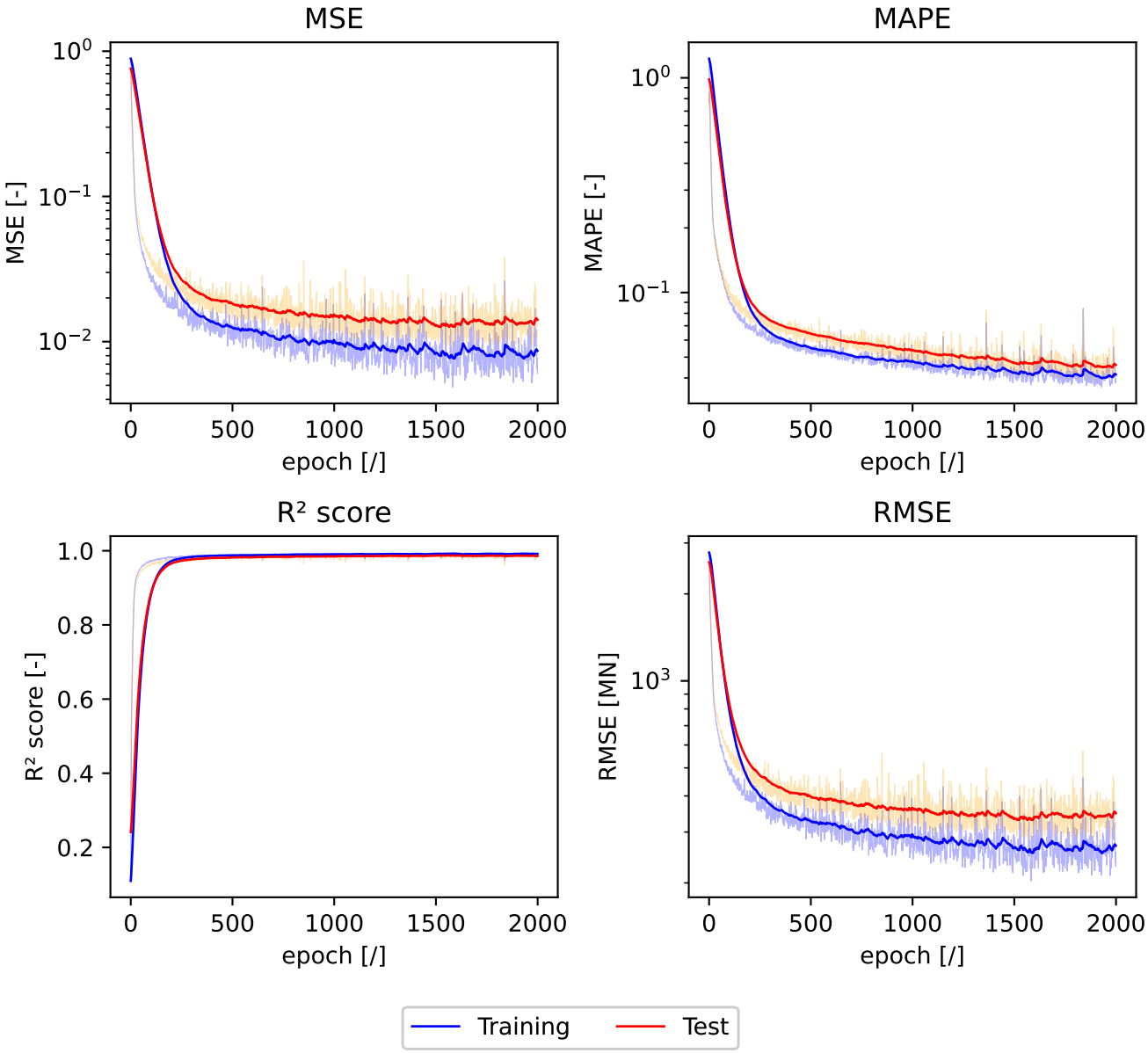


Figure 3.21: *Training and test metrics of an MLP with three hidden layers of 40 neurons each using* tanh *activation function trained with a dataset of 4096 samples and tested on a dataset of 512 samples.*

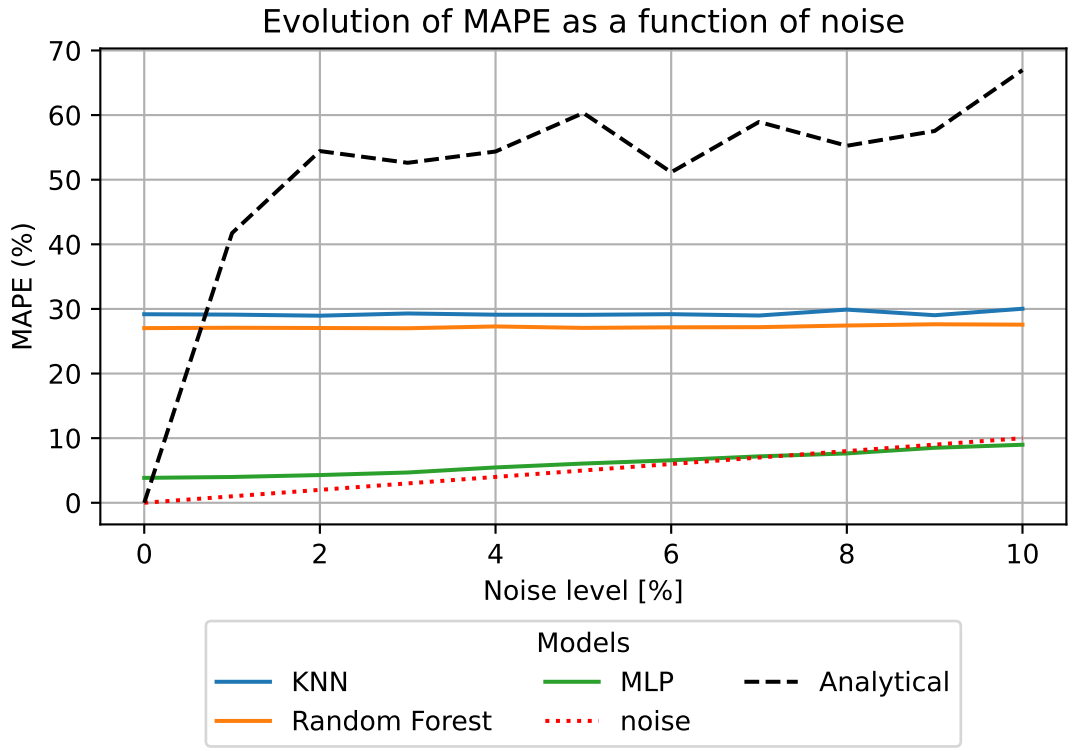


Figure 3.22: *Comparison of the MAPE for the MLP, KNN regressor, and random forest regressor against the analytical method under increasing noise levels. The noise level is indicated with a dotted red line for comparison.*

In conclusion, the MLP outperforms traditional methods across both noiseless and noisy conditions. Nonetheless, there remains potential for further improvement by incorporating physical knowledge directly into the model's architecture or learning process.

Example

To further demonstrate the predictive capabilities of the model, its performance is evaluated on the same randomly selected test example previously used. The axial rigidities EA are predicted using input features corrupted with 2% multiplicative random noise. These predicted rigidities are then used to recompute the displacements of the structure under the original applied load. The objective is to assess how closely the reconstructed structural response matches the true displacement field derived from the actual axial rigidities.

Figure 3.23 presents the results obtained using the trained MLP model.

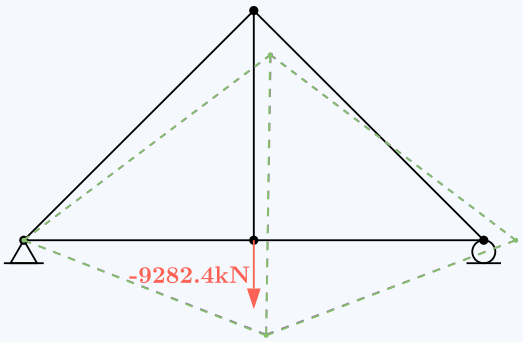


Figure 3.23: *Comparison of structural deformations computed using the true axial rigidities (purple) and those predicted from noisy input features (green), under identical loading conditions. Deformations are magnified by a factor of 100 for visual clarity.*

The figure reveals that the improved accuracy of the MLP model effectively translates into more faithful structural behaviour, as the predicted and actual deformed structures are nearly indistinguishable. This indicates a high level of reliability in the model's predictions, even under moderate noise conditions.

To further evaluate model performance, the parity plot presented in Figure 3.24 compares predicted axial rigidities $\widehat{EA}$ to their true values EA from the noisy test dataset. In an ideal scenario, all points would lie on the identity line $\widehat{EA} = EA$, indicating perfect agreement between prediction and ground truth. In this case, the points are densely clustered around the identity line, forming a narrow, well-aligned band. This pattern demonstrates that the model achieves high predictive accuracy overall, with only a limited number of deviations.

A complementary analysis of error distribution across structural members is shown in Figure 3.25. The results indicate that prediction errors are uniformly distributed

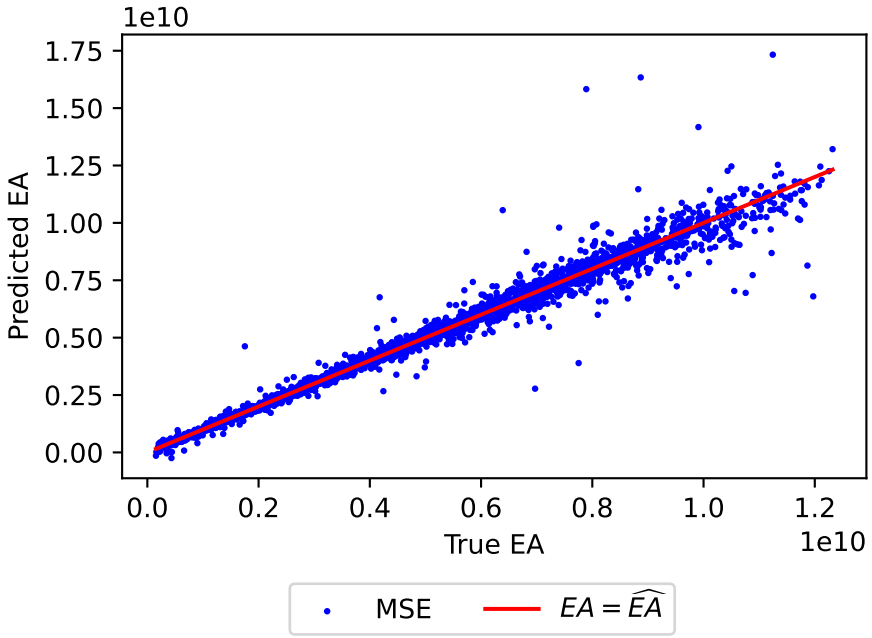


Figure 3.24: *Parity plot comparing the predicted axial rigidity $\widehat{EA}$ from the MSE-based MLP with the true axial rigidity EA for a test set of 512 examples subject to ±5% measurement noise.*

throughout the structure, with no single member consistently exhibiting largely higher error. This suggests that the model learns the behaviour of all members effectively, without bias toward specific elements.

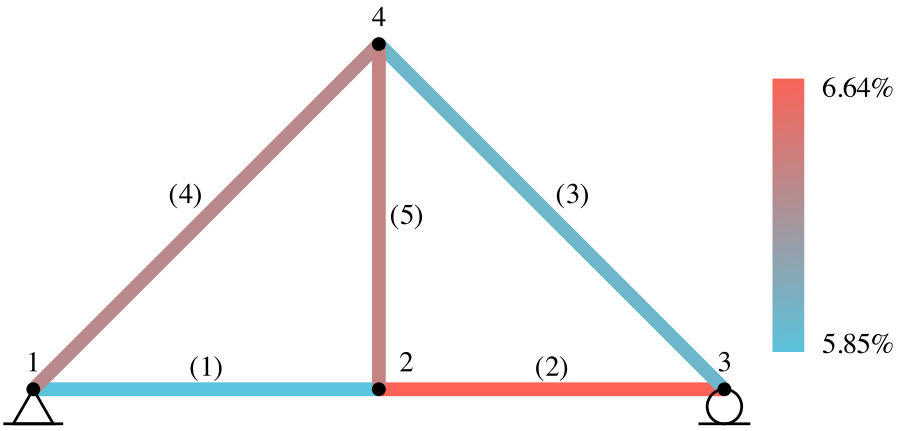


Figure 3.25: *Distribution of mean absolute error across members for a noisy test dataset with ±5% measurement noise.*

3.7 Physics-Informed Neural Networks

The multilayer perceptron (MLP) employed thus far relies solely on learning statistical correlations between the input features (i.e., nodal displacements, loads, and strains)and the target variable, the axial rigidities. While this data-driven approach has demonstrated strong predictive capabilities, it does not exploit any underlying physical princi-

ples governing the behaviour of structures. Incorporating domain-specific knowledge offers an opportunity to enhance both the accuracy and generalisation of the model.

To this end, we turn to the framework of *physics-informed neural networks* (PINNs), a paradigm introduced by Raissi et al. [16]. PINNs embed physical laws directly into the learning process by designing custom loss functions that penalise violations of known physical constraints. This results in models that not only fit the data but also remain consistent with established physical theory.

In this work, we adopt an adaptation of PINNs known as *finite-element informed neural networks* (FEI-NNs), as proposed in the context of structural mechanics by Leduc et al. [21] and Meethal et al. [22]. FEI-NNs extend the core concept of PINNs by leveraging finite element method (FEM) formulations.

The central idea of this technique lies in augmenting the loss function with physics-based terms that enforce structural consistency, effectively guiding the neural network to learn solutions that are not only accurate but also physically plausible.

3.7.1 Physics-Based Loss Functions

In the following sections, we formulate and develop on a series of physics-based loss functions, each designed to incorporate specific aspects of structural mechanics into the training of the model. These losses are constructed to penalise violations of physical principles, such as equilibrium, compatibility, or constitutive relations, and are intended to replace the conventional mean squared error (MSE) loss during training.

For each proposed loss function, a multilayer perceptron (MLP) is trained using only the physics-informed loss in isolation, thereby allowing a focused analysis of its individual effectiveness. To maintain clarity and conciseness, details related to hyperparameter tuning are omitted in this section. The tuning procedure remains consistent with the methodology described in Section 3.6.1.

Structure Stiffness Loss

This formulation adapts the loss function proposed by Le Duc et al. [21] and Meethal et al. [22] for addressing inverse problems in structural analysis.

We begin with the constitutive equation governing structural equilibrium:

$$\mathbf{K}\mathbf{u} = \mathbf{q} \tag{3.58}$$

which, accounting for boundary conditions, can be reduced to:

$$\mathbf{K}_{bb}\mathbf{u}_b = \mathbf{q}_{\text{ext}} \tag{3.59}$$

Here, $\mathbf{K}_{bb}$ is the reduced stiffness matrix corresponding to the free degrees of freedom, $\mathbf{u}_b$ the corresponding displacements, and $\mathbf{q}_{\text{ext}}$ the external loading vector.

Given the model predictions $\widehat{\mathbf{y}}$ for the axial rigidities EA, an approximate stiffness matrix $\widehat{\mathbf{K}}_{bb}(\widehat{\mathbf{y}})$ can be assembled using the finite element procedure described in Sec-

tion 3.2.2. In the ideal case where $\widehat{\mathbf{y}} = \mathbf{y}$, this matrix would satisfy:

$$\widehat{\mathbf{K}}_{bb}(\widehat{\mathbf{y}})\mathbf{u}_b - \mathbf{q}_{\text{ext}} = \mathbf{0} \tag{3.60}$$

Hence, we define the residual vector:

$$\boldsymbol{R}(\widehat{\mathbf{y}}, \mathbf{u}_b, \mathbf{q}_{\text{ext}}) = \widehat{\mathbf{K}}_{bb}(\widehat{\mathbf{y}})\mathbf{u}_b - \mathbf{q}_{\text{ext}} \tag{3.61}$$

The corresponding structure stiffness loss is the mean squared error of the residual:

$$\mathcal{L}_{\text{stiffness}}(\widehat{\mathbf{y}}, \mathbf{u}_b, \mathbf{q}_{\text{ext}}) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{R}_i^2 = \frac{1}{n}\sum_{i=1}^{n} \left(\widehat{\mathbf{K}}_{bb}(\widehat{\mathbf{y}})\mathbf{u}_b - \mathbf{q}_{\text{ext}}\right)_i^2 \tag{3.62}$$

Example

Consider the truss structure subjected to a vertical load of 1000 kN applied at the central node, as shown in Figure 3.26.

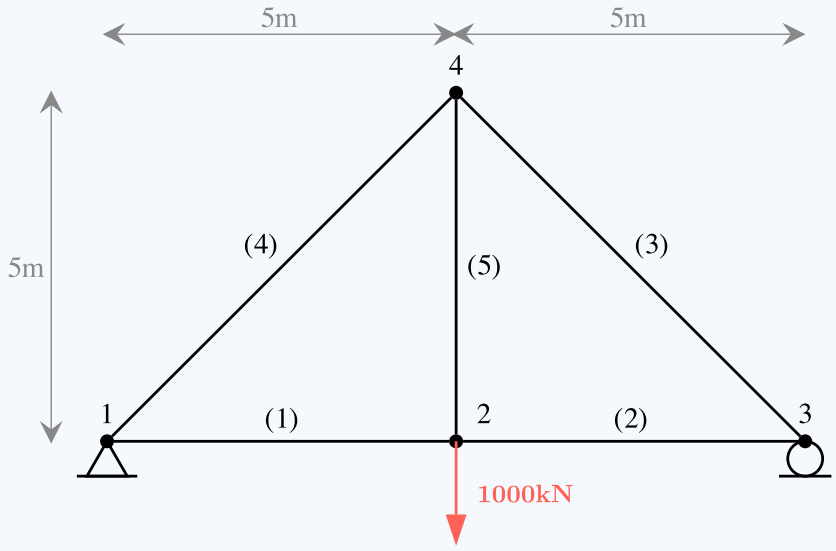


Figure 3.26: *Truss structure studied in this section.*

The measured displacement vector is:

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 4.761 \\ -27.754 \\ 9.524 \\ 0 \\ 4.762 \\ -18.231 \end{bmatrix} \times 10^{-3}\,[\text{m}] \tag{3.63}$$

Assume the model predicts the following axial rigidities:

$$\widehat{\mathbf{y}} = \begin{bmatrix} 500 \\ 525 \\ 510 \\ 520 \\ 525 \end{bmatrix} \times 10^6 \, [\mathrm{N}] \tag{3.64}$$

From this, we compute the reduced stiffness matrix:

$$\widehat{\mathbf{K}}_{bb} = \begin{bmatrix} 209.0 & 0.0 & -105.0 & 0.0 & 0.0 \\ 0.0 & 105.0 & 0.0 & 0.0 & -105.0 \\ -105.0 & 0.0 & 141.06 & -36.06 & 36.06 \\ 0.0 & 0.0 & -36.06 & 72.83 & 0.71 \\ 0.0 & -105.0 & 36.06 & 0.71 & 177.83 \end{bmatrix} \times 10^6 \tag{3.65}$$

The residual is then:

$$\boldsymbol{R} = \begin{bmatrix} -4971.0 \\ 85.0 \\ 14389.92 \\ -9524.02 \\ 18940.81 \end{bmatrix} \tag{3.66}$$

The stiffness loss becomes:

$$\mathcal{L}_{\text{stiffness}} = 1.3625 \times 10^8 \tag{3.67}$$

Following hyperparameter tuning, the model is trained under the same conditions as the MSE-based MLP. The adopted configuration is:

- **Loss:** Stiffness loss
- **Architecture:** Two hidden layers with 40 neurons each
- **Activation function:** LeakyReLU ($\alpha = 10^{-3}$)

Figure 3.27 displays the model's performance throughout training. The stiffness-based loss effectively guides improvement in validation accuracy over successive epochs. However, the model did not achieve a mean absolute percentage error (MAPE) below 11% after 2000 epochs, whereas the MSE-based model reached a MAPE below 4%.

Interestingly, the selected hyperparameters differ from those of the MSE-trained model, and the evolution of performance metrics also diverges. Notably, the R^2 score and RMSE initially deteriorate before improving, suggesting that the model learns fundamentally different patterns when trained using the physics-informed loss function.

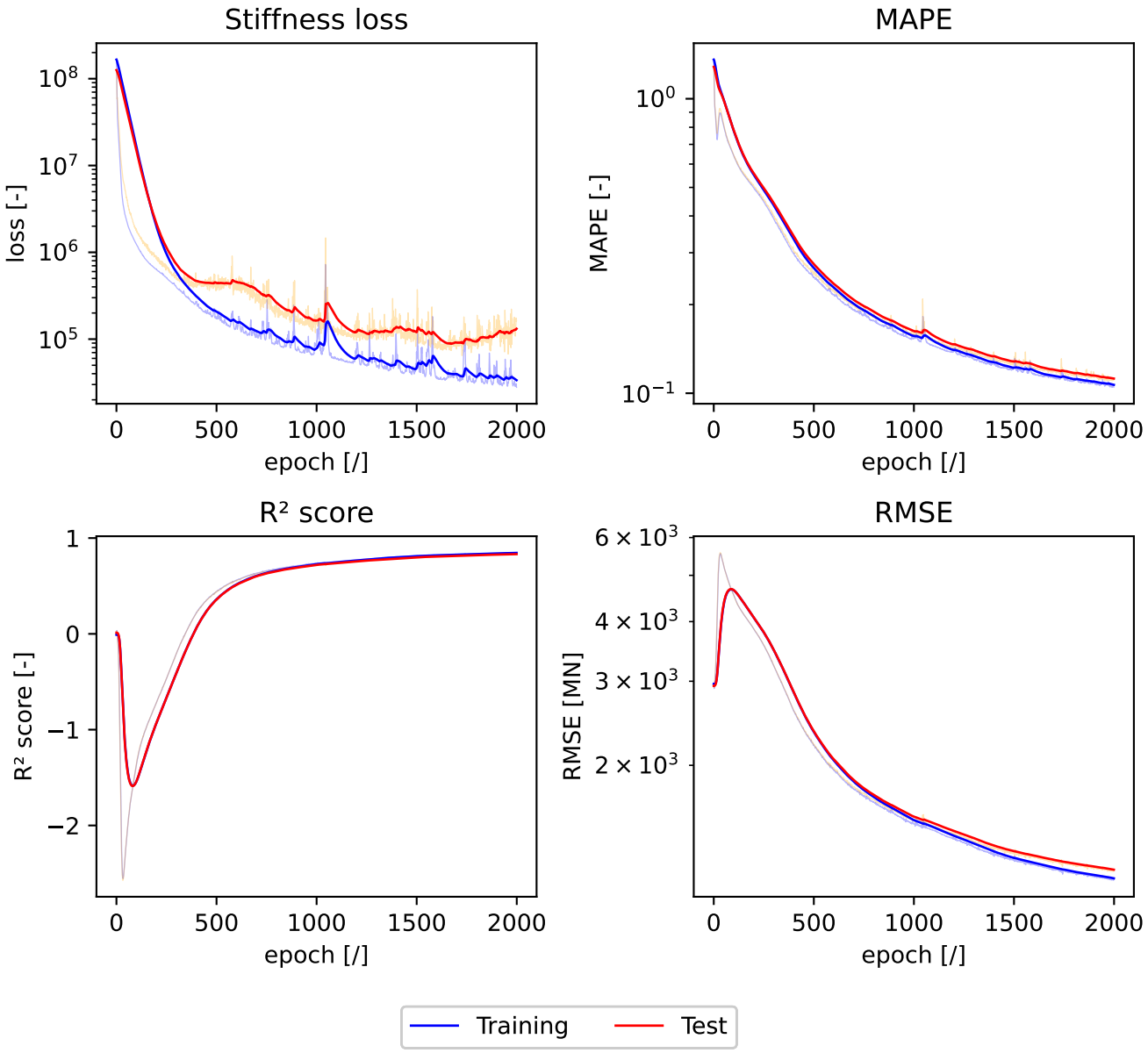


Figure 3.27: *Training performance of the MLP model using the stiffness loss. The network consists of two hidden layers with 40 neurons each and a LeakyReLU activation function. Training was conducted on a dataset of 4096 examples, with evaluation on a test set of size 512, using a learning rate of* 10^{-3} *and batch size of 512.*

Node Equilibrium Loss

Assuming the structure is in a static equilibrium state, classical mechanics dictates that two fundamental equilibrium conditions must be satisfied:

- **Force equilibrium:**

$$\sum_i \mathbf{F}_i = \mathbf{0} \tag{3.68}$$

- **Moment equilibrium:**

$$\sum_i \mathbf{M}_i = \mathbf{0} \tag{3.69}$$

where $\mathbf{F}_i$ and $\mathbf{M}_i$ represent the forces and moments acting on the structure, respectively.

For truss structures, the moment equilibrium condition is inherently satisfied due to the nature of axial member forces and idealised pin connections. Consequently, we focus solely on the force equilibrium condition to construct the loss function.

This equilibrium principle is valid globally and locally all across the structure. Let us consider a node i within the truss, subject to both internal axial forces and external loads, as illustrated in Figure 3.28.

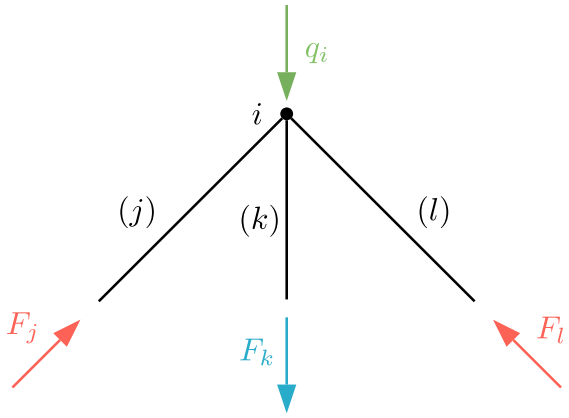


Figure 3.28: *Illustration of a node subjected to internal member forces and external loads.*

The external nodal load $\mathbf{q}_i$ is known, while the internal axial forces are derived from the strain and axial rigidity of each connected member. According to Hooke's Law (Equation 3.25), the internal force in member j is given by:

$$F_j = \varepsilon_j (EA)_j \tag{3.70}$$

Accordingly, the local force equilibrium at node i becomes:

$$\sum_{j \in C(i)} \varepsilon_j (EA)_j \mathbf{l}_{i,j} - \mathbf{q}_i = \mathbf{0} \tag{3.71}$$

where $C(i)$ denotes the set of members connected to node i, $\mathbf{l}_{i,j}$ is the unit direction vector for member j point away from node i, and $\mathbf{q}_i$ includes both applied loads and support reactions. The unit vectors $\mathbf{l}_{i,j}$ serve to convert scalar axial forces into vectorial contributions while maintaining the sign convention of F.

To form a loss function, we define the residual $\mathbf{R}_i$ using the predicted axial rigidities $\widehat{\mathbf{y}}$:

$$\mathbf{R}_i\left(\widehat{\mathbf{y}}, \boldsymbol{\varepsilon}, \mathbf{q}\right) = \sum_{j \in C(i)} \varepsilon_j \widehat{y}_j \mathbf{l}_{i,j} - \mathbf{q}_i \tag{3.72}$$

which represents the unbalanced force at node i. The node equilibrium loss is then computed as the mean squared norm of these residuals over all nodes:

$$\mathcal{L}_{\text{equilibrium}}\left(\widehat{\mathbf{y}}, \boldsymbol{\varepsilon}, \mathbf{q}\right) = \frac{1}{|\mathrm{N}|} \sum_{i=1}^{|\mathrm{N}|} \left\|\mathbf{R}_i\right\|^2 = \frac{1}{|\mathrm{N}|} \sum_{i=1}^{|\mathrm{N}|} \left\| \left(\sum_{j \in C(i)} \varepsilon_j \widehat{y}_j \mathbf{l}_{i,j} \right) - \mathbf{q}_i \right\|^2 \tag{3.73}$$

Although the underlying physical principle of this loss is analytically to the structure stiffness loss introduced in Equation 3.58, the formulation differs substantially. The stiffness-based loss penalises discrepancies at the global system level, whereas the node equilibrium loss enforces local consistency, which may be more sensitive to local inaccuracies in the prediction.

Example

Consider again the reference truss used for stiffness-based loss (Figure 3.26). The measured strains in the structure are:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} 0.0009524 \\ 0.0009524 \\ -0.0013469 \\ -0.0013469 \\ 0.0019048 \end{bmatrix} \tag{3.74}$$

Support reactions are known to be vertical and equal to $500 \times 10^3[N]$ at each support.

Assume the model predicts:

$$\widehat{\mathbf{y}} = \begin{bmatrix} 500 \\ 525 \\ 510 \\ 520 \\ 525 \end{bmatrix} \times 10^6 \,\mathrm{N} \tag{3.75}$$

From Equation 3.70, the predicted member forces are:

- $F_1 = 476.19 \times 10^3$ N
- $F_2 = 500.00 \times 10^3$ N
- $F_3 = -686.92 \times 10^3$ N

- $F_4 = -700.37 \times 10^3$ N
- $F_5 = 1000.00 \times 10^3$ N

Table 3.4 summarises the unit orientation vectors $\mathbf{l}_{i,j}$.

i \ j	(1)	(2)	(3)	(4)	(5)
1	$\begin{bmatrix}1 & 0\end{bmatrix}^T$			$\begin{bmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}^T$	
2	$\begin{bmatrix}-1 & 0\end{bmatrix}^T$	$\begin{bmatrix}1 & 0\end{bmatrix}^T$			$\begin{bmatrix}0 & 1\end{bmatrix}^T$
3		$\begin{bmatrix}-1 & 0\end{bmatrix}^T$	$\begin{bmatrix}-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}^T$		
4			$\begin{bmatrix}\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{bmatrix}^T$	$\begin{bmatrix}-\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{bmatrix}^T$	$\begin{bmatrix}0 & -1\end{bmatrix}^T$

Table 3.4: *Orientation vectors $\mathbf{l}_{i,j}$ for each member as a function of the reference node.*

Computing the residuals for each node yields:

- $\boldsymbol{R}_1 = \begin{bmatrix}-19047.619 & 4761.905\end{bmatrix}^T$
- $\boldsymbol{R}_2 = \begin{bmatrix}23809.524 & 0\end{bmatrix}^T$
- $\boldsymbol{R}_3 = \begin{bmatrix}-14285.714 & 14285.714\end{bmatrix}^T$
- $\boldsymbol{R}_4 = \begin{bmatrix}9523.81 & -19047.619\end{bmatrix}^T$

The equilibrium loss is then given by:

$$\mathcal{L}_{\text{equilibrium}} = 226.76 \times 10^6 \tag{3.76}$$

Following hyperparameter tuning, the MLP is trained using the node equilibrium loss under conditions identical to those of the MSE-based model. The selected configuration is:

- **Loss function:** Node equilibrium loss
- **Architecture:** Three hidden layers with 40 neurons each
- **Activation function:** $\tanh$

As shown in Figure 3.29, the model exhibits a progressive improvement in validation accuracy throughout the training process.

However, closer inspection reveals that the model has not effectively learned the desired patterns. Several indicators support this claim:

- **Magnitude of the loss:** Upon convergence, the node equilibrium loss reaches an order of magnitude of 10^{13}, which is significantly higher than the expected

scale observed in the illustrative example (Equation 3.76), where the loss is on the order of 10^8. Such a discrepancy suggests that the loss has not been effectively minimised.

- **Negative R^2 score:** The appearance of a negative coefficient of determination indicates that the model performs worse than a naive predictor, always predicting the average axial rigidity EA of the training dataset, implying a failure to capture the underlying relationship in the data.

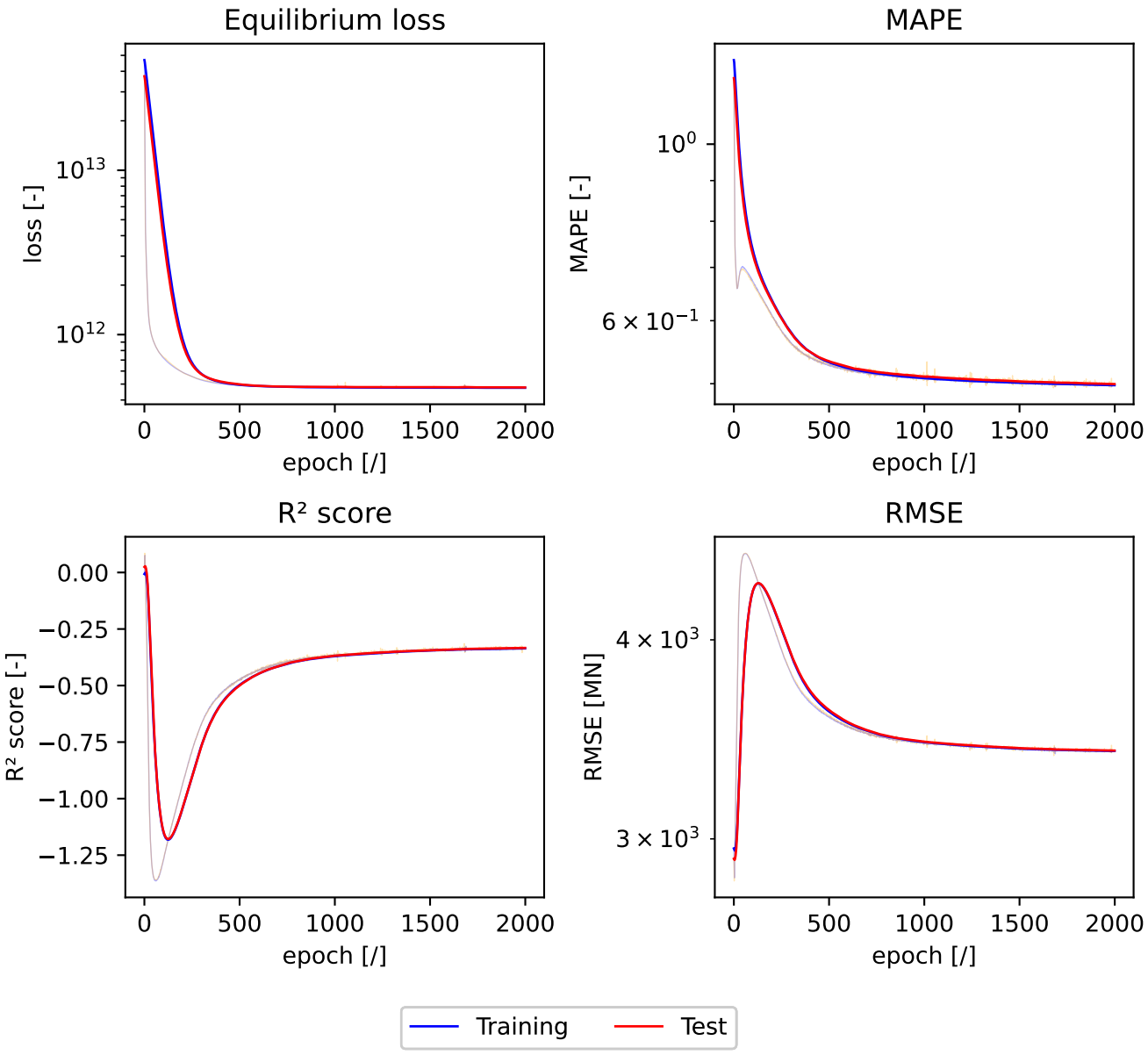


Figure 3.29: *Training performance of the MLP using the node equilibrium loss. The network comprises three hidden layers of 40 neurons each, trained with a learning rate of 10^{-3}, batch size of 512, and evaluated on a test set of size 512.*

While the loss formulation is grounded in physically valid principles, it appears that training the model using the node equilibrium loss alone does not result in mean-

ingful learning. Although multiple explanations could be considered, two intuitive hypotheses emerge:

1. **Insufficient global structure awareness:** The node equilibrium loss is highly localised, evaluating force balance at individual nodes. As a result, it may fail to capture the overall structural dependencies that are essential to predict global axial rigidity distributions. This may lead to optimisation that satisfies local equilibrium conditions without ensuring global consistency.
2. **Increased optimisation complexity:** The formulation of the loss incorporates numerous interacting member contributions, each dependent on predicted axial rigidities and measured strains. The resulting loss surface may be highly non-convex and complex, making it difficult for gradient-based optimisers to converge.

It is plausible that this loss could be more effective when used in conjunction with other loss functions. Such combinations may promote the learning of new patterns in the data and support improved generalisation in more complex structures. Doing so, this loss would also benefit from the stability in learning offered by the other loss functions. This hypothesis is further examined in Section 3.7.2.

Energy-Based Loss

When a structure undergoes deformation, it stores potential energy, analogous to the behaviour of a deformed spring. According to the principle of conservation of energy, the total stored potential energy U in the structure must be equal to the external work performed by the applied loads.

The total potential energy stored in the structure can be expressed as the sum of the energy stored in each member. This is given by:

$$U = \frac{1}{2} \sum_{i \in \mathrm{M}} \frac{E_i A_i}{L_i} \Delta L_i^2 = \frac{1}{2} \sum_{i \in \mathrm{M}} E_i A_i L_i \varepsilon_i^2 \tag{3.77}$$

where L_i denotes the length of member i, and ε_i is the corresponding axial strain.

The external work performed on the structure is defined by:

$$W_{\text{ext}} = \frac{1}{2} \sum_{i=1}^{n} u_i q_{\text{ext},i} = \frac{1}{2} \mathbf{u} \cdot \mathbf{q}_{\text{ext}}^T \tag{3.78}$$

To ensure physical consistency, it is required that:

$$U = W_{\text{ext}} \tag{3.79}$$

Using the model's predicted axial rigidities $\widehat{\mathbf{y}}$, an estimate of the potential energy can be computed as:

$$\widehat{U} = \frac{1}{2} \sum_{i \in \mathrm{M}} \widehat{y}_i L_i \varepsilon_i^2 \tag{3.80}$$

The corresponding loss function is defined as the mean squared error (MSE) between the estimated stored potential energy and the external work:

$$\mathcal{L}_{\text{energy}}\left(\widehat{\mathbf{y}}, \mathbf{u}, \mathbf{q}_{\text{ext}}\right) = \left(\widehat{U} - W_{\text{ext}}\right)^2 \tag{3.81}$$

Example

Consider again the reference truss used in the stiffness-based loss (Figure 3.26). The measured displacements and strains are:

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 4.761 \\ -27.754 \\ 9.524 \\ 0 \\ 4.762 \\ -18.231 \end{bmatrix} \times 10^{-3}\,\text{m}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} 0.0009524 \\ 0.0009524 \\ -0.0013469 \\ -0.0013469 \\ 0.0019048 \end{bmatrix} \tag{3.82}$$

Assume that the model predicts the axial rigidities:

$$\widehat{\mathbf{y}} = \begin{bmatrix} 500 \\ 525 \\ 510 \\ 520 \\ 525 \end{bmatrix} \times 10^6\,\text{N} \tag{3.83}$$

The external work, computed using Equation (3.78), is:

$$W_{\text{ext}} = 13877.20\,\text{J} \tag{3.84}$$

The estimated stored potential energy, using Equation (3.80), is computed as follows:

- $\widehat{U}_1 = 1133.78\,\text{J}$
- $\widehat{U}_2 = 1190.47\,\text{J}$
- $\widehat{U}_3 = 3270.97\,\text{J}$
- $\widehat{U}_4 = 3335.11\,\text{J}$
- $\widehat{U}_5 = 4761.90\,\text{J}$

Summing these contributions yields:

$$\widehat{U} = 13692.24\,\mathrm{J} \tag{3.85}$$

The resulting energy-based loss is:

$$\mathcal{L}_{\text{energy}} = (13877.20 - 13692.24)^2 = 34211.19 \tag{3.86}$$

Following hyperparameter tuning, the MLP is trained using the energy-based loss under conditions identical to those employed for the MSE-based model. The selected configuration is:

- **Loss function:** Energy-based loss
- **Architecture:** Three hidden layers with 40 neurons each
- **Activation function:** $\tanh$

As shown in Figure 3.30, the model exhibits progressive improvement in validation accuracy throughout the training process.

However, analysis reveals that the model fails to capture the desired patterns. In particular, the negative R^2 score observed on the test set indicates performance inferior to that of a naive model, which always predicts the average axial rigidity EA of the training dataset. This highlights a failure to generalise or meaningfully approximate the inverse mapping.

Despite being based on a physically grounded formulation, A model trained using only the energy-based loss does not appear to learn correctly to predict the expected output. Several plausible explanations arise:

1. **Weak coupling to the inverse problem:** While energy conservation is a fundamental physical principle, it may not provide sufficient information to resolve the inverse problem of identifying member axial rigidities from displacement and strain data alone.
2. **Non-uniqueness of solutions:** The structure of the loss function permits multiple distributions of EA values that satisfy the total energy condition. For example, the energy may be distributed equally across members or concentrated in a few, while still yielding low loss values. This indeterminacy likely hinders the learning process.

Given these limitations, it is plausible that this loss function may prove more effective when used in combination with other loss formulations, such as MSE or stiffness-based losses. Such hybrid strategies may provide additional stability and support more robust learning while benefiting from the novel physical constraints introduced by the energy-based approach. This hypothesis is explored further in Section 3.7.2.

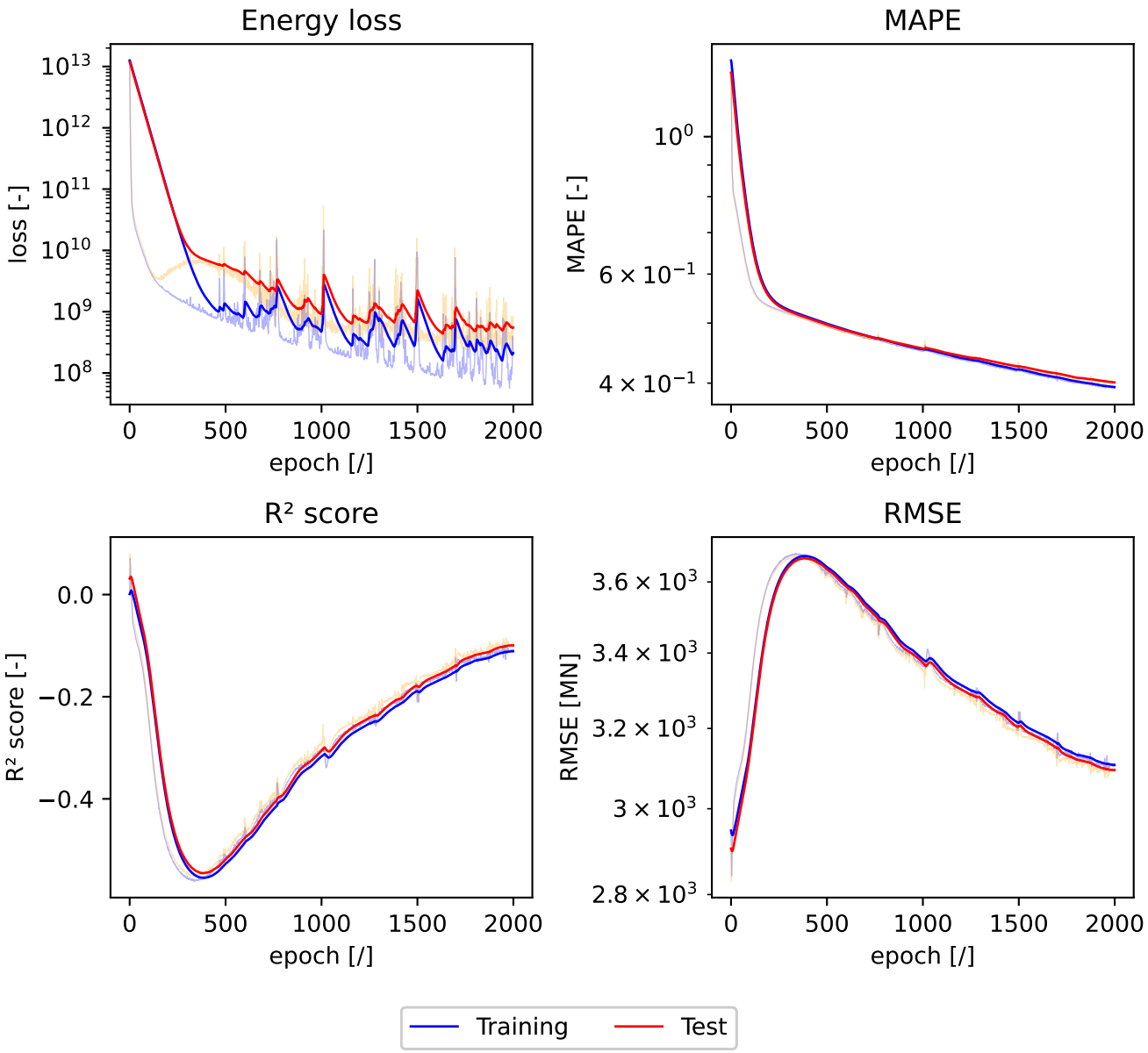


Figure 3.30: *Training performance of the MLP using the energy-based loss. The network consists of three hidden layers of 40 neurons each, trained with a learning rate of 10^{-3}, batch size of 512, and evaluated on a test set of size 512.*

Combined Loss

Although individual physics-based losses, such as the stiffness-based formulation, have demonstrated promising performance when used in isolation [22], the core philosophy behind Physics-Informed Neural Networks (PINNs) is to integrate physical knowledge into data-driven models. This integration is typically achieved through a hybrid loss formulation, where traditional losses such as the Mean Squared Error (MSE) are augmented with physics-informed components.

The general form of the combined loss function is a linear combination of the previously introduced loss terms:

$$\mathcal{L} = \lambda_{\text{MSE}}\mathcal{L}_{\text{MSE}} + \lambda_{\text{stiffness}}\mathcal{L}_{\text{stiffness}} + \lambda_{\text{equilibrium}}\mathcal{L}_{\text{equilibrium}} + \lambda_{\text{energy}}\mathcal{L}_{\text{energy}} \tag{3.87}$$

where λ_i are scalar weighting coefficients controlling the contribution of each individual loss to the overall objective.

However, a challenge arises due to the disparate orders of magnitude associated with each loss component. This discrepancy complicates the tuning of the λ_i parameters, as some losses may dominate the total loss irrespective of their relevance to the learning objective.

To mitigate this issue, the losses are normalised using their initial values before any training. Let $\mathcal{L}_i^{(0)}$ denote the initial value of the i-th loss term. The scaled hybrid loss is then expressed as:

$$\mathcal{L} = \lambda_{\text{MSE}}\frac{\mathcal{L}_{\text{MSE}}}{\mathcal{L}_{\text{MSE}}^{(0)}} + \lambda_{\text{stiffness}}\frac{\mathcal{L}_{\text{stiffness}}}{\mathcal{L}_{\text{stiffness}}^{(0)}} + \lambda_{\text{equilibrium}}\frac{\mathcal{L}_{\text{equilibrium}}}{\mathcal{L}_{\text{equilibrium}}^{(0)}} + \lambda_{\text{energy}}\frac{\mathcal{L}_{\text{energy}}}{\mathcal{L}_{\text{energy}}^{(0)}} \tag{3.88}$$

This normalisation scheme ensures that each term contributes comparably to the total loss at the start of training, thereby facilitating more balanced learning dynamics [42]. The use of such a scaled combination allows exploration of hybrid training strategies that simultaneously benefit from data-driven fitting and adherence to governing physical principles.

The behaviour and performance of this combined loss formulation will be explored in the subsequent sections.

3.7.2 Hyperparameter Tuning

The tuning process for the hybrid physics-informed model follows the same methodology as that employed for the standard MLP described in Section 3.6.1. However, an essential distinction lies in the inclusion of four additional hyperparameters: the weighting coefficients λ_i associated with the loss terms in Equation 3.88.

Given their interdependence, these coefficients exhibit a high degree of coupling, as altering one can significantly affect the influence of the others. Therefore, dedicated optimisation is required prior to reapplying the broader hyperparameter tuning strategy.

Loss Coefficient Optimisation

To determine appropriate values for the loss coefficients, the Optuna framework is employed, as in the previous classical model hyperparameter tuning described in Section 3.5.3. This approach leverages Bayesian optimisation to navigate the complex and interrelated search space efficiently.

Insights gained from training models using individual physics-based losses were incorporated to inform the fixed architectural configuration used during this phase. Specifically:

- The network architecture consists of a multilayer perceptron with three hidden layers of 40 neurons each, which demonstrated consistent performance across prior experiments.
- The activation function is chosen as $\tanh$, having yielded the best results across previous configurations.
- The learning rate is fixed at 10^{-3}, based on earlier tuning outcomes.

The training and validation procedures utilised a dataset comprising 4096 samples, evaluated using 5-fold cross-validation and a batch size of 512.

The resulting optimal loss coefficients obtained from Optuna are summarised in Table 3.5.

λ_{MSE}	$\lambda_{\text{stiffness}}$	$\lambda_{\text{equelibrium}}$	λ_{energy}
6.90572	0.79226	0.001062	0.001057

Table 3.5: *Optimal loss coefficients for the hybrid physics-informed loss function.*

It appears that the intuitions expressed previously are correct. The MSE Loss learns properly on its own, but is augmented by infusing the other loss. Notably, we observe that the stiffness loss benefit dominates the other two.

The results corroborate the previously stated intuitions regarding the behaviour of individual loss functions. Specifically, the model trained using the MSE loss alone is capable of effective learning and generalisation. However, performance is further enhanced by incorporating additional physics-based loss components.

Among the physics-informed terms, the stiffness-based loss exhibits the most pronounced contribution to overall model improvement. This observation aligns with prior results, where the use of stiffness loss, when employed independently, yielded meaningful learning progress. In contrast, the equilibrium- and energy-based losses, though physically sound, exert a comparatively weaker influence. Their primary utility appears to lie in complementing the dominant MSE and stiffness terms, potentially guiding the model towards alternative learning patterns and enhancing robustness under specific conditions.

Number of Hidden Layers and Neurons per Layer

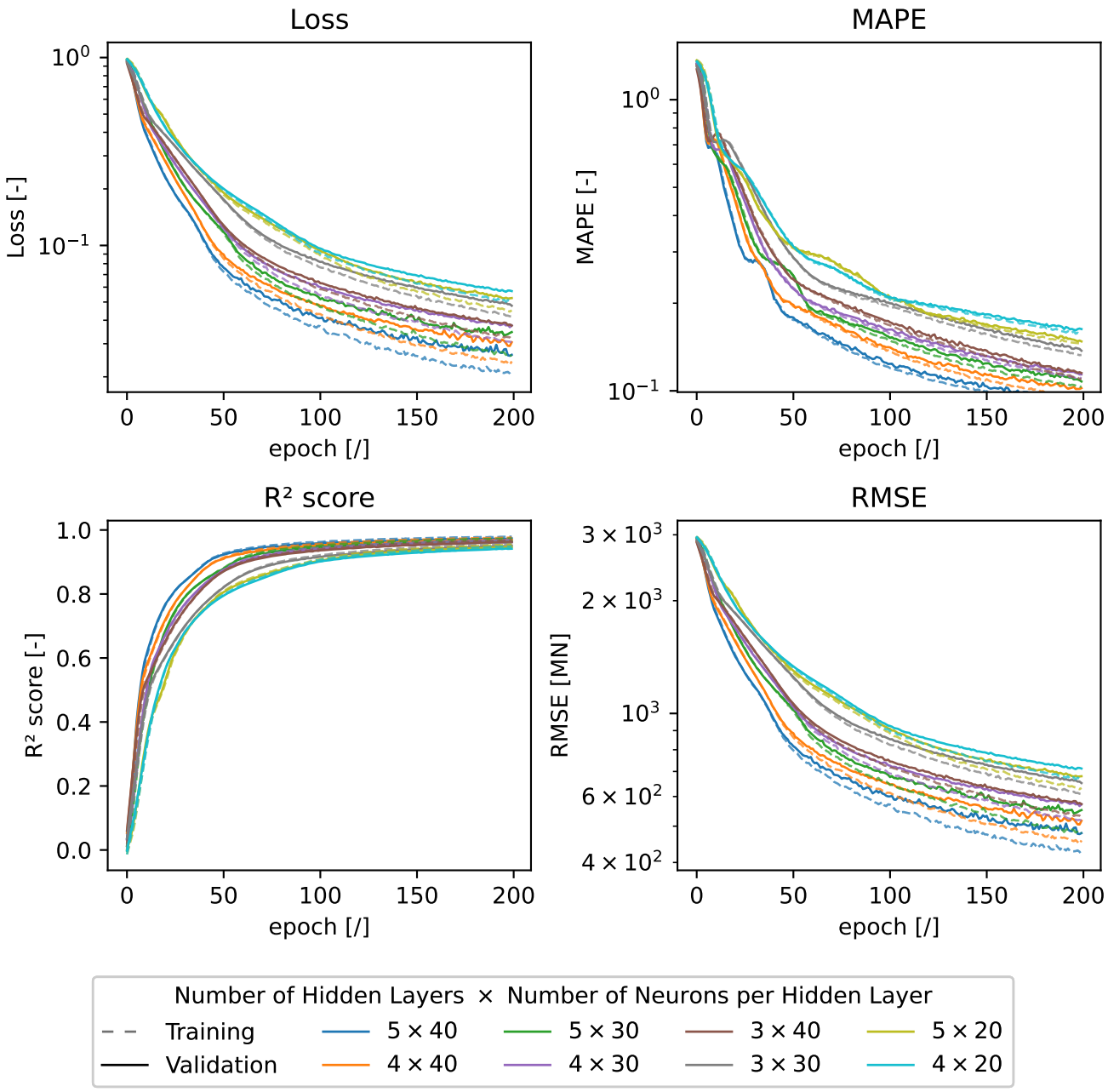


Figure 3.31: *Average metrics across 5-fold training with a physics-informed MLP using* $\tanh$ *activation and a learning rate of* 10^{-3}*, evaluated for varying numbers of hidden layers and neurons per layer.*

Figure 3.31 presents the performance metrics obtained from models of varying architectural capacities. Following the selection criteria established in Section 3.6.1, the optimal configuration corresponds to a network with five hidden layers, each comprising 40 neurons. This configuration demonstrated a favourable balance between predictive accuracy and model stability across validation folds.

Activation Function

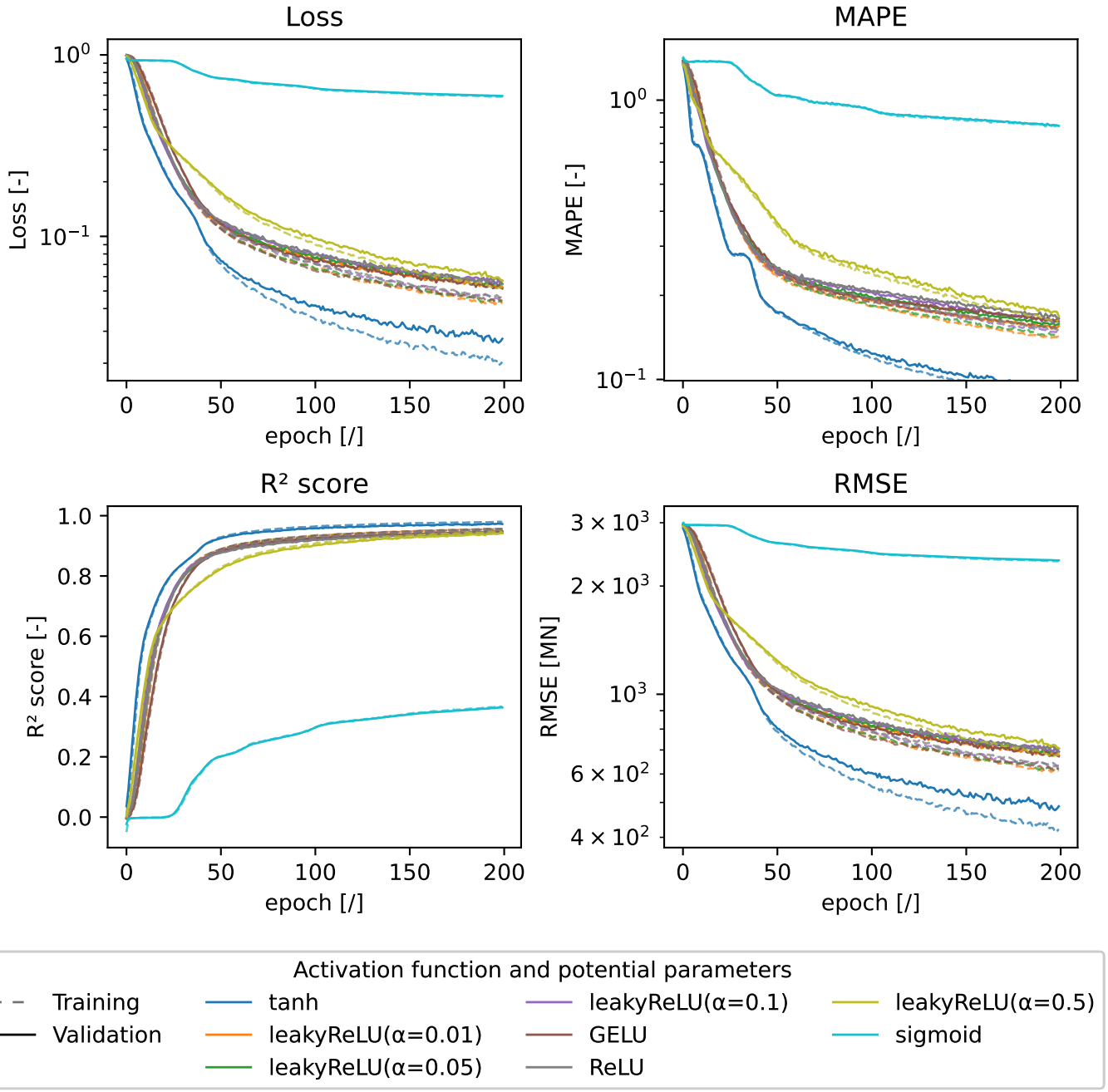


Figure 3.32: *Average metrics across 5-fold training with an MLP consisting of five hidden layers of 40 neurons each, trained with a learning rate of 10^{-3}, across various activation functions.*

As depicted in Figure 3.32, the tanh activation function consistently yields the best validation performance among all tested configurations. This outcome is consistent with previous observations in Section 3.6.1, where tanh was shown to facilitate effective learning across a range of loss formulations. It is therefore adopted as the activation function for the hybrid model.

Learning Rate

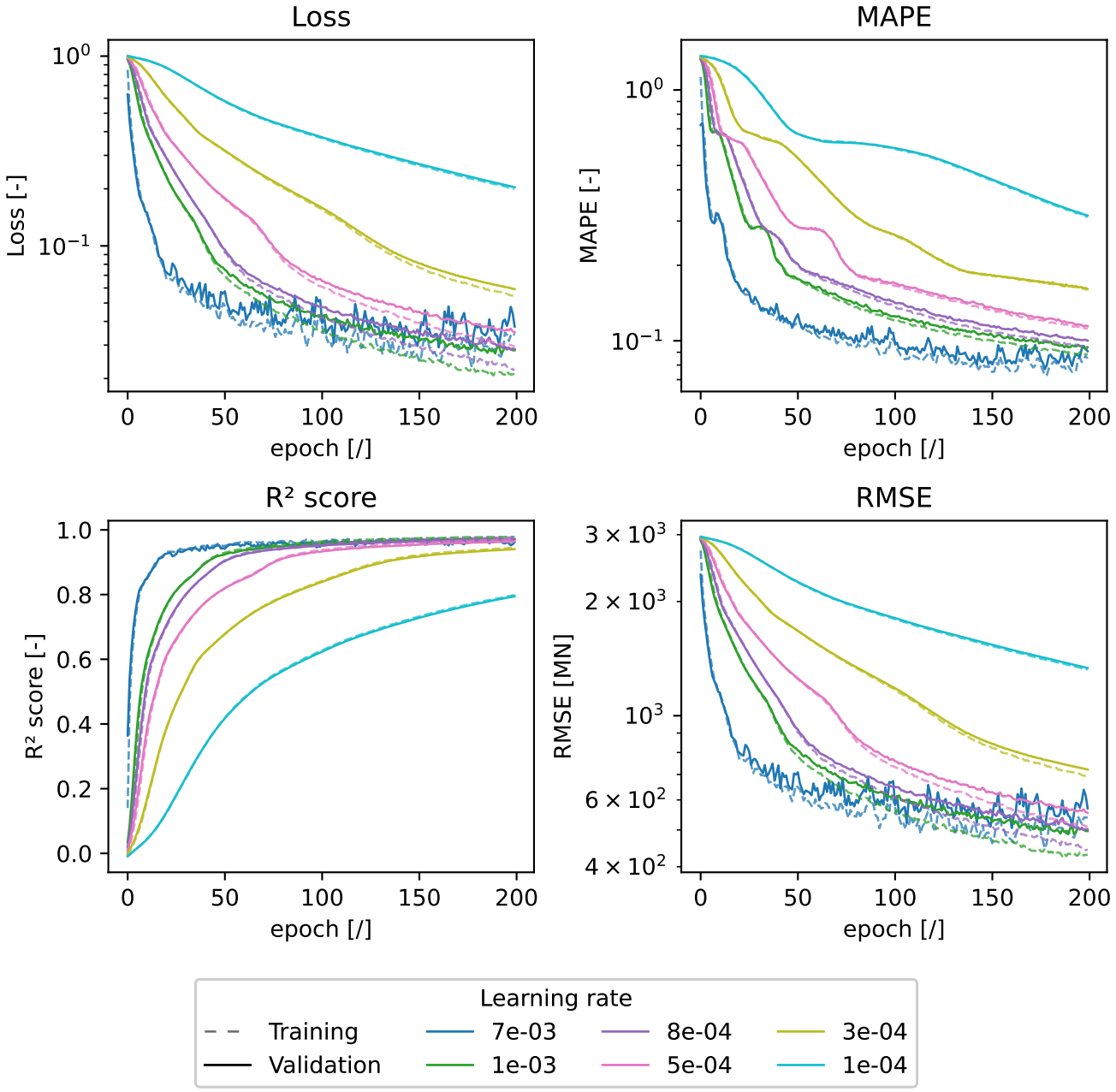


Figure 3.33: *Average metrics across 5-fold training with an MLP consisting of five hidden layers with 40 neurons each, using* tanh *activation, evaluated across various learning rates.*

Figure 3.33 compares the model performance across a range of learning rates. A learning rate of 10^{-3} provides the most favourable trade-off between convergence speed and stability. This setting promotes consistent descent in loss while avoiding oscillations or divergence in training. Consequently, this value is retained for training the final hybrid model.

3.7.3 Model Comparison

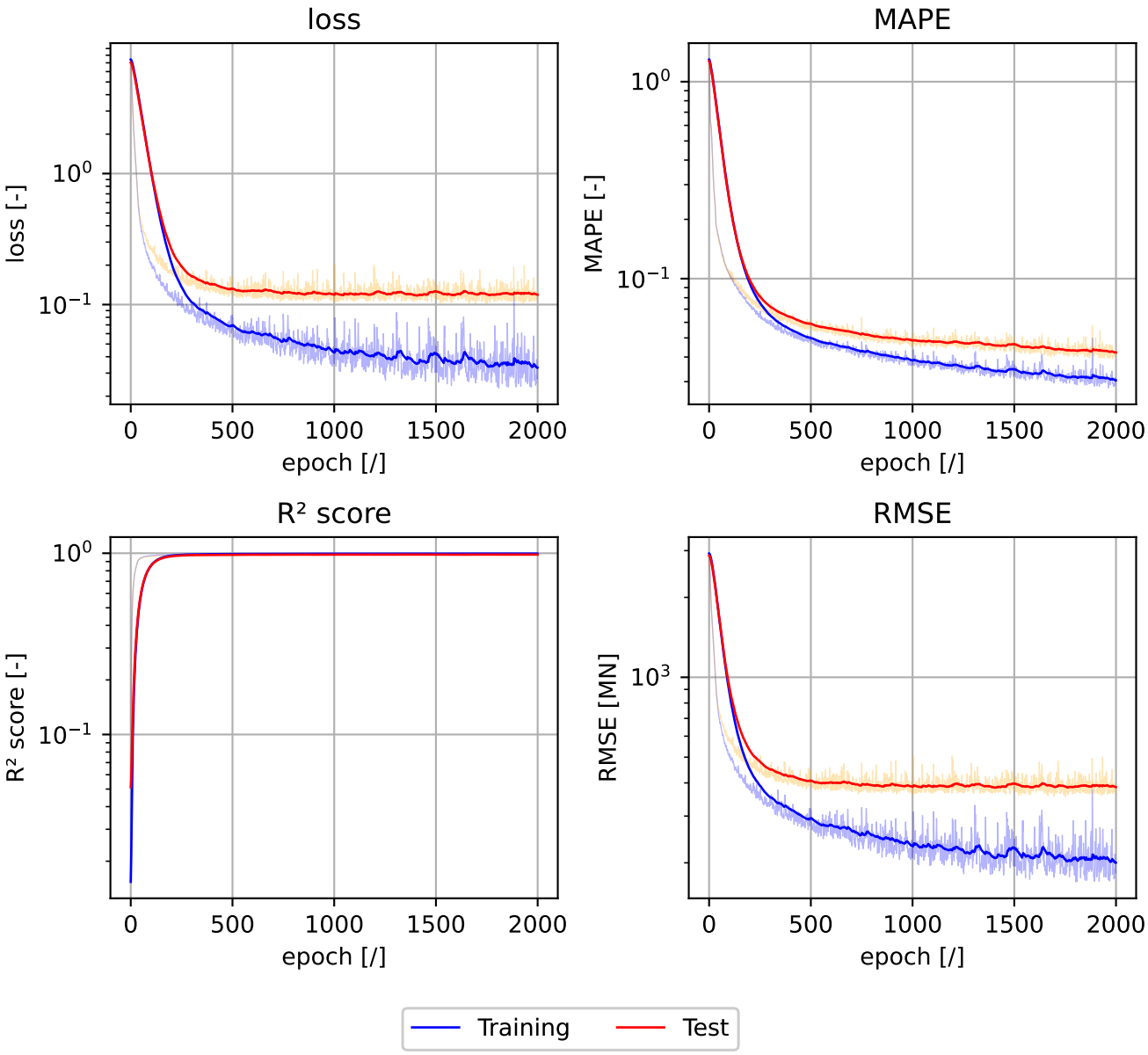


Figure 3.34: *Training and test metrics of an MLP with five hidden layers of 40 neurons each using the* tanh *activation function, trained on 4096 samples and evaluated on a test set of 512 samples with batch size 512.*

The model trained using the physics-informed loss exhibits performance comparable to that of the MSE-trained baseline. Figure 3.34 presents the training and test metrics over 2000 epochs. The model achieves a minimum test MAPE of approximately 4%, matching the performance of the best MLP trained using MSE alone. This result suggests that the hybrid loss formulation is capable of maintaining the predictive accuracy of data-driven approaches while incorporating domain knowledge.

To analyse the interaction between loss components, Figure 3.35 shows the evolution of each term within the hybrid loss function during training. The MSE term

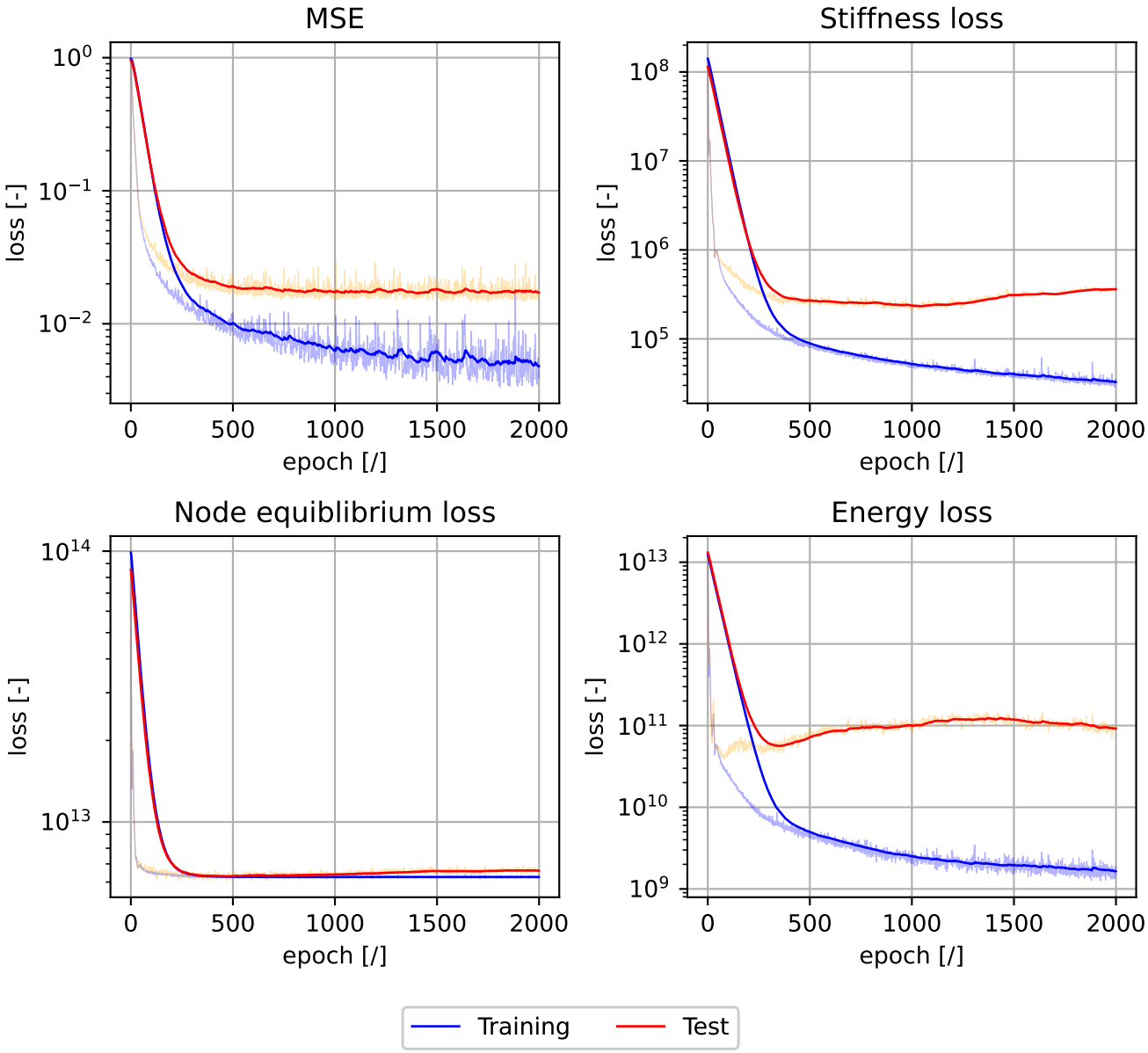


Figure 3.35: *Decomposition of the hybrid loss from Figure 3.34 into its individual components.*

exhibits a smooth and stable descent, indicating that it plays a central role in guiding the optimisation. The physics-based losses display more stability during training due to the effect of the MSE term, as suggested in the previous section. This behaviour contrasts with the performance of models trained using physics-informed losses alone (Figures 3.27, 3.29, and 3.30), where the absence of a stabilising loss results in poor convergence and inconsistent accuracy.

These findings support the hypothesis that although standalone physics-informed losses often struggle with convergence due to ill-conditioning or ambiguity in optimisation, their combination within a hybrid framework enables effective training. The combination of statistical and physical knowledge promotes convergence toward physically plausible solutions while maintaining predictive accuracy.

Despite this, the use of PINNs does not result in a significant improvement over the purely data-driven approach. As shown in Table 3.6, the physics-informed MLP achieves comparable test metrics but not superior to the baseline MLP. Specifically, while the MAPE values are similar, the physics-informed model exhibits a slightly lower R^2 score and higher RMSE.

Model	**MAPE [%]**	**R^2**	**RMSE [MN]**
Regression	116.02	0.0966	2805.35
Ridge	106.58	0.2591	2540.52
Lasso	114.42	0.1479	2724.48
KNN Regressor	29.18	0.6334	1787.18
Random Forest Regressor	27.03	0.7318	1528.51
MLP	**3.85**	**0.990**	**290.00**
Physics-Informed MLP	3.91	0.984	366.99

Table 3.6: *Summary of test metrics for all best models trained on a noiseless training dataset of size 4096 and tested on a noiseless test dataset of size 512.*

Furthermore, the robustness of the models under noisy input conditions was evaluated. As shown in Figure 3.36, the physics-informed model exhibits reduced resilience to noise when compared to the MLP trained with MSE. This behaviour is consistent with the expectation that physics-based constraints, while ensuring physical fidelity, are inherently less tolerant to noisy or inconsistent data, mirroring the limitations of the analytical model under similar conditions.

The parity plot for the Physics-Informed MLP, shown in Figure 3.37, together with the error distribution in Figure 3.38, indicates that this model achieves performance comparable to the MSE-based MLP.

In Figure 3.37, the predicted values $\widehat{EA}$ align closely with the identity line $\widehat{EA} = EA$, forming a compact, linear cluster. This alignment confirms that the Physics-Informed MLP is capable of delivering accurate predictions under noisy input conditions, with an error profile similar to that observed in the MSE-based model.

The corresponding error distribution across structural members, presented in Figure 3.38, shows no member-specific bias in prediction accuracy. Errors remain evenly distributed across the structure, further demonstrating that the model captures the be-

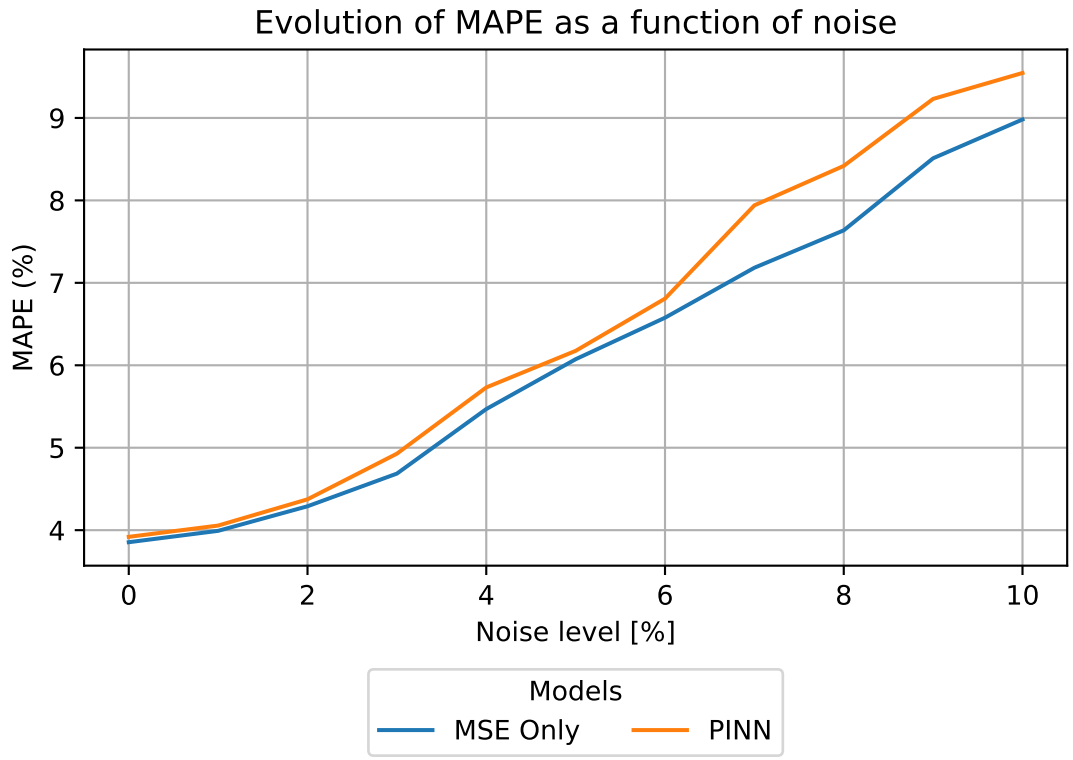


Figure 3.36: *Comparison of the MAPE for the MLP trained with physics-informed loss against the MLP trained with MSE under increasing levels of noise.*

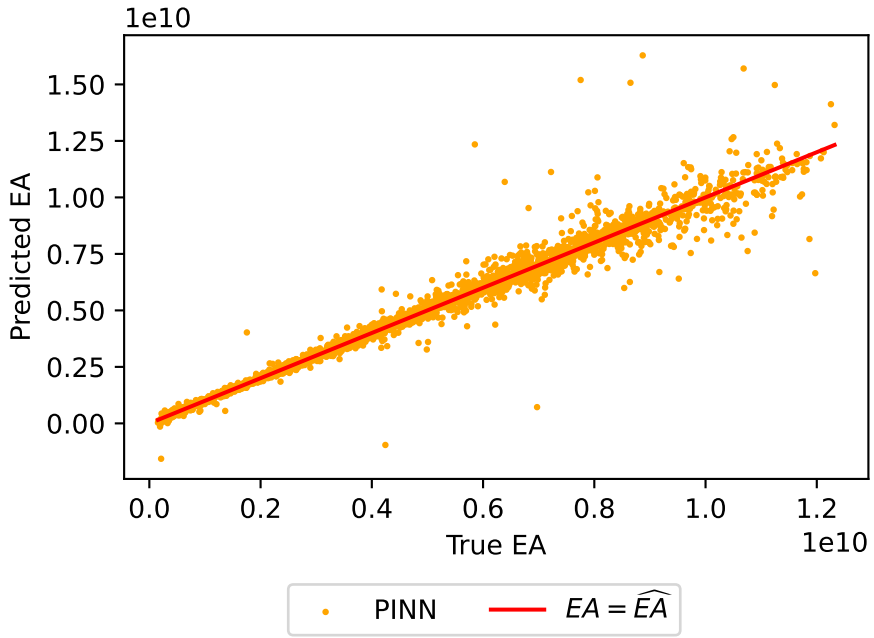


Figure 3.37: *Parity plot comparing the predicted axial rigidity $\widehat{EA}$ from the Physics-Informed MLP with the true axial rigidity EA for a test set of 512 examples subject to ±5% measurement noise.*

haviour of all members consistently and without overfitting to particular elements.

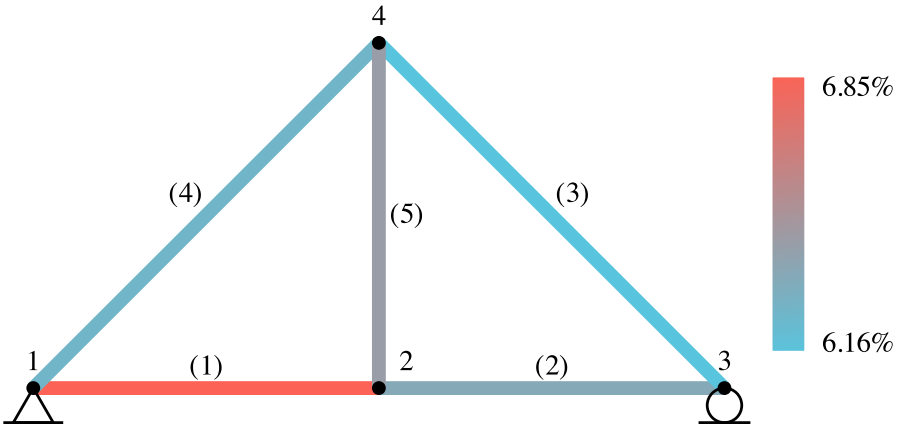


Figure 3.38: *Distribution of mean absolute error across members for a noisy test dataset with ±5% measurement noise.*

Nonetheless, the PINN approach exhibits several noteworthy properties. Examination of the parity plot comparing the predictions of the MSE-based MLP and the Physics-Informed MLP, shown in Figure 3.39, reveals that both models generally achieve comparable predictive accuracy on the test set, with measurement errors within ±2%.

However, this consistency does not hold when the true parameter values fall outside the range represented in the training dataset. We refer to that as *out of distribution* input. To illustrate this, consider a test dataset with content widely different from the training dataset (e.g., different orders of magnitude for loads, and EA. Figure 3.39 displays the parity plot for the prediction of both types of models.

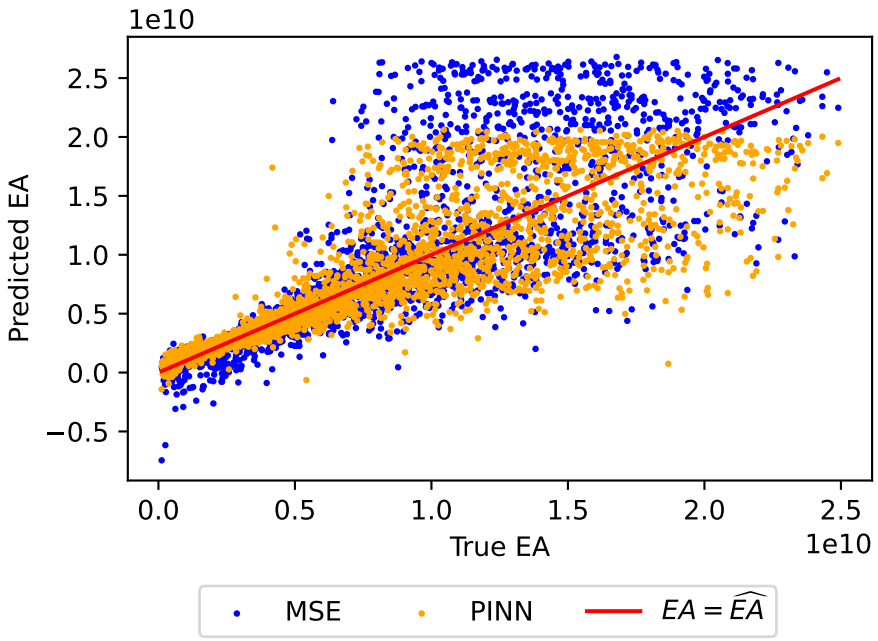


Figure 3.39: *Comparison of the true EA values and predicted $\widehat{EA}$ values on an out-of-distribution dataset with noise input of around 5%.*

We observe that the PINN model predictions are closer to the true EA than the prediction of the MSE-based MLP. For these unseen data, while the error is expectably higher than for input similar to the training set, the PINN model offers better results.

Moreover, it is essential to highlight that Physics-Informed Neural Networks (PINNs) are particularly advantageous in scenarios where training data are sparse, as the embedded physical knowledge can effectively compensate for the lack of observations [16]. To evaluate this behaviour, the same network was trained on progressively smaller subsets of the original dataset.

As shown in Figure 3.40, while training a model with a dataset of 4096 examples causes both models to converge toward a similar precision, for smaller training sets, the Physics-Informed MLP consistently outperforms the MSE-based MLP. This finding suggests that, while the physics-informed model does not offer a performance advantage over the MSE-based approach when trained on large datasets, it provides a clear benefit in data-scarce regimes.

In such cases, the inclusion of physics-based constraints enables the model to generalise more effectively from limited information, reinforcing the value of integrating domain knowledge into the learning process when data availability is restricted.

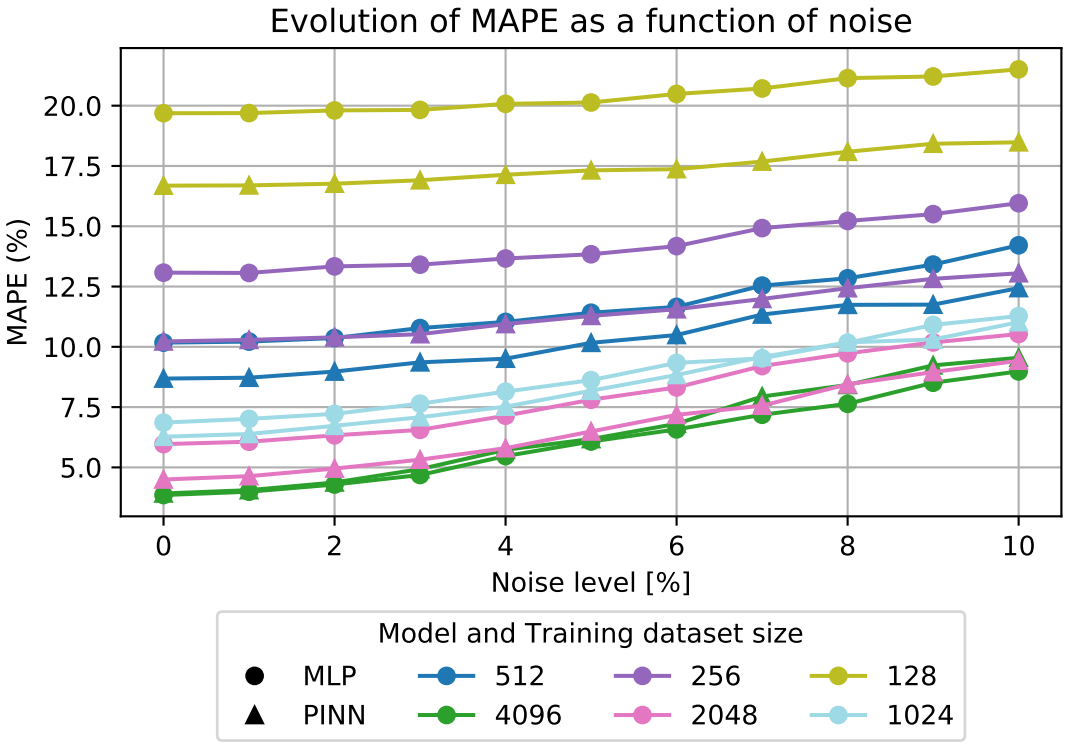


Figure 3.40: *Comparison of the MAPE for the MLP trained with physics-informed loss and the MLP trained with MSE loss across different training dataset sizes.*

3.8 Conclusion

Through the detailed exploration presented in this chapter, we demonstrate the practical implementation and evaluation of physics-informed neural networks in comparison to traditional machine learning models. The physics-informed neural networks approach successfully integrates physical principles directly into the learning process, resulting in improved predictive accuracy and robustness, particularly in scenarios involving noisy data.

The results illustrate that while physics-informed neural networks do not outperform traditional MSE-trained multilayer perceptrons on large datasets, they exhibit superior performance when applied to smaller datasets. This characteristic makes physics-informed neural networks particularly advantageous for engineering applications where extensive data acquisition can be challenging or expensive. The application of PINN also provides better results for structural behavior that is not displayed in the training dataset, suggesting better extrapolation capabilities. Further exploration of this advantage will be detailed in subsequent chapters.

This chapter serves as a comprehensive explanation of the theoretical concepts through a simplified and accessible example. Building upon this foundation, the next chapter will delve into the dataset generation process, while the final chapter will demonstrate the application of these concepts to a more complex structural scenario.

4

Dataset Generation

4.1 Dataset Sources

A dataset is defined as a structured collection of data in which each entry represents an individual observation or measurement derived from a particular process or experiment. Each observation is characterised by a predefined set of variables, the nature and interpretation of which depend on the underlying context of data acquisition.

With the proliferation of data-driven approaches and the rapid development of Big Data technologies, the volume and availability of open-access datasets have expanded significantly. Several large-scale public repositories now exist to support a wide range of applications. Notable examples include:

- **Common Crawl** [43], a web archive containing over 250 billion pages, commonly used for natural language processing and web mining tasks.
- **Data Commons** [44], an open-source platform developed by Google that integrates a vast number of datasets from various domains, enabling seamless data analysis and exploration.
- **ImageNet** [45], a large-scale visual database comprising approximately 14 million annotated images, extensively used for benchmarking image classification and object recognition models.

Despite the growing accessibility of open datasets across scientific disciplines, publicly available datasets tailored to construction and civil engineering remain scarce. In particular, numerical results obtained from structural experiments are infrequently shared in academic publications, limiting reproducibility and data reuse within the domain.

Nonetheless, several initiatives have begun to address this limitation by releasing curated datasets focused on structural diagnostics, particularly within the context of computer vision. Notable examples include:

- **SDNet2018** [46], a dataset comprising 56,000 images (256×256 pixels) of concrete

surfaces, annotated for the presence or absence of cracks with widths ranging from 0.6mm to 25mm.

- **CODEBRIM** [47], a labelled dataset designed for surface damage classification on concrete bridges, including defects such as cracks, corrosion, exposed reinforcement, and spalling.
- **CSSC** [48], a dataset consisting of approximately 47,000 annotated images capturing cracked and spalled regions of concrete structures.
- **DeepCrack** [49], containing around 7,000 annotated images of concrete piers and decks, widely used in crack segmentation research.

Although these resources provide valuable benchmarks for visual inspection and surface damage classification, their scope is limited to image-based defect detection. They do not incorporate numerical data or mechanical properties, nor are they derived from controlled experimental procedures such as flexural or axial testing. As a result, they are not directly applicable to problems requiring detailed structural analysis or inverse identification of material parameters.

Consequently, this research investigates two strategies for dataset generation. Firstly, using experimental data. Secondly, and more extensively, running numerical simulations to generate synthetic datasets tailored to the inverse problem.

4.1.1 Experimental Data

Experimental data represent a direct and valuable source of information for structural analysis. Such data capture the real-world behavior of materials and structures, accounting for inherent complexities such as material heterogeneity, construction tolerances, and other uncontrolled variables. They also include measurement uncertainty and noise introduced by the specific instrumentation and methodologies employed.

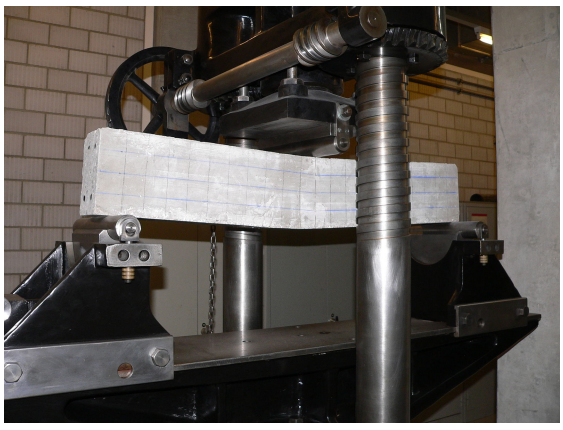


Figure 4.1: *Three-point flexural test* [50]

Although experimentally obtained data are highly valuable, exclusive reliance on them presents several limitations:

- **Limited availability:** Acquiring sufficient quantities of experimental data for the

training of machine learning models is challenging. Model performance is often strongly dependent on the size and diversity of the dataset.

- **Inconsistencies across data sources:** Combining data from different experiments frequently results in inconsistencies. Certain datasets may lack key information, while others may include additional or incompatible features, complicating integration and comparison.
- **Preprocessing requirements:** Considerable effort is often needed to standardize, clean, and harmonize experimental datasets. This process can be both time- and resource-intensive, particularly when dealing with diverse experimental protocols and formats.

Due to these challenges, the exclusive use of experimental data is impractical. In such cases, synthetic data generation provides a viable and scalable alternative [25, 24].

4.1.2 Synthetic Dataset

Synthetic data generation offers an alternative approach for producing datasets, either as fully artificial constructs or as augmentations of limited empirical data. This method has been applied in various structural contexts, such as the augmentation of crack patterns in masonry walls, as demonstrated by Boerema [51].

In structural analysis, data can be generated through simulation tools, where arbitrary structural models are defined, analyzed using finite element or comparable numerical methods, and the resulting outputs systematically extracted.

Although synthetic data does not capture the full variability observed in real-world conditions, this approach provides several advantages relevant to structural applications:

- **Absence of measurement noise:** Simulation results are derived from deterministic models governed by established physical laws. Provided the model assumptions hold, outputs are exact, consistent, and reproducible.
- **Scalability:** The volume of data generated is not constrained by physical limitations. Large datasets can be produced to meet the demands of data-driven methods, particularly in machine learning.
- **Expanded feature coverage:** Unlike experimental data, synthetic data generation is not restricted by fabrication or testing constraints. Structural configurations can be systematically varied to explore a broad spectrum of parameters and their influence on response variables.

In light of these benefits, and the requirement for large-scale, standardized data, this study employs a fully synthetic dataset for model development and training.

4.2 Case Study

This study focuses on a representative yet non-trivial structural typology: a statically indeterminate Pratt truss bridge with a total span of 60 m and a height of 7.5 m vertically loaded on the deck nodes (i.e., the surface of a bridge, where users cross the structure). The loading configuration is highly flexible: vertical loads may be applied to any subset of nodes 2 through 8 (e.g., simultaneously on all nodes, only on nodes 2 and 5, exclusively on node 3, or even with no loads at all). This allows the model to account for the full spectrum of possible load distributions over the deck. The structure is supported at both extremities by pinned supports, resulting in a hyperstatic configuration in which the number of constraints exceeds the available degrees of freedom. This configuration is deliberately adopted to investigate whether the presence of structural redundancy introduces difficulties in the learning process, as the example from chapter 3 was not hyperstatic. The geometry and loading scheme of the structure are illustrated in Figure 4.2.

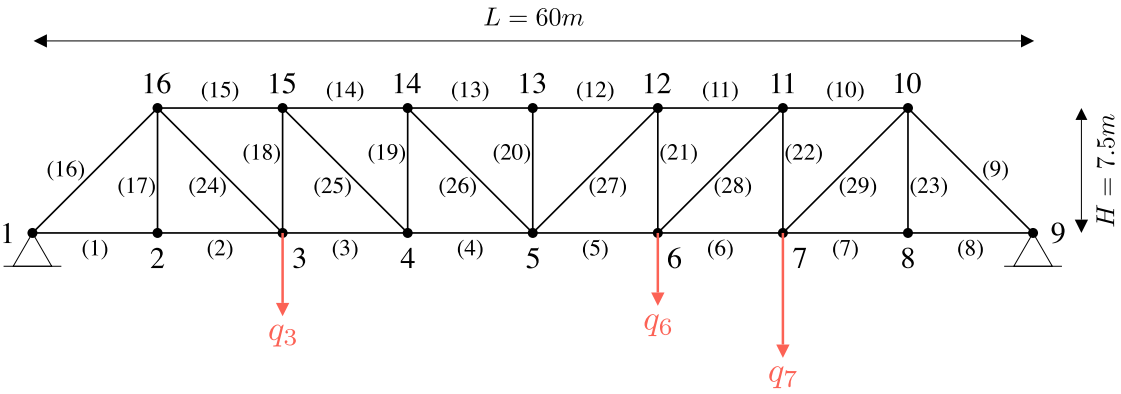


Figure 4.2: *Definition of geometric layout, member indices, and representative external loading configuration.*

The choice of a Pratt truss is motivated by its advantageous structural properties. This configuration is widely used in bridge engineering and is well understood from both analytical and numerical perspectives. It exhibits a regular and symmetric layout, which facilitates systematic parametrisation and scalability through simple adjustments to the number of panels. Such structural regularity is particularly valuable in the context of synthetic data generation and pattern recognition tasks in machine learning, where well-structured and interpretable behaviour is desirable.

Under conventional loading conditions, the internal force distribution in the Pratt truss is relatively predictable: the top chords experience compressive forces while the bottom chords are subjected to tension. This consistent mechanical response simplifies the verification and validation of model predictions.

Given these characteristics, the Pratt truss serves as an effective benchmark structure for the development and testing of data-driven models. While the relationships between input variables and structural response remain complex and high-dimensional, they are sufficiently structured to permit systematic investigation.

The structural configuration includes both geometric and material parameters, summarised in Table 4.1. Key geometrical properties, such as the total span, height, and panel count, are fixed. Likewise, the material density is set to represent structural steel.

Parameter	Description	Value / Range	Unit
L	Total span of the bridge	60	m
H	Height of the truss	7.5	m
n_{panels}	Number of panels in the truss	8	–
ρ	Mass density of the material	7850	kg/m^3
$f_{y,d}$	Design yield stress	205	MPa
A_i	Cross-sectional area of member i	[10, 500]	cm^2
E_i	Young's modulus of member i	[150, 250]	GPa
$q_{\text{ext},i}$	Vertical load at node i	[-1000, 0]	kN

Table 4.1: *Structural parameters of the Pratt truss and sampling bounds.*

In this model, the self-weight of the structure is computed separately from the externally applied loads. The nodal self-weight w_i is calculated as a function of the geometry and material properties of the elements connected to node i:

$$w_i = \frac{1}{2} \sum_{j \in C_i} L_j A_j \rho g \tag{4.1}$$

Here, C_i represents the set of members connected to node i, L_j and A_j denote the length and cross-sectional area of member j, respectively, and $g = 9.81\,\text{m/s}^2$ is the gravitational acceleration constant.

Figure 4.3 illustrates the total load acting on the structure, comprising both the self-weight and the external nodal loads sampled according to the specified ranges.

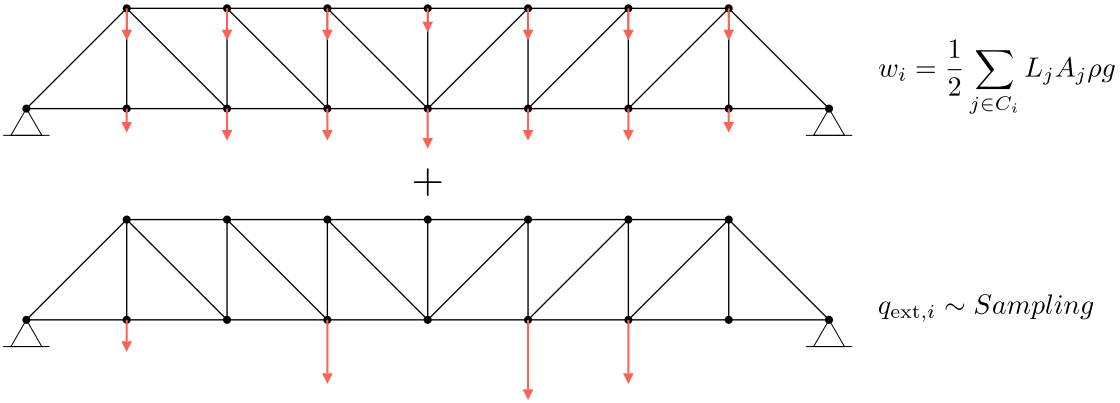


Figure 4.3: *Total loading applied to the structure, showing the superposition of self-weight and sampled external nodal loads.*

The ranges of values for the parameters in this setup are chosen empirically within plausible engineering bounds, allowing for a wide range of structural responses while maintaining physical realism and attention to the European construction norms on

material strength and structural deformation. To be more precise, the range bounds are motivated by the Eurocode 0 (EN 1990) [52] and Eurocode 1 (EN 1991) [53].

4.3 Generation Procedure

The development of a fully synthetic dataset necessitates the creation and simulation of numerous structural configurations, each designed to capture a diverse and representative range of responses. This process is implemented through a systematic four-steps process:

1. **Parameter Sampling:** In this initial step, the values for the structural parameters are selected. This defines the specific instance of the structure to be modelled and included in the dataset. For example, a configuration may involve a bridge with uniform member cross-sectional areas $A = 10\,\text{cm}^2$, a uniform Young's modulus $E = 200\,\text{GPa}$, and applied external nodal loads $q_{\text{ext},2} = -1000\,\text{kN}$ and $q_{\text{ext},5} = -1500\,\text{kN}$. This stage establishes the geometric, material, and loading conditions that characterise the synthetic structure. For better dataset representativity, the ranges in which parameters are sampled (see Table 4.1) shall be as close as possible to the actual value using an educated estimation of the civil engineer.
2. **Structural Modelling:** Once the parameters are defined, the structure is encoded in a finite element analysis (FEA) software. This involves creating a numerical model that accurately reflects the geometry, boundary conditions, material properties, and loading scheme.
3. **Simulation and Data Extraction:** The structural analysis is executed using the FEA software, yielding a set of results corresponding to the displacements, internal forces, strains, and reactions. These outputs are then post-processed to extract relevant features required for the dataset. Care is taken to ensure consistency across simulations and completeness of the extracted data.
4. **Data Storage:** The final step involves formatting and storing the extracted results in a structured manner. The data is organized to facilitate access, interpretability, and integration with machine learning workflows. This includes standardising units, ensuring consistent indexing, and preserving metadata such as input parameters and simulation identifiers.

4.3.1 Sampling the Parameters

Parameter sampling refers to the process of selecting values for structural variables from predefined probability distributions within a constrained *sampling space*. As the geometric configuration of the structure is fixed, the sampling process is restricted to the non-geometric parameters, which include material properties and loading conditions.

Each member i of the structure is assigned a cross-sectional area A_i and a Young's modulus E_i, while each node j is associated with an external load $q_{\text{ext},j}$ and a binary indicator $b_j \in \{\text{True}, \text{False}\}$ signifying whether that node is effectively loaded.

Example

Consider that the sampled external loads $\mathbf{q}_{\text{ext}}$ and load activation flag $\mathbf{b}$ are

$$\mathbf{q}_{\text{ext}} = \begin{bmatrix} -100 & -300 & -40 & -50 & -630 & -250 & -120 \end{bmatrix}^T \tag{4.2}$$

$$\mathbf{b} = \begin{bmatrix} \text{True} & \text{True} & \text{False} & \text{True} & \text{False} & \text{False} & \text{True} \end{bmatrix}^T \tag{4.3}$$

The effective external load thus becomes

$$\mathbf{q}_{\text{ext}} = \begin{bmatrix} -100 & -300 & 0 & -50 & 0 & 0 & -120 \end{bmatrix}^T \tag{4.4}$$

The inclusion of the binary variable b_j is crucial, as it facilitates the generation of diverse loading configurations. Without it, randomly assigning near-zero loads to the majority of nodes while preserving significant loads on others would be statistically improbable.

As shown in Table 4.2, the dimensionality of the sampling space reaches up to 75, accounting for the 29 cross-sectional areas, 29 Young's moduli, and 14 load-related variables (7 magnitudes and 7 activation flags).

Parameter	Number of Dimensions
A_i	29
E_i	29
q_i	7
b_i	7
Total	**75**

Table 4.2: *Dimensionality of the sampling space*

An ideal dataset should provide a representative coverage of the sampling space to ensure the model generalises well across varying structural scenarios. However, the process of uniformly sampling high-dimensional spaces encounters significant challenges due to the *curse of dimensionality*. As illustrated in Figure 4.4, the number of possible combinations increases exponentially with the number of dimensions, rendering comprehensive sampling computationally infeasible.

Example

Consider the analogy of rolling fair six-sided dice. A single die yields 6 possible outcomes. With two dice, the total combinations increase to $6^2 = 36$, and with three dice, to $6^3 = 216$. Extending this to 75 dice results in approximately 2.3×10^{58} combinations. Even storing a mere 1% of this total would require an astronomical amount of memory.

This combinatorial explosion illustrates the curse of dimensionality, where even modest increases in dimensionality cause the sampling space to become prohibitively large, particularly when dealing with continuous-valued parameters, such as those used in truss modelling.

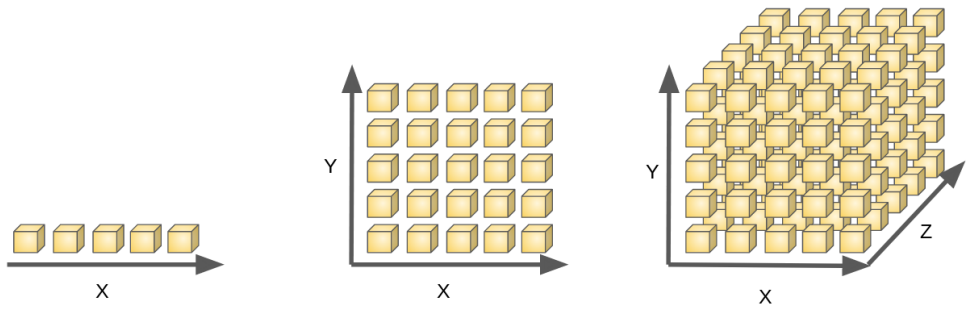


Figure 4.4: *Illustration of the curse of dimensionality* [54].

To overcome this challenge, appropriate sampling strategies must be employed. Figure 4.5 demonstrates the effect of various sampling methods using a two-dimensional example over the domain $[0, 1]^2$ with 400 sampled points.

A naive approach would involve grid-based sampling, where each parameter range is discretised (e.g., sampling Young's modulus at intervals of 20 GPa). While systematic, this approach is inherently limited in its ability to capture nuanced variations, as the diversity in values is limited, and becomes intractable when extended to high dimensions due to the high number of possible combinations.

Alternatively, random uniform sampling allows values to be drawn independently from uniform distributions. Because we can perform an arbitrary number of samplings, this method avoids the exponential scaling of grid-based strategies. It also provides greater diversity. However, in high-dimensional settings, this approach suffers from uneven coverage, often forming clusters and leaving significant portions of the space unsampled.

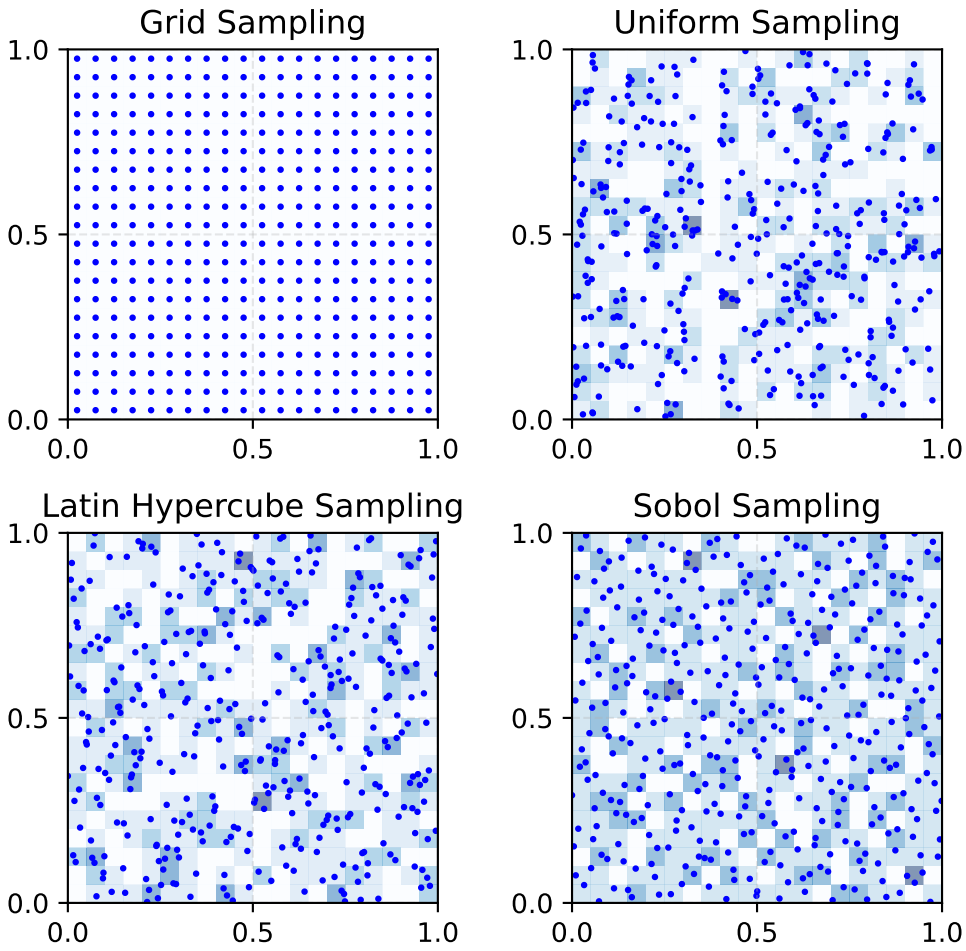


Figure 4.5: *Comparison of sampling methods over* $[0, 1]^2$ *with 400 samples. Point density is illustrated by background colour.*

To enhance sampling efficiency and uniformity, more sophisticated techniques are applied:

- **Latin Hypercube Sampling (LHS):** A stratified approach that divides the range of each variable into equal-probability intervals and ensures that each interval is sampled. LHS improves coverage in lower dimensions and reduces clustering compared to pure random sampling [55].
- **Sobol Sequences:** A class of quasi-random low-discrepancy sequences designed for uniform space-filling properties in high-dimensional spaces. These sequences are deterministic and empirically provide superior convergence rates for integration and optimisation tasks. [56]

In this study, Sobol sequences are selected as the preferred sampling strategy. Their ability to uniformly populate high-dimensional spaces makes them especially well-suited for structural parameter sampling, ensuring balanced representation across the complex and expansive sampling space.

Although Sobol sequences appear to provide a better general approach due to their absence of parametrization, extension of this generation process with the LHS method could benefit from the stratified approach given the specifics of the problem.

4.3.2 Modeling the Structure

Although the theoretical framework outlined in Section 3.2 provides the necessary basis for truss analysis, it does not incorporate computational optimisations essential for efficient large-scale structural simulations. As a result, the dataset generation pipeline leverages the open-source finite element solver *OpenSees* [57], interfaced through its Python wrapper, *OpenSeesPy* [58].

The use of *OpenSeesPy* enables seamless integration of structural modelling, simulation, and data extraction within a unified Python-based environment. This allows for full parametric control over the structure generation process and facilitates automation across thousands of model instances.

Within this framework, the Pratt trusses are instantiated using the sampled parameters and modelled in accordance with the linear truss assumptions described in Chapter 3. These assumptions include axial force–only members, linear-elastic material behaviour, and idealised pin connections. This abstraction permits the representation of the structure as a set of bar elements with nodal connections.

4.3.3 Solving and Extracting the Results

Once the structural model is instantiated, the full load is applied in a single load step, consistent with the assumptions of static linear elastic analysis. Following the solution of the structural system, a series of output quantities is extracted directly from the simulation results. Additional quantities are then computed to enhance the dataset and support subsequent machine learning tasks. These derived quantities include the member strains ε, elongations ΔL, the axial forces F, and the global stiffness matrix $\mathbf{K}$, all of which are analytically linked to the system's internal state.

Table 4.3 summarises the full set of features stored for each simulation instance. These include geometric descriptors, structural response quantities, and connectivity information.

Although not all stored features are directly required for the current study, their inclusion ensures both extensibility for future research and complete reproducibility of results. This approach supports the development of alternative modelling strategies or training objectives without requiring regeneration of the dataset.

4.3.4 Storing the Results

Proper storage of the dataset is essential to ensure experiment reproducibility and practical usability, especially when dealing with large-scale data that cannot be generated dynamically during model training.

Feature	Description	Dimension
L	Truss span	1
H	Truss height	1
n_{panels}	Number of panels in the truss	1
ρ	Material mass density	1
$\mathbf{c}$	Cartesian coordinates of nodes	2×8
$\mathbf{u}$	Displacements of nodes	2×8
$\mathbf{q}$	Applied external loads on deck nodes	2×8
$\boldsymbol{\varepsilon}$	Strain of the members	29
F_j	Axial force in member j	29
$\mathbf{l}_0$	Initial length of members	29
$\Delta\mathbf{l}$	Elongation of members	29
$\mathbf{K}$	Global stiffness matrix of the structure	$(2 \times 8)^2$
$\mathbf{C}$	Connectivity matrix	2×29

Table 4.3: *Stored output features for each simulation*

A standard format for numerical data is comma-separated values (CSV), which represents tabular data in plain text. However, the CSV format is not well suited for storing structured data such as vectors or matrices and presents limitations in terms of storage efficiency and read/write performance.

To address these issues, the dataset is stored using the Hierarchical Data Format version 5 (HDF5) [59]. This format provides compact binary storage, higher precision for floating-point numbers, and significantly improved read/write performance. Additionally, HDF5 supports hierarchical organisation, enabling the storage of complex structures, such as nested arrays or labelled datasets, in a logically organised and efficient manner. This makes HDF5 a natural choice for managing large volumes of structured data in scientific computing and machine learning workflows.

4.3.5 Generation Framework

To support the automated and reproducible generation of large-scale structural datasets, a modular and extensible software framework has been developed. The framework is designed to be both geometry-agnostic and analysis-agnostic, facilitating application across a wide range of structural typologies and numerical solvers with minimal code modifications.

The architecture of the framework is structured around four core abstract classes, each corresponding to a critical phase of the data generation pipeline:

1. **AbstractStructure:** Encapsulates the definition of structural geometry, material properties, and boundary conditions. This class serves as the foundation for specifying the physical configuration of the structure to be analysed.
2. **AbstractAnalysis:** Manages the simulation configuration, including the selection of the analysis type (e.g., linear static) and solver settings. This abstraction allows for seamless integration with OpenSees.

3. **AbstractGenerator:** Governs the data generation logic. It includes parameter sampling, model instantiation, execution of the simulation, postprocessing of results, and data validation steps.
4. **AbstractHDF5Dataset:** Defines the output data schema and handles the structured storage of results in the HDF5 format. This class ensures that all generated datasets follow a consistent and standardised layout.

All generation parameters—including input distributions, output specifications, and dataset size—are defined via human-readable configuration files. These are designed for transparency and usability, and several examples are included within the project repository.

Throughout this research, several structural typologies were implemented using this framework. Each new typology required fewer than 200 lines of Python code, underscoring the framework's simplicity, modularity, and adaptability. Figure 4.6 illustrates a selection of representative structural configurations generated using the tool.

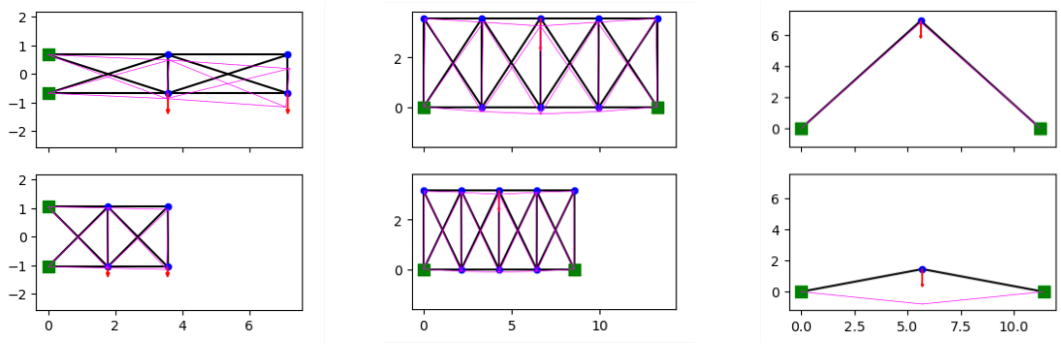


Figure 4.6: *Examples of structural configurations generated using the framework*

Comprehensive documentation, configuration templates, and implementation examples are provided in the associated codebase to facilitate future reuse and extension.

4.4 Overview of the Synthetic Dataset

Following the description of the generation process, this section presents an overview of the resulting synthetic datasets.

Three categories of datasets have been generated, each corresponding to progressively more complex prediction tasks. To reflect common loading scenarios in bridge design, 20% of all load cases were assigned a uniform load magnitude across nodes, while the remaining 80% involved random load magnitudes. The goal is to provide some structure in the dataset. This repartition is displayed in Figure 4.7.

4.4.1 Dataset with Member-Specific Axial Rigidity

The most complex dataset introduces unique axial rigidity values for each member, departing from typical design conventions. While this scenario is less common in initial

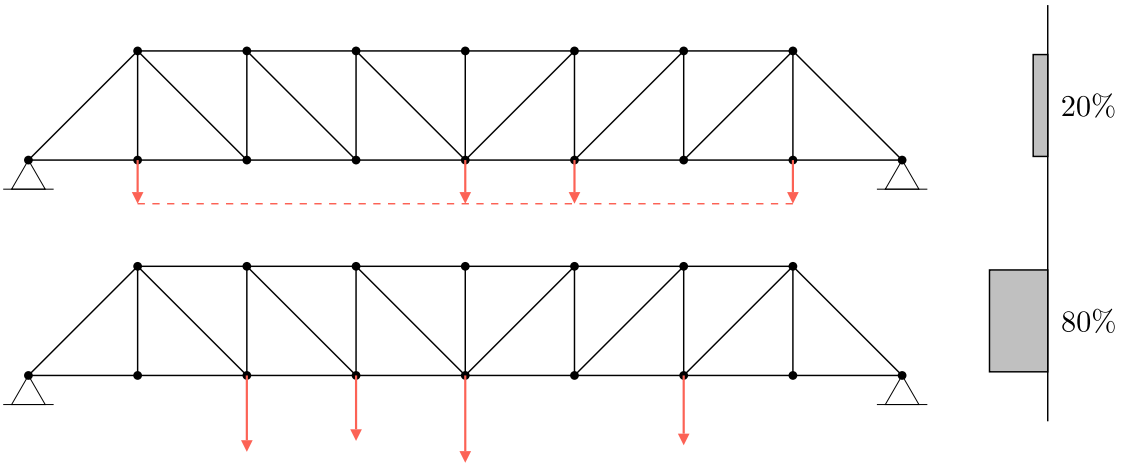


Figure 4.7: *Distribution of load case categories across all datasets*

design phases, it may become relevant for the evaluation of existing structures that have undergone aging, deterioration, or repair.

To preserve some structure within the dataset, 5% of the samples use uniform member properties, and an additional 5% apply category-based member properties. This balance enables models trained on the dataset to generalize across a wide range of structural conditions.

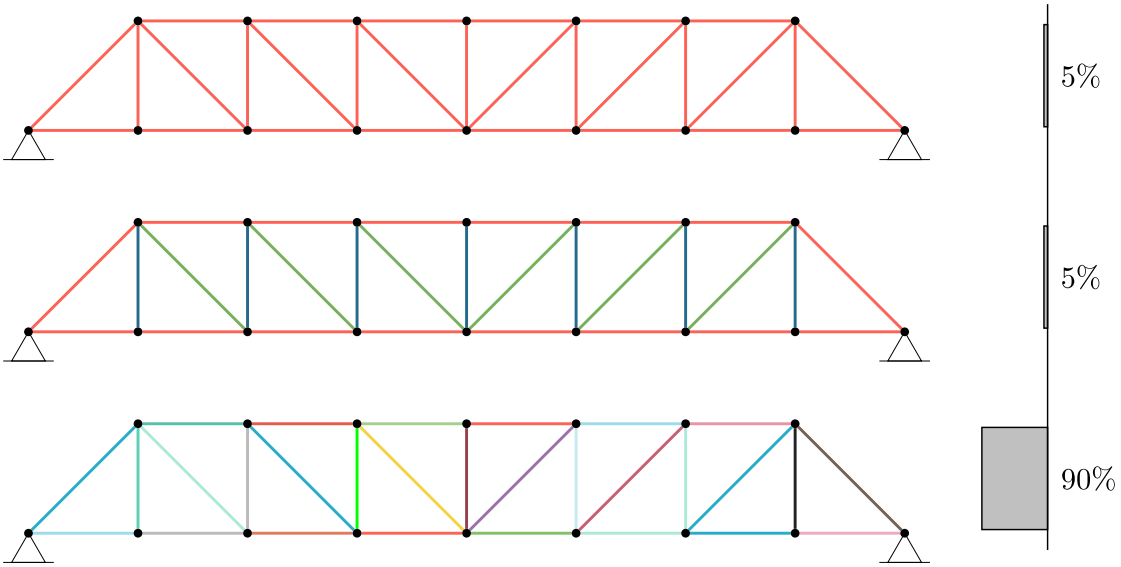


Figure 4.8: *Distribution of targets across the member-specific dataset*

4.4.2 Dataset with Category-Based Axial Rigidity

In this dataset, axial rigidity is assigned based on structural member category. Given the predictable force distribution in Pratt trusses, an economical and practical design strategy often involves assigning different cross-sections to horizontal, vertical, and diagonal members. Accordingly, three distinct axial rigidities are sampled and assigned based on member orientation.

To introduce variability and maintain diversity in the dataset, 10% of the samples

retain uniform axial rigidity across all members.

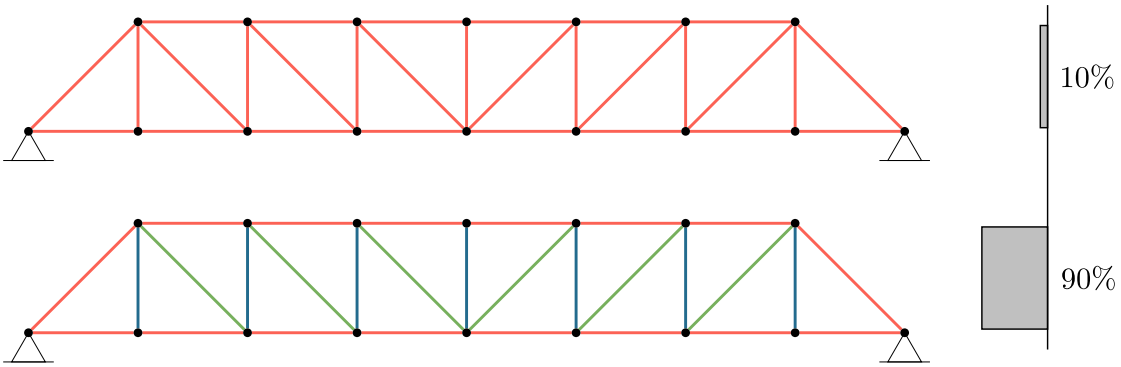


Figure 4.9: *Distribution of targets across the category-based dataset*

5
Application to the Pratt Truss

This chapter applies the MLP-based predictive frameworks developed in Chapter 3 to the synthetic Pratt truss dataset generated in Chapter 4. Two categories of models are evaluated:

- A *purely data-driven* model, trained exclusively with the mean squared error (MSE) loss function.
- A *physics-informed* model, integrating domain knowledge through the hybrid loss formulation developed in Section 3.7.

As demonstrated in Chapter 3.7, training with physics-based losses in isolation does not achieve the level of accuracy required for practical application. Therefore, the physics-informed model adopts a hybrid objective function, combining the MSE term with three physics-based terms:

$$\mathcal{L} = \lambda_{\text{MSE}}\mathcal{L}_{\text{MSE}} + \lambda_{\text{stiffness}}\mathcal{L}_{\text{stiffness}} + \lambda_{\text{equilibrium}}\mathcal{L}_{\text{equilibrium}} + \lambda_{\text{energy}}\mathcal{L}_{\text{energy}}, \tag{5.1}$$

where λ_i are scalar weighting coefficients governing the relative influence of each term in the overall training process.

The process of hyperparameter tuning follows the same methodological framework established in Section 3.6.1 for the MSE-based MLP and Section 3.7.2 for the physics-informed MLP. For brevity, the detailed iterative search, intermediate configurations, and tuning diagnostics are omitted here. Instead, only the final, optimised hyperparameters are presented in the subsequent sections, allowing the discussion to focus on the comparative performance of the models rather than the mechanics of parameter search. All trainings are done using a training dataset of size 16384 and a batch size of 512 and the Adam Optimizer.

For reference, the dataset utilised in this chapter is illustrated in Figure 5.1, and comprises the following quantities:

- nodal displacements $\mathbf{u}$;

- applied nodal loads $\mathbf{q}$;
- axial strains in each member $\boldsymbol{\varepsilon}$;
- axial rigidities of the members EA.

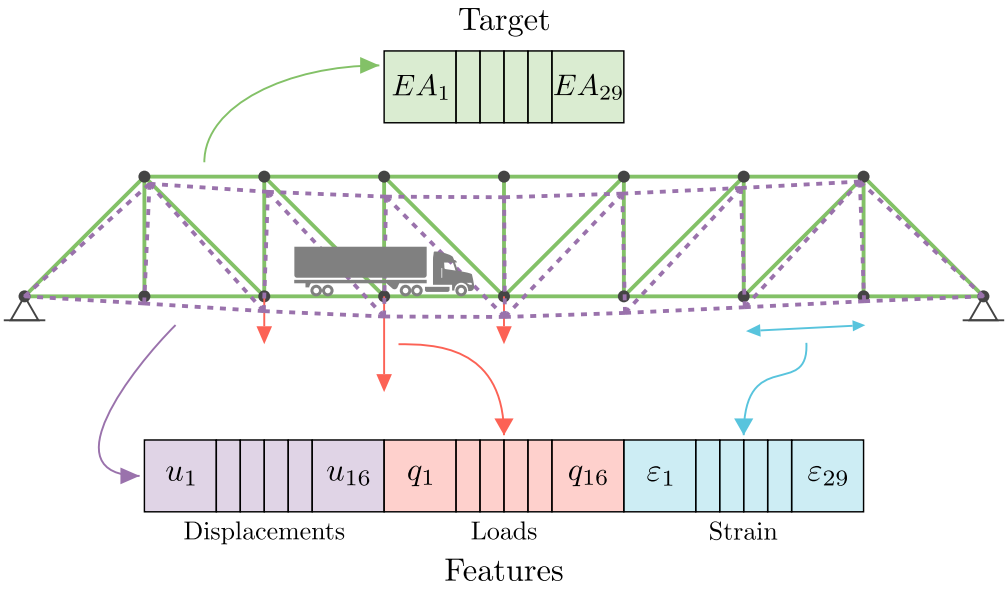


Figure 5.1: *Illustration of the dataset composition.*

5.1 Predicting Axial Rigidity for Each Member

This section investigates the capacity of a multilayer perceptron (MLP) to predict the axial rigidity EA of each structural member individually, using the member-level dataset generated in Section 4.4.1. The number of outputs, thus, is 29 since the structure contains that many members.

5.1.1 MSE-Based Model

The first model is trained using the mean squared error (MSE) loss function exclusively. The MLP architecture comprises three hidden layers, each with 120 neurons, and employs the $\tanh$ activation function. The learning rate is set to 10^{-3}.

Figure 5.2 presents the training metrics using a training dataset containing 16384 examples with batches of size 512. On a test dataset of 8192 unseen examples, the model achieves a MAPE of 16.6% and an R^2 score of 0.917. These metrics indicate that the model captures the underlying relationships with reasonable fidelity, while being unable to produce high-accuracy predictions.

To assess the robustness of the model against observational uncertainty, controlled noise is introduced into the input features. The noise model follows the formulation in Section 3.4, where a multiplicative perturbation of intensity α is applied to both nodal displacements $\mathbf{u}$ and external loads $\mathbf{q}$:

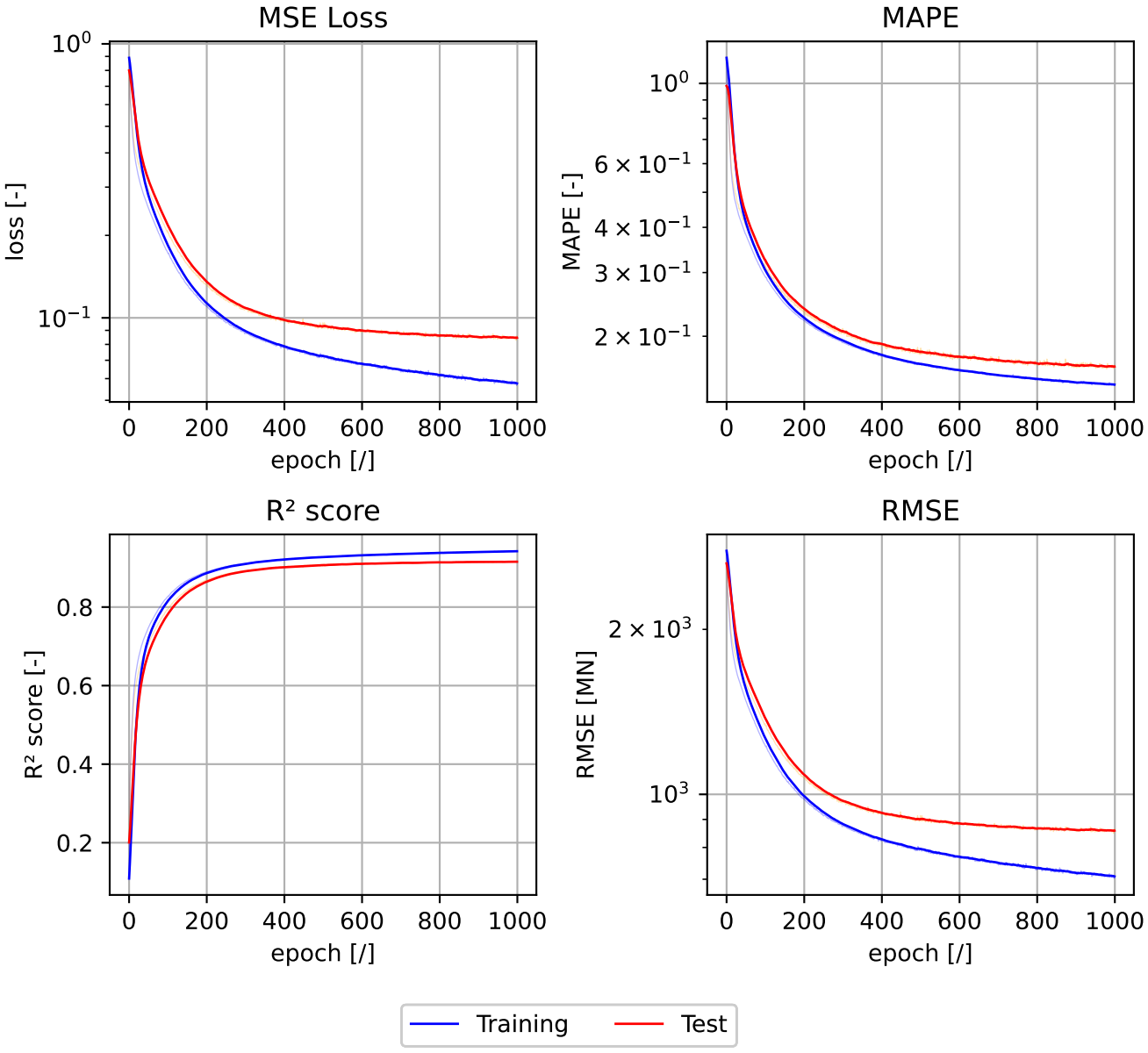


Figure 5.2: *Training and test metrics of an MLP trained with MSE loss with three hidden layers of 120 neurons each using the* tanh *activation function, trained on 16384 samples and evaluated on a test set of 8192 samples with batch size 512.*

$$\widetilde{\mathbf{u}} = \boldsymbol{\alpha}_u \odot \mathbf{u}, \quad \widetilde{\mathbf{q}} = \boldsymbol{\alpha}_q \odot \mathbf{q}, \tag{5.2}$$

where $\odot$ denotes the elementwise product, and the noise terms are drawn from a normal distribution:

$$\alpha_i \sim \mathcal{N}(\mu = 1, \sigma = \frac{\epsilon}{2}). \tag{5.3}$$

Figure 5.3 shows the model's sensitivity to increasing noise levels. The degradation in performance is approximately linear with respect to the noise amplitude, indicating predictable behaviour under uncertainty.

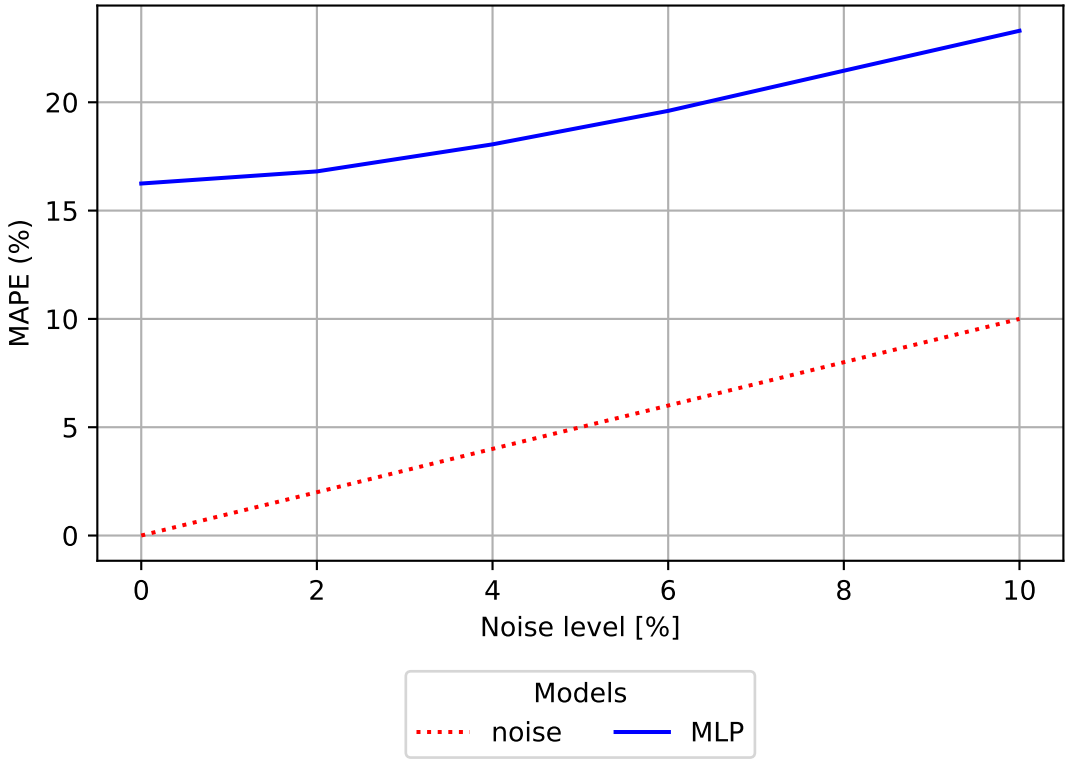


Figure 5.3: *Model sensitivity to increasing noise levels in displacement and load inputs.*

To further interpret the model's performance, the parity plot in Figure 5.4 visualises the predicted axial rigidity $\widehat{EA}$ against the true values EA for a test set with ±5% measurement noise. The approximate linear alignment of points suggests that the model predictions are globally accurate. However, noticeable dispersion indicates that while trends are captured, certain predictions deviate significantly from their ground truth values.

From Figure 5.4, the prediction error for the MSE-based model on a test dataset with ±5% measurement noise can be estimated at approximately 18% MAPE. The distribution of these errors across individual members, shown in Figure 5.5, indicates that the overall error remains below this expected threshold. However, a subset of horizontal members, specifically members 1 to 8, exhibit noticeably larger errors. Among these,

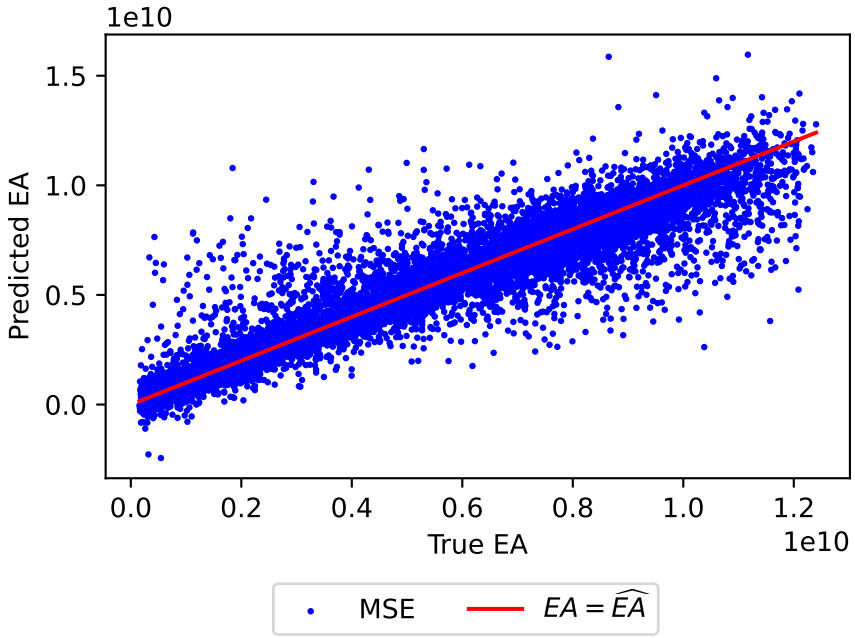


Figure 5.4: *Parity plot comparing predicted versus true axial rigidities for a noisy test set* (±5% *measurement error*).

members 3 and 6 present an apparent anomaly, with substantially higher prediction errors than the rest.

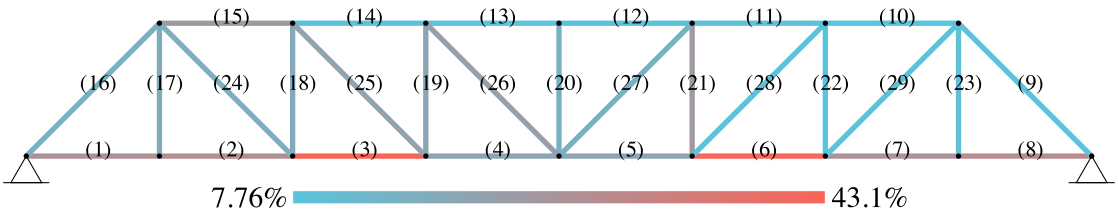


Figure 5.5: *Distribution of MAPE across individual members for a test set with* ±5% *measurement noise.*

This localisation of error is likely attributable to geometric redundancy within the structure, specifically the alignment of members 1 to 8 along a straight load path between the supports. A similar phenomenon was identified in the analytical study of Section 3.4. For instance, under the geometric linearity hypothesis (Section 3.2), the connection between members 1 and 2 possesses no vertical stiffness, causing them to behave analytically as a single continuous element. This configuration hinders the model's ability to differentiate their respective axial rigidity. Such ambiguity in the estimation of members 1 and 2 can propagate to member 3, leading to elevated prediction errors. Owing to the structural symmetry, the same reasoning applies to member 6.

5.1.2 Physics-Informed Model

The physics-informed model adopts a multilayer perceptron (MLP) architecture comprising three hidden layers of 120 neurons each, with $\tanh$ as the activation function. The learning rate is fixed at 10^{-3}. This model is trained using a hybrid loss function that incorporates both data-driven and physics-based components. The weighting coefficients for each loss term are listed in Table 5.1.

λ_{MSE}	$\lambda_{\text{stiffness}}$	$\lambda_{\text{equilibrium}}$	λ_{energy}
0.800	0.763	0.297	0.108

Table 5.1: *Optimal weighting coefficients for the hybrid physics-informed loss function.*

The model is trained on 16,384 examples with batches of size 512 and evaluated on a test set of 8192 unseen instances. As shown in Figure 5.6, the model achieves a mean absolute percentage error (MAPE) of 15.9% and a coefficient of determination $R^2 = 0.918$. These metrics indicate that the model captures meaningful structural relationships from the input data.

The robustness of the model is examined by evaluating its sensitivity to varying levels of noise in the displacement and load inputs. The noise model follows the formulation established earlier, applying multiplicative Gaussian noise to the input features. As shown in Figure 5.7, the error increases approximately linearly with the amplitude of the noise, suggesting consistent and predictable behaviour under uncertainty.

To visualise the alignment between predicted and true values of axial rigidity, a parity plot is presented in Figure 5.8. The points align broadly along the identity line, indicating that the model captures the underlying trends in the data. Nevertheless, some dispersion persists, reflecting the complexity of the inverse prediction task in the presence of measurement uncertainty.

Finally, Figure 5.9 illustrates the spatial distribution of the prediction error across the individual members of the structure. The error remains high for members 3 and 6, similar to the previous model.

These results demonstrate that the hybrid physics-informed approach yields stable and interpretable predictions, while embedding structural principles directly into the training process.

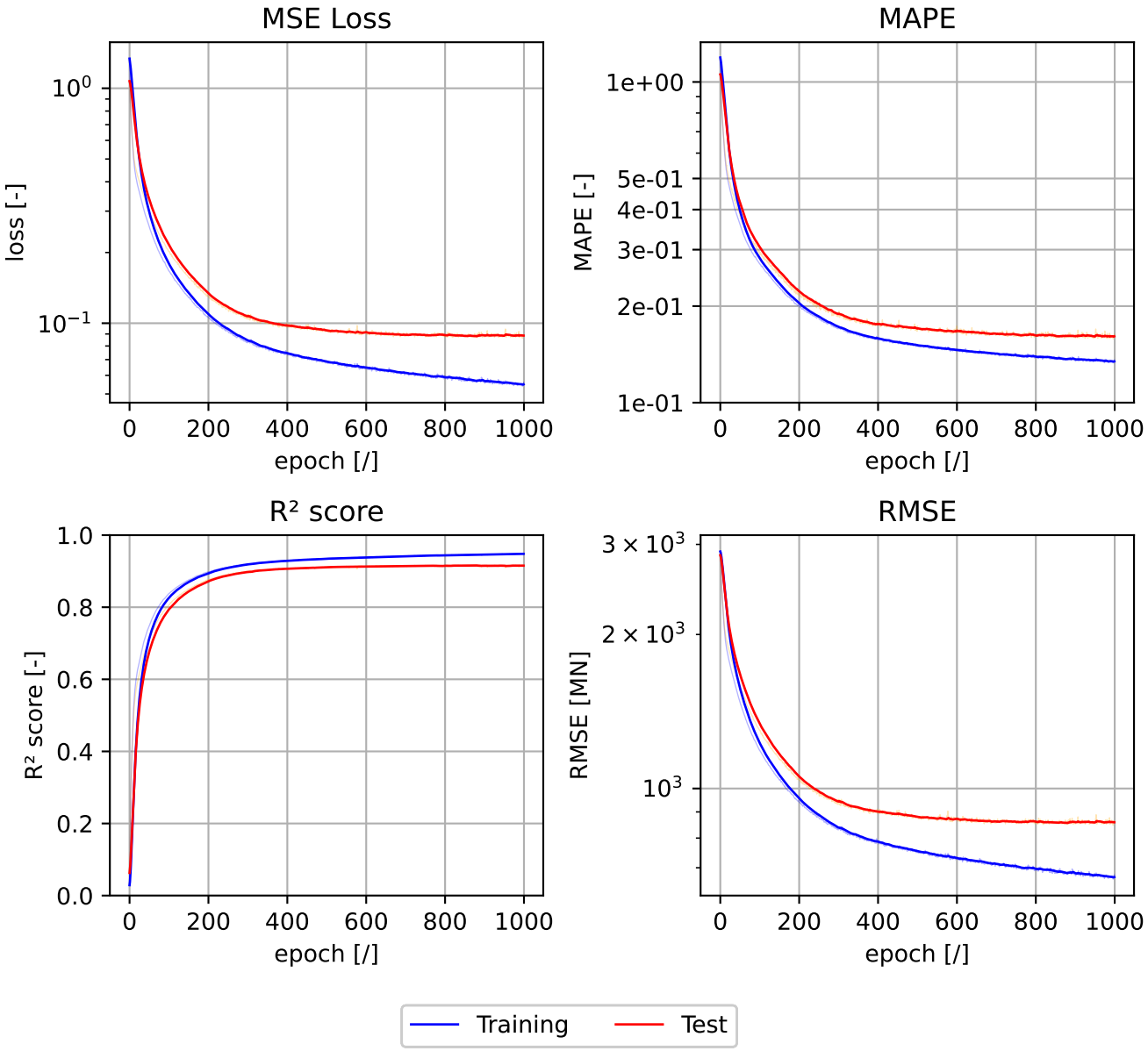


Figure 5.6: *Training and test performance of the physics-informed MLP, using a hybrid loss function and trained on 16,384 examples with a batch size of 512 and tested on a dataset of size 8192.*

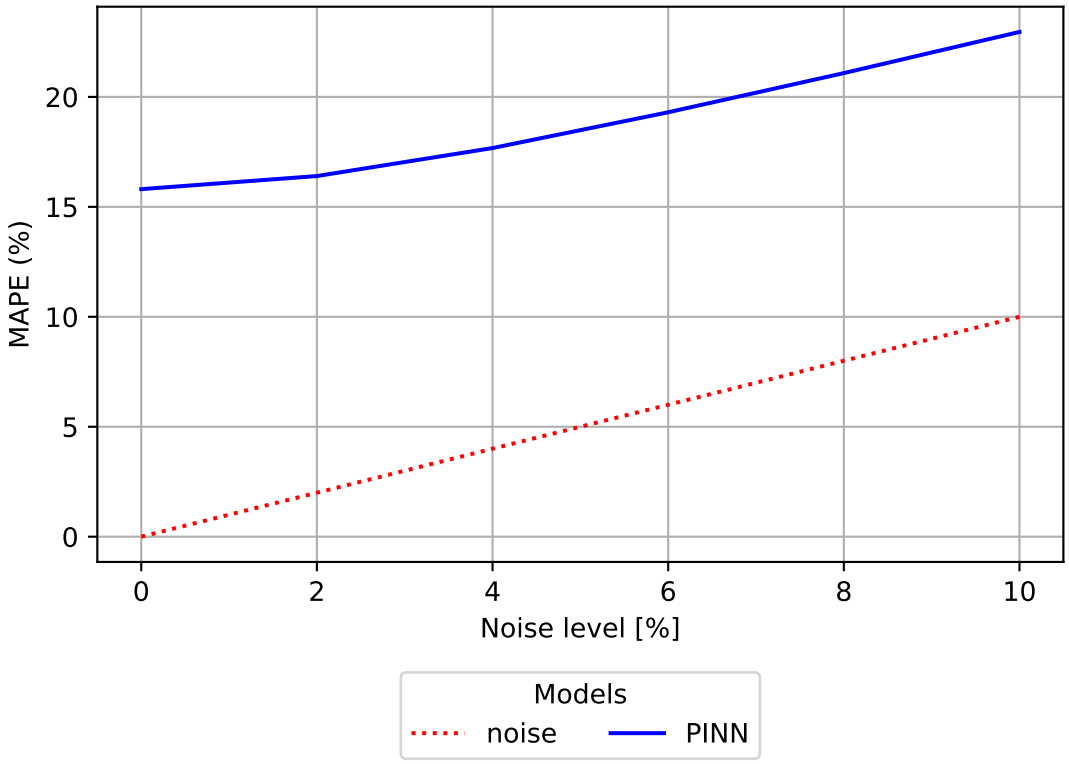


Figure 5.7: *Prediction error (MAPE) of the physics-informed model under increasing levels of input noise.*

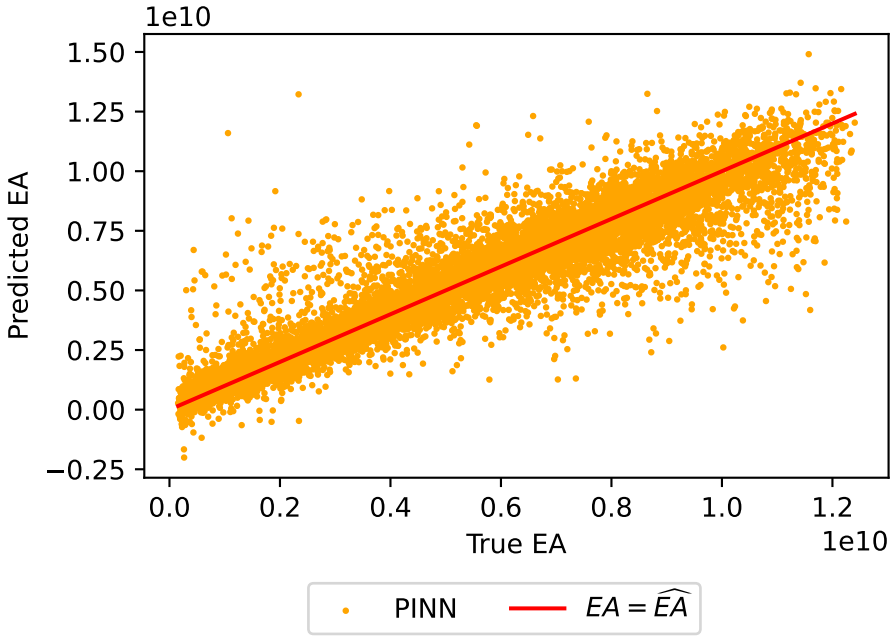


Figure 5.8: *Parity plot of predicted versus true axial rigidities for a test dataset with ±5% measurement noise.*

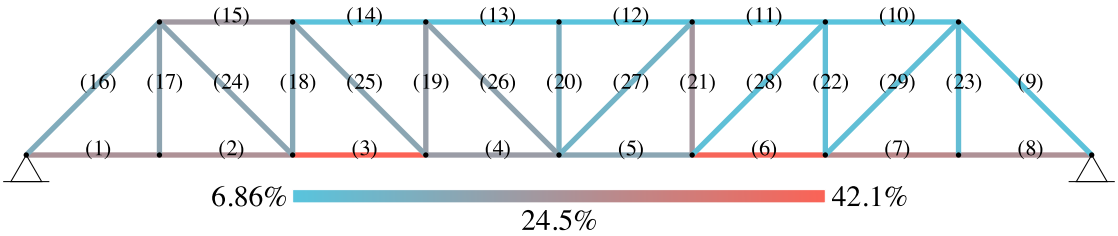


Figure 5.9: *Member-wise distribution of mean absolute percentage error on the test dataset with ±5% noise.*

5.1.3 Models Comparison

The final performance metrics for both models on the noiseless test set are summarised in Table 5.2. The results indicate that the physics-informed neural network (PINN) consistently outperforms the purely data-driven MSE-based model, achieving higher predictive accuracy and superior R^2 scores.

Model	MAPE [%]	R^2	RMSE [MN]
MSE-based MLP	16.24	0.914	863.02
Physics-Informed MLP	**15.73**	**0.918**	**845.83**

Table 5.2: *Comparison of test set performance for the MSE-based MLP and the physics-informed MLP on the member-level EA prediction task.*

A defining characteristic of this problem is the impact of noise from on-site measurements on prediction. Figure 5.10 combines the sensitivity curves from Figures 5.3 and 5.7, showing that the PINN maintains a clear advantage over the MSE-based model across all tested noise levels. Notably, the performance gap widens as noise intensity increases, indicating that the physics constraints act as a stabilising prior that improves robustness to imperfect inputs.

The benefits of the physics-informed approach become even more pronounced when the training dataset size is reduced. As shown in Figure 5.11, for smaller datasets, the PINN delivers substantially better performance than the MSE-based model, in line with the established role of PINNs in compensating for data scarcity through embedded domain knowledge.

Despite these advantages, the absolute performance of both models remains unsatisfactory for practical deployment. A natural strategy for improving accuracy would be to increase the training dataset size beyond the current limit of 16384 examples, taking advantage of the synthetic nature of the data. However, this approach faces two key constraints:

1. **Model performance plateau.** As shown in Figure 5.12, performance appears to converge to a lower bound of around 11% MAPE for this inverse prediction task and structural configuration. Overcoming this plateau may require alternative model architectures or formulations, as discussed in Chapter 6.

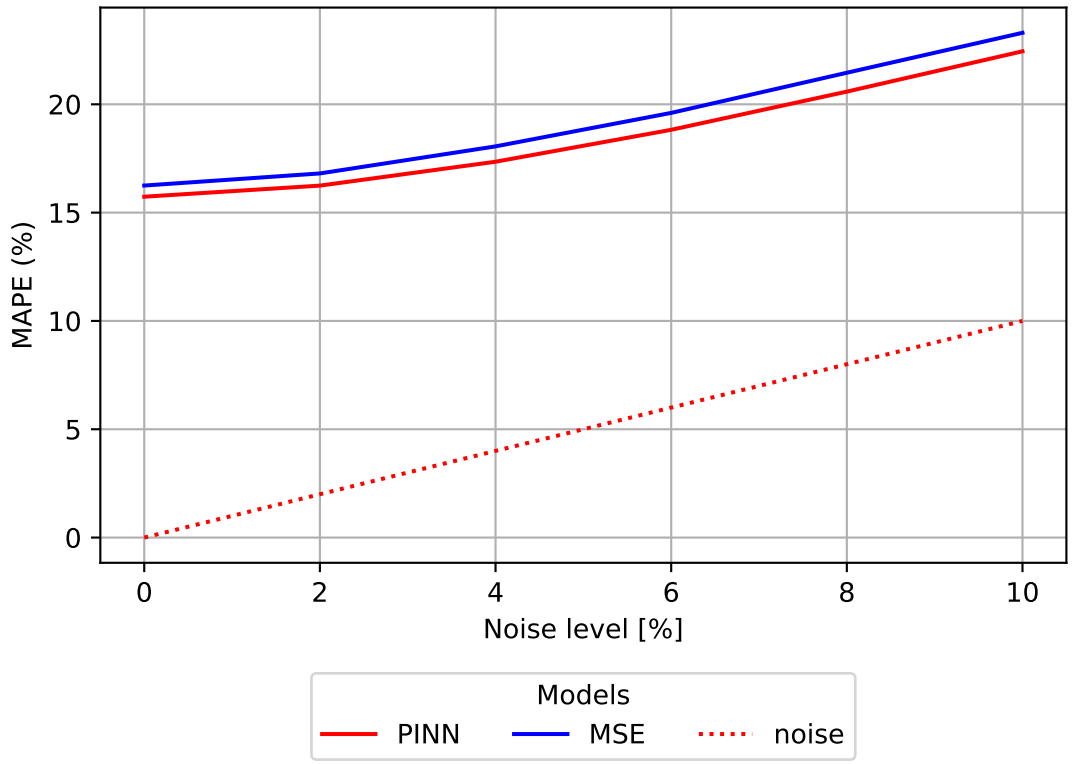


Figure 5.10: *Sensitivity of the MSE-based and physics-informed MLP models to increasing noise levels in displacement and load inputs.*

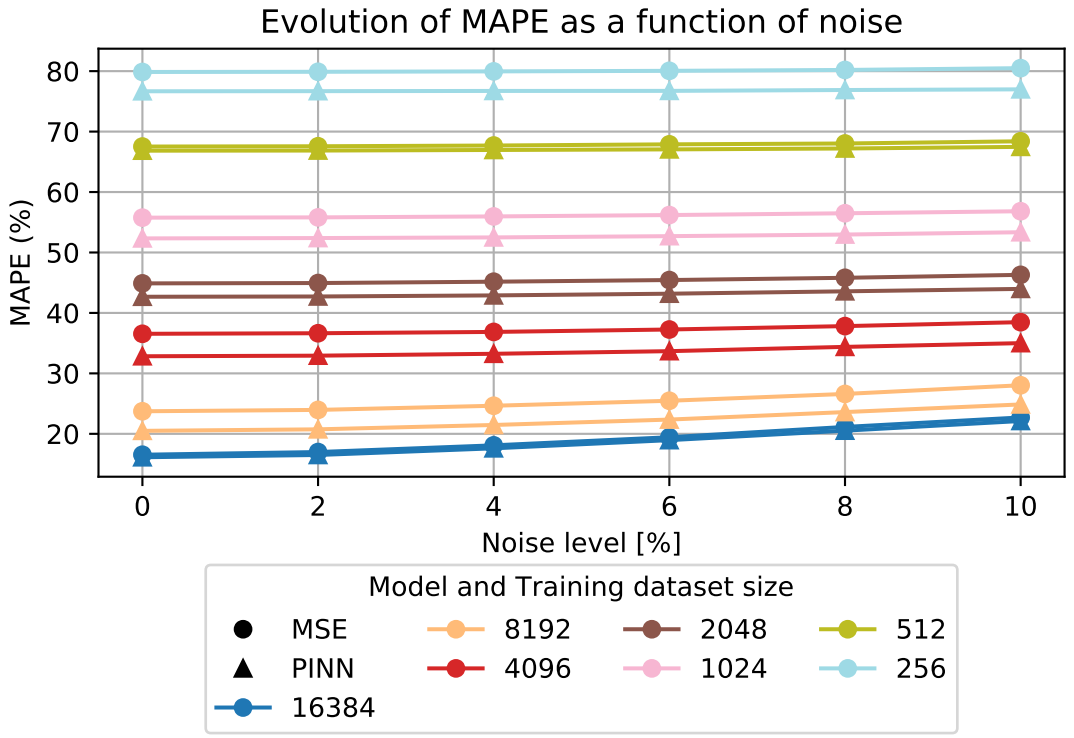


Figure 5.11: *Performance comparison of the two models on a test dataset with 5% multiplicative noise, across different training dataset sizes.*

2. **Computational cost.** While linear truss analysis is among the fastest finite element methods, generating very large datasets for complex structures still entails significant time costs. These computational demands grow even more in the context of nonlinear or dynamic analyses.

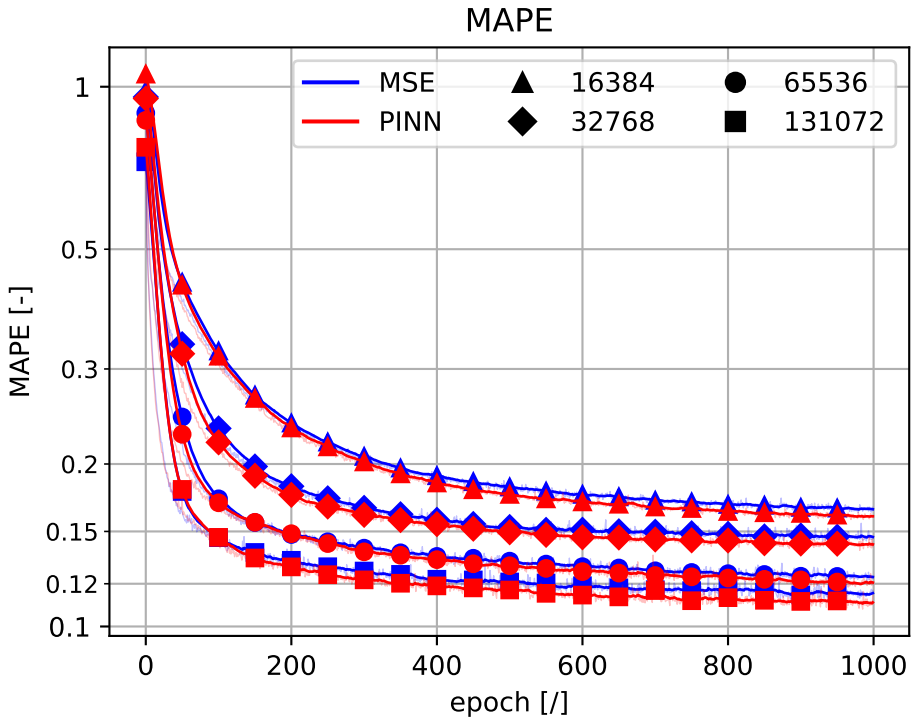


Figure 5.12: *Evolution of MAPE during training for both models across multiple training dataset sizes, highlighting performance convergence around 11% error.*

In light of these limitations, the next section investigates a more pragmatic adaptation of the modelling strategy that is better aligned with the realities of construction practice and on-site measurement constraints.

5.2 Predicting category-based EA

We have observed that the model consistently struggles to predict the axial rigidity of members 3 and 6, and that, globally, the model struggles to provide an accurate prediction for all 29 bars. Fortunately, in practice, the design of a structure follows rules to make the structure more rational; one of those rules is that the members of a structure are often grouped into categories of members sharing the same cross-sectional area A and material, thus sharing the same Young's modulus E. Using this hypothesis, we can solve the accuracy issue encountered in the previous sections.

Firstly, it appears that the properties of aligned members can be hard to differentiate as developed at the end of section 5.1.1. We will assume that all these members have the same axial rigidity. To further relate to actual design principles, we will form other groups to a total of three groups:

- **The horizontal members:** the members from 1 to 16;
- **The diagonal members:** the members from 17 to 23 ;
- **The vertical members** the members from 24 to 29.

The dataset generation under this design hypothesis has already been developed in section 4.4.2.

This reduces the prediction of the model to three values, illustrated in Figure 5.13, where colors identify the category to which each member belongs.

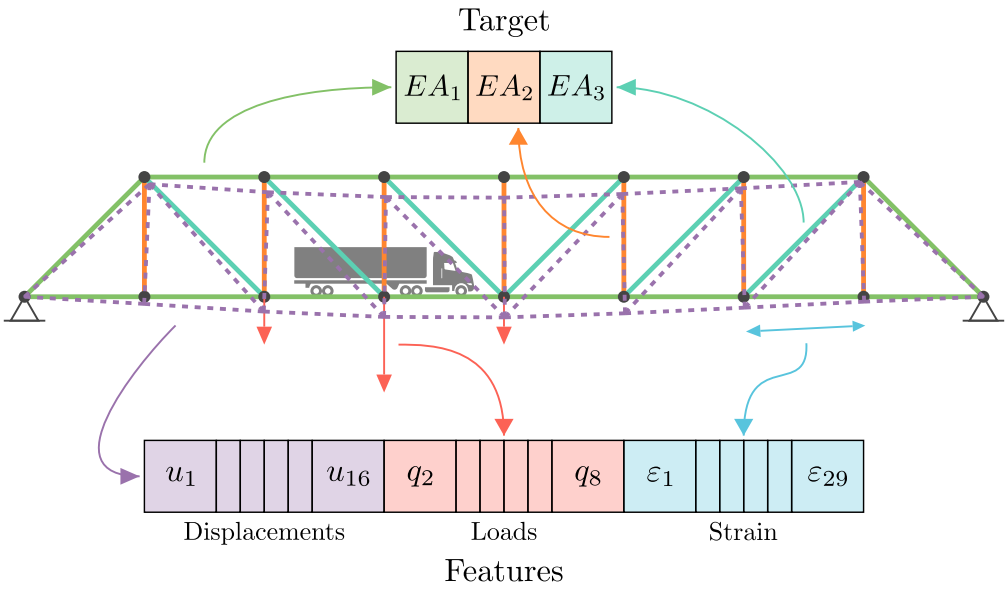


Figure 5.13: *Illustration of the dataset composition and target with the member cateogries hypothesis.*

5.2.1 MSE-Based Model

Following hyperparameter tuning, the MSE-based MLP was configured with four hidden layers of 120 neurons each, using the $\tanh$ activation function. The model was trained on 16,384 examples with a batch size of 512 and evaluated on a separate test set of 8,192 unseen examples.

The training metrics, presented in Figure 5.14, show that the model achieves a MAPE of 3.6%, an RMSE of 284.8 MN, and an R^2 score of 0.99. These results indicate that the model successfully captures the main relationships within the data and is able to provide accurate predictions for the grouped axial rigidity values.

The parity plot in Figure 5.15 compares predicted and true axial rigidity values for a test set subject to $\pm 5\%$ measurement noise. The strong alignment of points along the diagonal demonstrates a high correlation between predicted and true values. Occasional outliers contribute to the relatively high RMSE but remain identifiable and interpretable from an engineering perspective.

Robustness to noise was further assessed by progressively increasing the measurement error level in the inputs. Figure 5.16 shows that the growth in prediction error

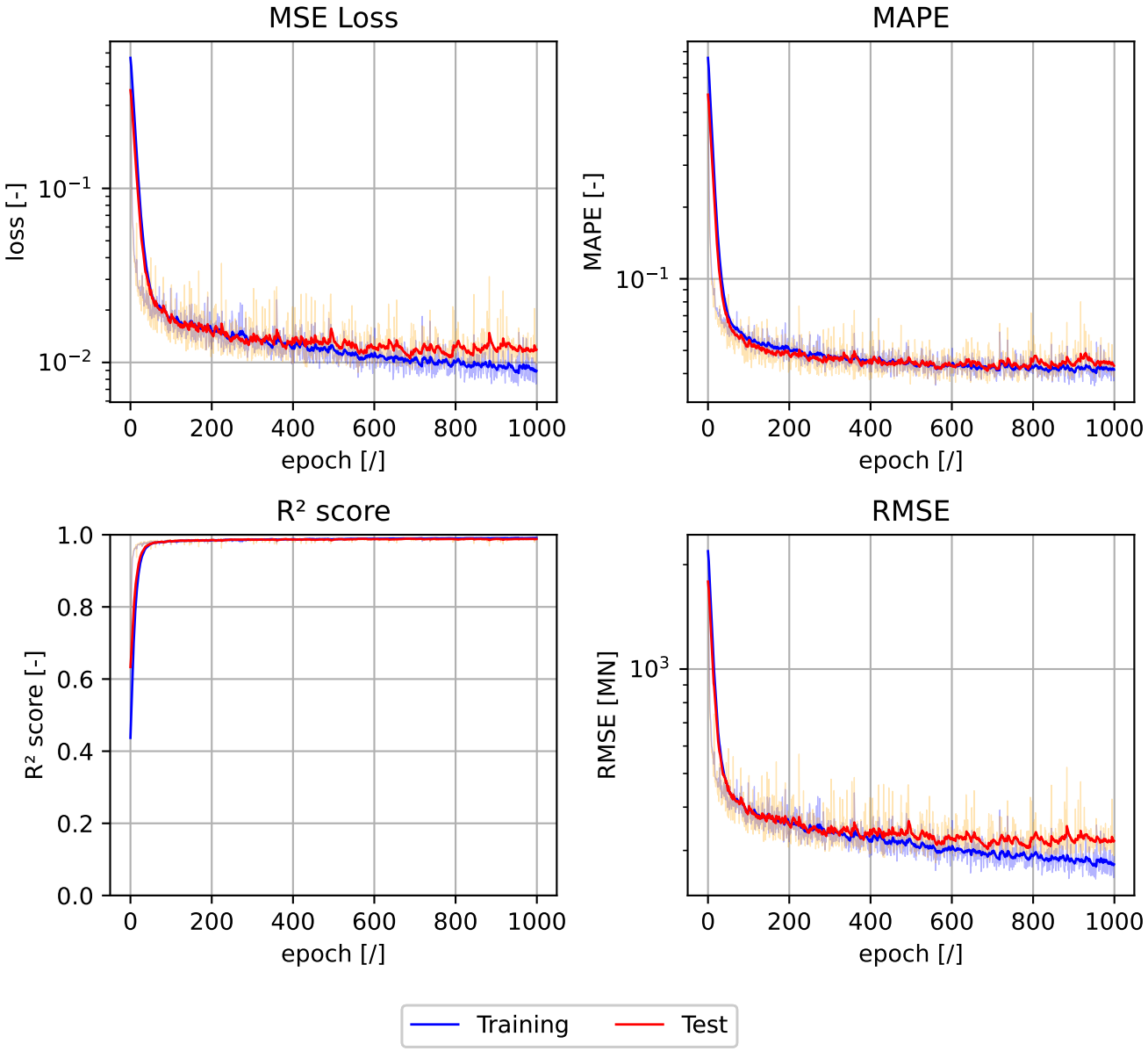


Figure 5.14: *Training and test performance of the category-based MSE model. The model comprises four hidden layers with 120 neurons each, trained on 16,384 samples and evaluated on 8,192 test samples using the* tanh *activation function and a batch size of 512.*

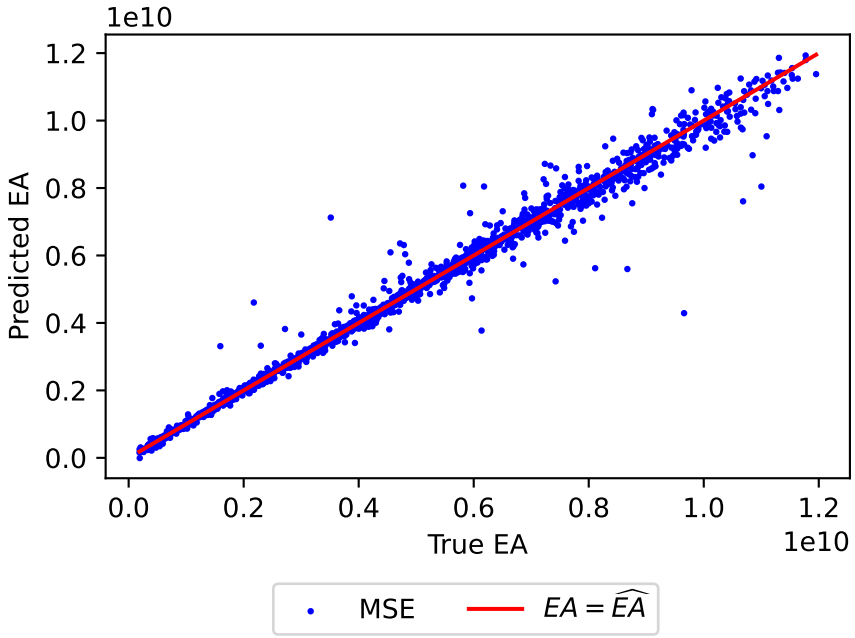


Figure 5.15: *Parity plot of predicted versus true axial rigidity values for the MSE-based category model on a noisy test set (±5% measurement error). The close alignment with the diagonal indicates strong agreement between prediction and ground truth.*

remains consistently below the imposed noise amplitude, highlighting the model's resilience to noise contamination.

The spatial distribution of prediction errors, presented in Figure 5.17, confirms that the previously localised high-error regions in horizontal members have been eliminated under the category-based modelling assumption. The error distribution is now uniform and remains below the 5% noise threshold.

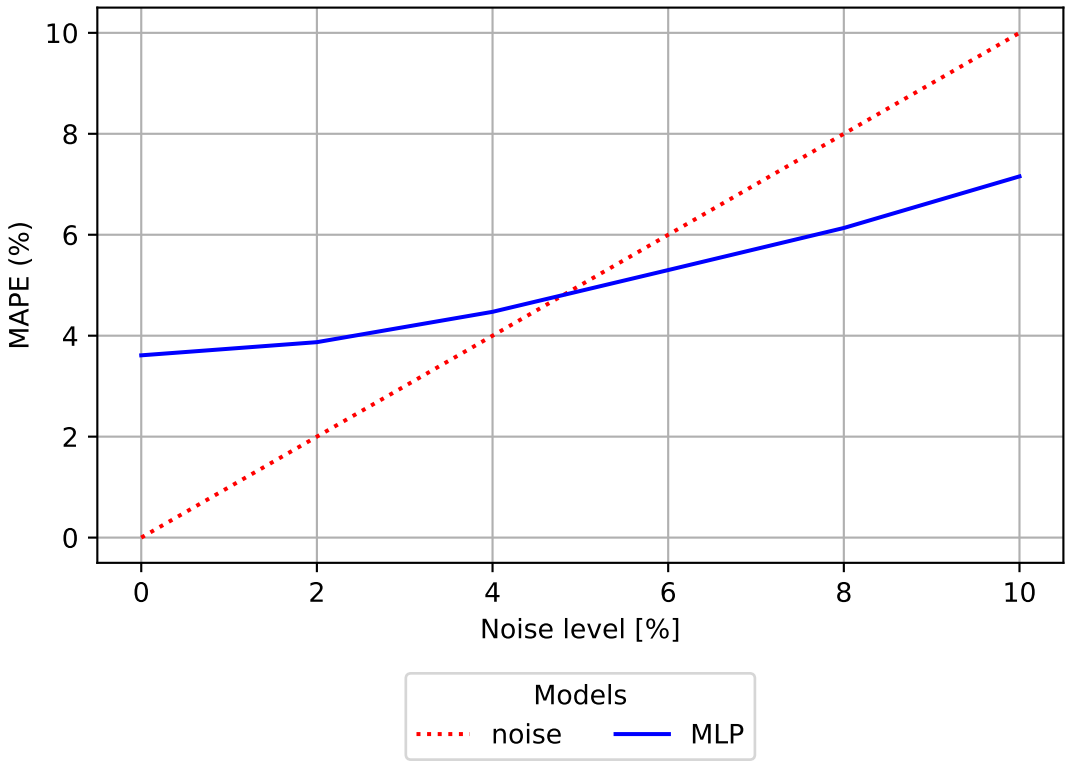


Figure 5.16: *Sensitivity analysis of the MSE-based category model under increasing measurement noise levels. The prediction error grows more slowly than the injected noise amplitude, indicating robustness to measurement uncertainty.*

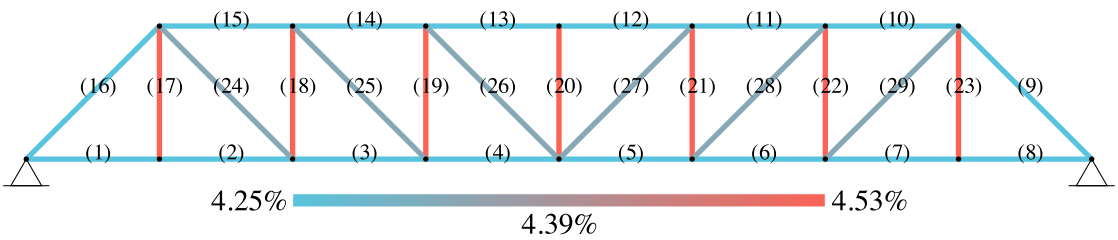


Figure 5.17: *Distribution of MAPE across individual members for a test set with ±5% measurement noise. The category-based modelling assumption eliminates the high-error clusters observed in earlier per-member predictions.*

5.2.2 Physics-Informed Model

The physics-informed category-based model follows a similar MLP architecture but comprises four hidden layers with 100 neurons each, using $\tanh$ as the activation function. The learning rate is set to 10^{-3}. Training is conducted using a hybrid loss function combining data-driven and physics-based terms. The optimal weighting coefficients for the loss components are summarised in Table 5.3.

λ_{MSE}	$\lambda_{\text{stiffness}}$	$\lambda_{\text{equilibrium}}$	λ_{energy}
0.997	0.787	0.209	0.011

Table 5.3: *Optimal weighting coefficients for the hybrid physics-informed category model loss function.*

Although the loss weights differ slightly from those used in the member-level physics-informed model of Section 5.1.2, their orders of magnitude remain comparable. This suggests that the optimal balance between physics and data-driven terms is indeed problem-specific.

The model is trained on 16,384 examples with batches of size 512 and evaluated on a test set of 8,192 unseen examples. Figure 5.18 shows that the best performance is achieved at epoch 761, with a MAPE of 3.61%, an RMSE of 284.79 MN, and $R^2 = 0.991$. These metrics indicate that the model extracts physically consistent relationships while improving prediction accuracy.

The parity plot in Figure 5.19 for a noisy test set ($\pm 5\%$ error) shows strong agreement between predicted and true values, with a reduced dispersion compared to the MSE-based model. This tighter clustering explains the improvement in RMSE.

As with the MSE-based model, robustness testing under increasing noise levels (Figure 5.20) shows that the prediction error increases more slowly than the applied noise, reinforcing the model's stability under measurement uncertainty.

Finally, the error distribution in Figure 5.21 shows a uniform spatial pattern with no pronounced high-error regions, confirming that the category-based hypothesis addresses the localised prediction difficulties seen in per-member modelling.

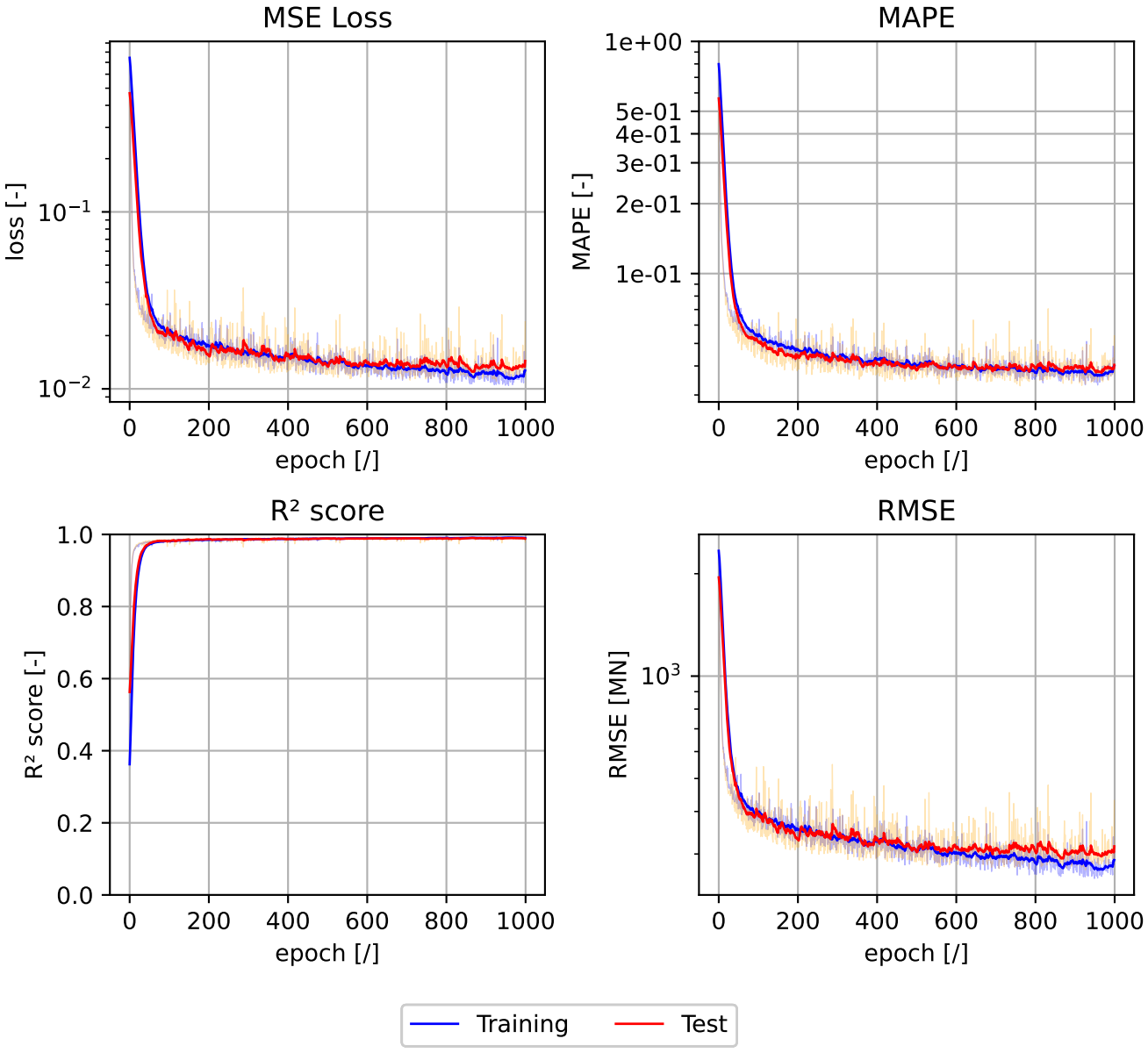


Figure 5.18: *Training and test performance of the physics-informed category model. The model comprises four hidden layers of 100 neurons each, trained on 16,384 samples and evaluated on 8,192 test samples using the* tanh *activation function and a hybrid loss.*

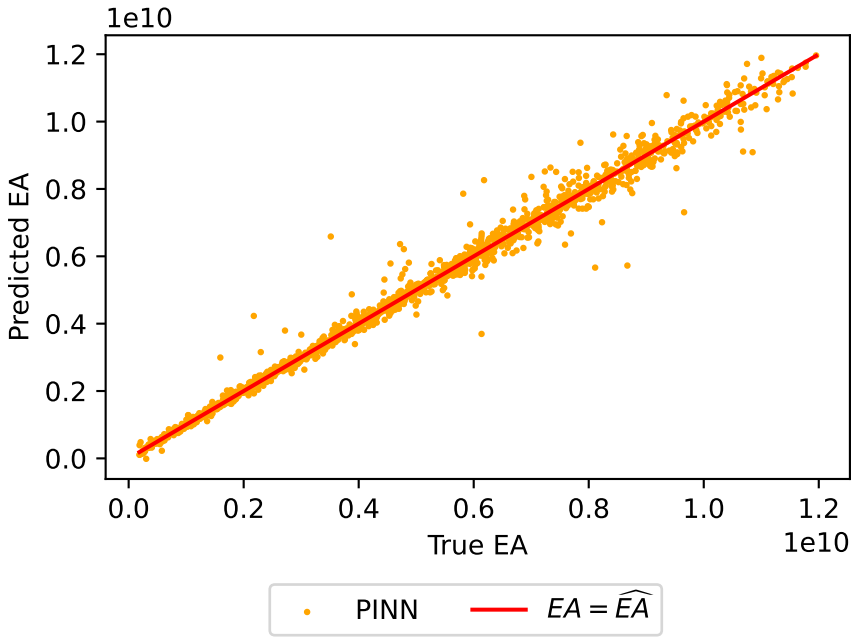


Figure 5.19: *Parity plot of predicted versus true axial rigidity values for the physics-informed category model on a noisy test set ($\pm 5\%$ measurement error). The reduced dispersion around the diagonal reflects the improved RMSE.*

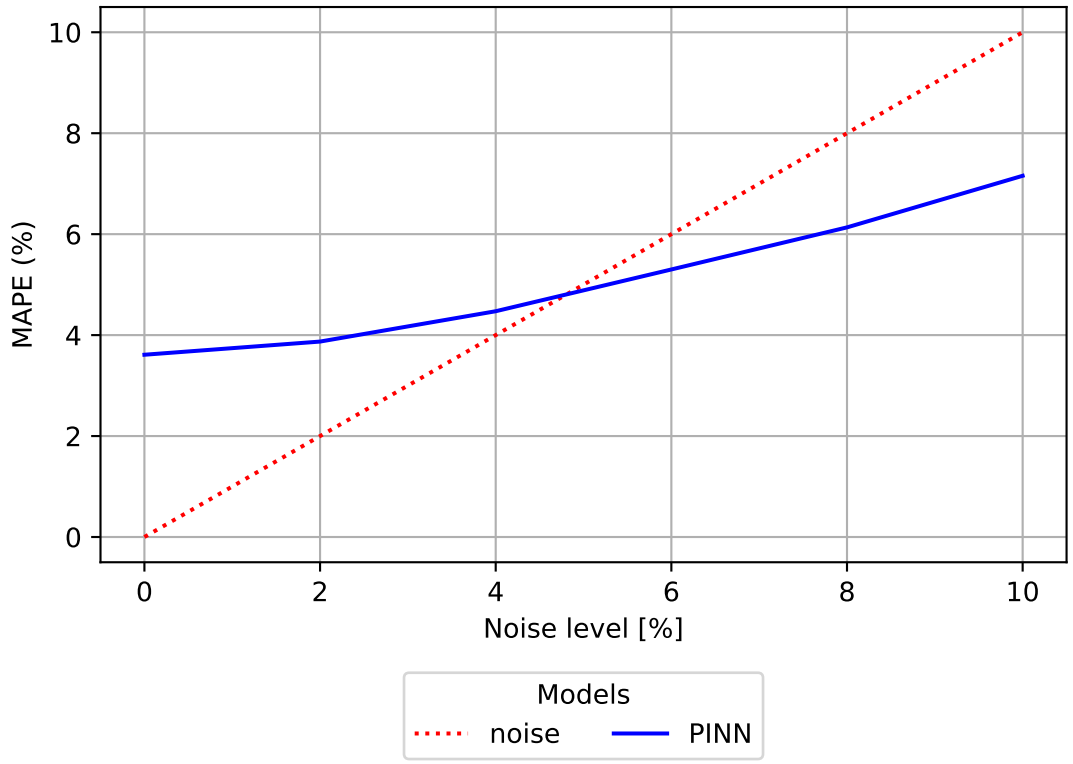


Figure 5.20: *Noise sensitivity analysis for the physics-informed category model. The growth in prediction error remains consistently below the applied noise amplitude.*

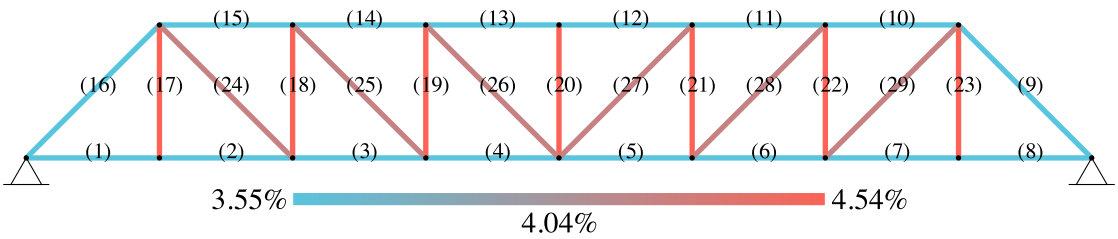


Figure 5.21: *Distribution of MAPE across individual members for a test set with ±5% measurement noise. The category-based physics-informed approach eliminates localised prediction errors and ensures uniform accuracy across members.*

5.2.3 Model Comparison

The final performance metrics for both category-based models on the noiseless test set are presented in Table 5.4. The physics-informed neural network (PINN) consistently outperforms the purely data-driven MSE-based model, achieving lower MAPE, reduced RMSE, and marginally higher R^2 values.

Model	MAPE [%]	R^2	RMSE [MN]
MSE-based MLP	4.20	0.987	332.95
Physics-Informed MLP	**3.61**	**0.991**	**284.79**

Table 5.4: *Performance comparison of the MSE-based and physics-informed MLP models on the category-based EA prediction task using the noiseless test set.*

When compared to the results obtained in Section 5.1.2 for the per-member prediction task, these category-based models deliver markedly superior performance. In the previous formulation, the best models—whether data-driven or physics-informed—produced MAPE values around 16% on noisy data, with error patterns heavily influenced by structural redundancy (notably in members 3 and 6). By contrast, the introduction of rational design constraints through member grouping eliminates such localised weaknesses, enabling both models to achieve single-digit percentage errors while delivering uniform accuracy across the structure.

Noise robustness remains a critical consideration, particularly for field applications where measurement uncertainty is unavoidable. Figure 5.22, which overlays the sensitivity curves from Figures 5.16 and 5.20, shows that the PINN retains a performance advantage over the MSE-based model across all noise levels tested. Importantly, the performance gap widens with increasing noise, indicating that the physics-informed constraints act as a stabilising prior, improving prediction reliability under degraded input conditions.

Beyond robustness, this formulation exhibits strong data efficiency. Figure 5.23 presents the MAPE obtained for noisy test inputs as a function of training dataset size. Both models remain accurate even with reduced training sets, but the PINN shows a pronounced advantage in the small-data regime. This finding implies a practical trade-off: for applications with limited data availability, the additional complexity of implementing a physics-informed loss is justified by the resulting gains in accuracy

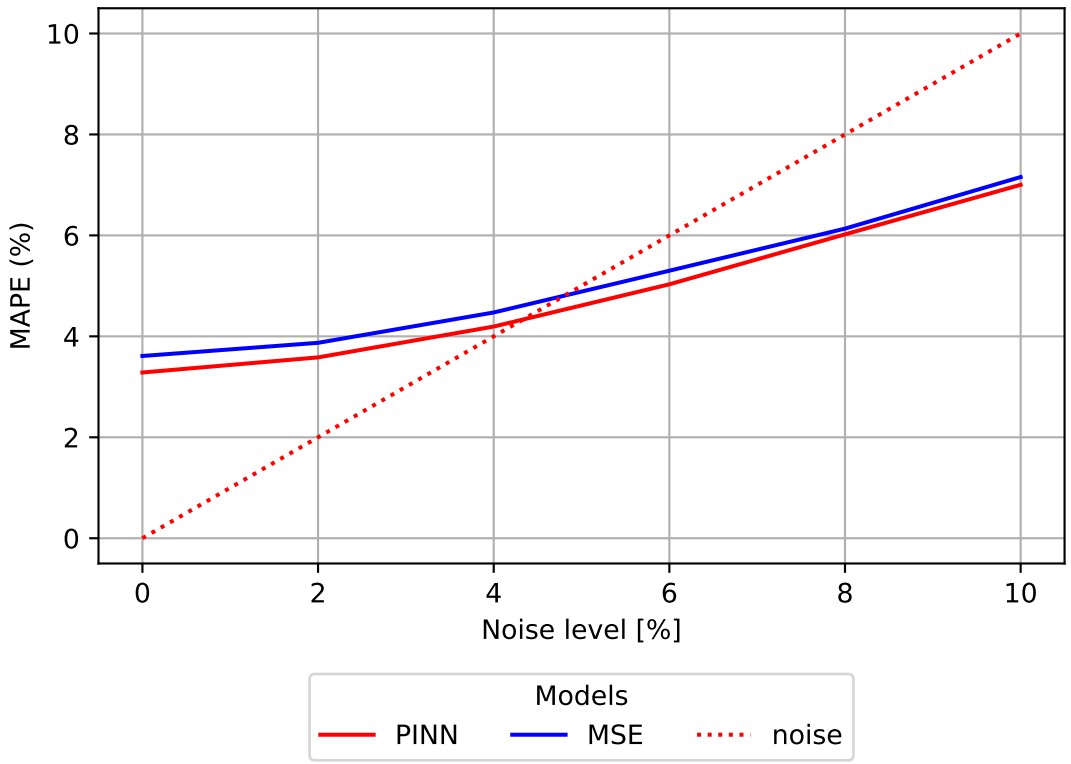


Figure 5.22: *Noise sensitivity comparison between the category-based MSE and physics-informed models. The physics-informed approach maintains lower error rates at all noise levels, with the advantage increasing as noise amplitude grows.*

and robustness.

In summary, the category-based formulation represents a substantial advancement over the per-member prediction task presented earlier in this chapter. By introducing design-informed grouping, the prediction task becomes better conditioned, eliminating structurally induced ambiguities and enabling both models to achieve high accuracy with uniform error distribution. Within this improved setting, the physics-informed approach further enhances performance, particularly under noisy or data-limited conditions. This combination of structural rationality and embedded physical knowledge therefore offers a robust and efficient pathway for practical inverse analysis of truss systems.

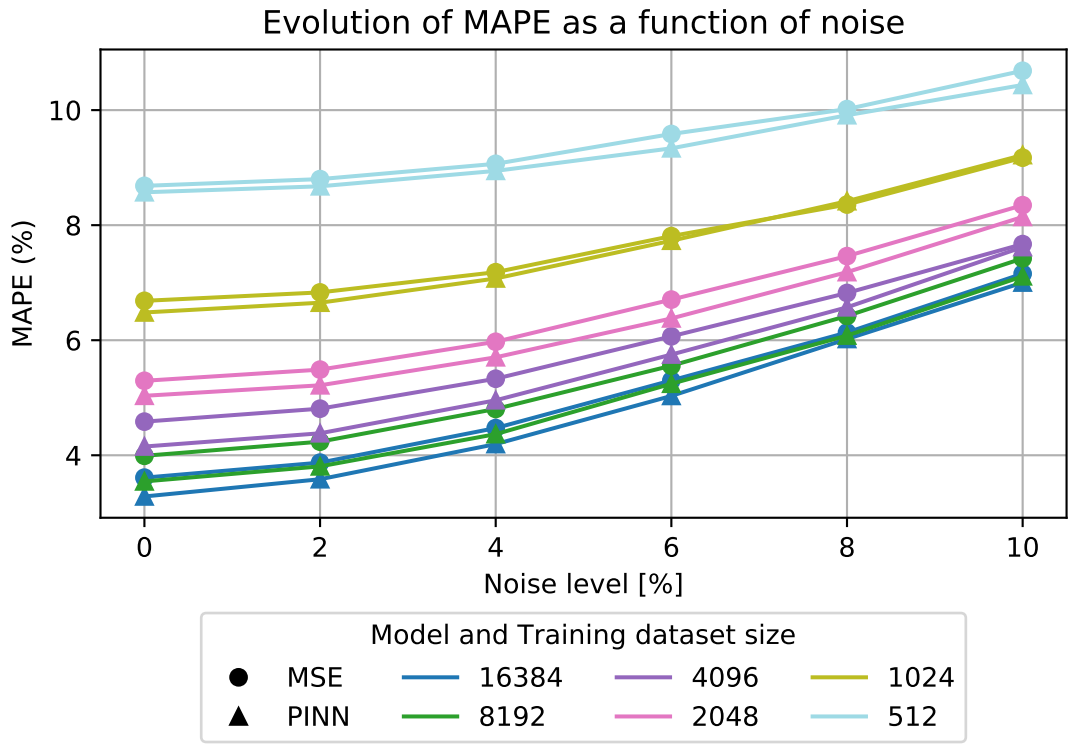


Figure 5.23: *Impact of training dataset size on noisy test set performance* ($\pm 5\%$ *multiplicative noise*). *The physics-informed model consistently outperforms the MSE-based model, with particularly large gains when data is scarce.*

6 Conclusion and Outlook

6.1 Conclusion

This thesis addressed the calibration of finite element models (FEM) for a truss structure under noisy and/or limited measurements. Using a Pratt truss bridge as a case study, we investigated how machine learning models can infer structural parameters—specifically, the members' axial rigidities EA—from observed structural responses. The motivation stems from practice: engineers are often tasked with locating stiffness loss or damage using sparse, noise-contaminated sensor data, where traditional trial-and-error or purely deterministic calibration can be unreliable. This motivates data-driven techniques capable of robustly tackling the underlying inverse problem.

We first generated a comprehensive *synthetic dataset* via high-fidelity simulations of the Pratt truss. Synthetic data afforded full control over noise and parameter variation, enabling us to emulate diverse measurement conditions and structural states. Following recent evidence that networks trained solely on simulation can generalize to real structures [25], the dataset spanned wide ranges of member stiffnesses (including deliberate reductions to mimic damage) together with corresponding displacement responses under load. Controlled noise was injected into the test set to emulate on-site measurement errors. This pipeline provided a rich training and evaluation ground while avoiding the logistical constraints of large-scale field data collection.

On this basis, we implemented and compared *classical machine learning* models,*multilayer perceptron* (MLP), and *physics-informed neural networks* (PINNs) for FEM parameter calibration. Among the classical baselines, comprising analytical solvers and regressors such as k-nearest neighbors and random forests, a standard MLP trained with mean-squared error (MSE) achieved the best accuracy. In the simple truss scenario, the MLP reached mean absolute percentage errors on the order of a few percent on the test set and generalized well: for moderate noise amplitudes, prediction errors remained *below* the injected noise level, indicating that the network learned stable patterns from noisy inputs. These findings confirm the viability of purely data-driven surrogate models for

structural calibration, consistent with recent surrogate-modeling results in structural engineering [25, 24].

We then introduced a PINN that augments the data loss with physics-based penalties derived from the governing equations. Building on prior work that exploits the constitutive relation $\mathbf{Ku} = \mathbf{q}$ for learning [22, 21], we adapted these ideas to the inverse setting and added novel loss terms enforcing nodal equilibrium and global energy conservation. In effect, the network is trained both to fit measured responses and to satisfy the *finite element governing equations* of the truss. The central hypothesis was that embedded physics acts as a regularizer, improving data efficiency and robustness to noise.

Applied to the Pratt truss, the PINN did yield improvements over the purely data-driven MLP. Quantitatively, the PINN slightly reduced test errors (MSE, MAPE) and produced tighter parity plots (predicted vs. true stiffness) with marginally lower RMSE. Crucially, the advantage became *pronounced* under adverse conditions: when training data were curtailed or measurement noise increased. Under such stress tests, the PINN's error growth was more gradual than that of the MSE-trained MLP, reflecting superior data efficiency and noise robustness. This behavior is illustrated in Figure 5.23, where, for training sets of only a few hundred samples, the PINN consistently outperforms the MLP across noise levels.

That said, the magnitude of the gain under well-instrumented, low-noise conditions remained modest: both models were already highly accurate, leaving limited headroom for improvement. Moreover, the physics-informed approach incurs higher implementation and computational costs per epoch—owing to stiffness assembly and equilibrium evaluations inside the training loop. Thus, PINNs offer a clear *trade-off*: they deliver physically consistent, more data-efficient solutions at the expense of added complexity and compute.

In summary, this case study shows that PINNs can modestly but meaningfully outperform conventional data-driven models for FEM calibration, especially when data are scarce or noisy, while promoting physically plausible predictions. When abundant, high-quality data and simplicity are paramount, a calibrated MLP or similar learner may suffice. However, in low-data regimes or whenever physical interpretability is critical, the physics-informed approach proves its value by yielding parameter estimates that better align with true structural behavior under uncertainty for both isostatic and hyperstatic trusses.

6.2 Outlook

The case study and methods developed here are promising, yet several avenues remain to consolidate, broaden, and operationalize their impact. We outline below a set of directions that progress from immediate validation to increasing generality and, finally, to more challenging physics.

1. **Real-world data integration:** The most immediate priority is to validate the pro-

posed approaches on experimental or field data from real structures. All training in this thesis relied on synthetic data; although prior work suggests that models trained with *synthetic structural dataset* can generalize to physical assets [25, 24], this needs to be confirmed for the FEM calibration task. A natural next step is to collect sensor measurements from an actual truss structure and attempt calibration using both the MLP and the PINN as formulated here. Such a study would expose modeling mismatches and test robustness to truly noisy—and potentially biased—measurements. If the PINN maintains its edge in this setting, it would provide strong evidence for its practical adoption.

2. **Transfer learning:** A central question after this work is whether the learned representation transfers across structural systems. The intuition is that the network has captured patterns relevant to structural analysis more generally, not solely to the eight-panel Pratt truss with a 60 m span and 7.5 m height. This can be probed by studying transfer learning: fine-tuning the same model on a new dataset for a different structure and testing for positive transfer, evidenced by faster convergence and/or higher asymptotic accuracy relative to training from scratch.
3. **Toward a foundation model:** A limitation of straightforward transfer is that a standard neural network fixes the number of inputs and outputs and as such are limited to a specific structure typology. Although one can artificially augment input and output dimensions before retraining, future work could explore architectures that naturally accommodate variable topology and size, aiming at a *foundation model* applicable to arbitrary trusses, or structures more broadly. Recent results are encouraging, notably StructGNN, a graph neural network that predicts structural response for 3D building frame systems [30]. GNNs operate on graphs at the node/edge level, leveraging local connectivity. Their application to inverse problems, such as parameter calibration, remains underexplored. An interesting path is to combine this line with transfer learning by leveraging knowledge embedded in StructGNN models.
4. **Automatic defect detection and calibration:** Lee et al. [27] proposed a model to localize damage via dynamic response analysis. However, localization alone does not estimate the updated properties of damaged members. A promising hybrid pipeline would couple their damage-detection architecture with our calibration approach, assigning a design group to each detected damaged member in the spirit of Section 5.2. The detector would flag *where* change occurred, and the calibrator would estimate *how much* the properties have shifted. This could substantially enhance the practical utility of ML–FEM hybrid models for structural assessment.
5. **Adapting the framework to nonlinear modeling:** Our study focused on a linear truss under static loads. In practice, all structures exhibit some degree of nonlinearity, often negligible but sometimes essential. When nonlinear effects matter, the core challenges studied here become more acute: analytical solutions are generally unavailable, and dataset generation becomes far more time-consuming due

to the cost of nonlinear analyses. These constraints make physics-informed learning especially attractive under data scarcity. Extending the present framework to *nonlinear FEM calibration* thus constitutes an exciting avenue for future research.

In conclusion, this thesis takes a step toward more *intelligent*, physics-integrated tools for structural engineering. We have shown that modest gains in accuracy and reliability are achievable by blending data-driven models with fundamental engineering knowledge. These gains come with increased complexity, yet that complexity is increasingly manageable as algorithms and hardware advance. Looking ahead, structural analysis and monitoring will likely combine simulation-based models with learning algorithms, each reinforcing the other's strengths. By pursuing the directions above, researchers and practitioners can continue to expand what machine learning can deliver for civil structures. Ultimately, this line of inquiry supports the vision of safer, smarter infrastructure: bridges and buildings that learn from data without losing sight of the physics that govern their behavior.

This page intentionally left blank.

Bibliography

[1] IPCC. *Summary for Policymakers*. Synthesis Report, Summary for Policymakers. Geneva, Switzerland: Intergovernmental Panel on Climate Change, 2023, pp. 1–34. DOI: 10.59327/IPCC/AR6-9789291691647.001. URL: https://www.ipcc.ch/report/ar6/syr/.

[2] Alexander R. Hartloper et al. *Database of Uniaxial Cyclic and Tensile Coupon Tests for Structural Metallic Materials*. Version 1.0.0. Zenodo, Aug. 2022. DOI: 10.5281/zenodo.6965147. URL: https://doi.org/10.5281/zenodo.6965147.

[3] Brian Everitt. *The Cambridge dictionary of statistics*. Cambridge, UK; New York: Cambridge University Press, 2002. ISBN: 9780521810999. URL: http://www.worldcat.org/search?qt=worldcat_org_all&q=052181099X.

[4] Jan-Willem Romeijn. "Philosophy of Statistics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Fall 2022. Metaphysics Research Lab, Stanford University, 2022.

[5] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: http://dx.doi.org/10.1016/0893-6080(89)90020-8. URL: http://www.sciencedirect.com/science/article/pii/0893608089900208.

[7] Lu Lu Lu Lu et al. "Dying ReLU and Initialization: Theory and Numerical Examples". In: *Communications in Computational Physics* 28.5 (Jan. 2020), pp. 1671–1706. ISSN: 1815-2406. DOI: 10.4208/cicp.oa-2020-0165. URL: http://dx.doi.org/10.4208/cicp.OA-2020-0165.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007. ISBN: 0387310738.

[9] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". In: *Soviet Mathematics Doklady* 27 (1983), pp. 372–376.

[10] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12.1 (1999), pp. 145–151.

[11] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[12] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45 (1989), pp. 503–528. URL: https://api.semanticscholar.org/CorpusID:5681609.

[13] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[14] David Silver et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[15] Rahima Khanam and Muhammad Hussain. *What is YOLOv5: A deep look into the internal features of the popular object detector*. 2024. arXiv: 2407.20892 [cs.CV]. URL: https://arxiv.org/abs/2407.20892.

[16] M. Raissi, P. Perdikaris, and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[17] Adam Paszke et al. "Automatic Differentiation in PyTorch". In: *Proceedings of the NIPS 2017 Workshop on Autodiff*. Long Beach, California, USA, 2017. URL: https://openreview.net/forum?id=BJJsrmfCZ.

[18] Ameya D Jagtap, Elham Kharazmi, and George Em Karniadakis. "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems". In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028. DOI: 10.1016/j.cma.2020.113028.

[19] Georgios Kissas et al. "Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 358 (2020), p. 112623. DOI: 10.1016/j.cma.2019.112623.

[20] Pratik Rathore et al. *Challenges in Training PINNs: A Loss Landscape Perspective*. 2024. arXiv: 2402.01868 [cs.LG]. URL: https://arxiv.org/abs/2402.01868.

[21] Thang Le-Duc, H. Nguyen-Xuan, and Jaehong Lee. "A finite-element-informed neural network for parametric simulation in structural mechanics". In: *Finite Elements in Analysis and Design* 217 (2023), p. 103904. ISSN: 0168-874X. DOI: https://doi.org/10.1016/j.finel.2022.103904. URL: https://www.sciencedirect.com/science/article/pii/S0168874X22001779.

[22] Rishith Ellath Meethal et al. *Finite Element Method-enhanced Neural Network for Forward and Inverse Problems*. 2022. arXiv: 2205.08321 [cs.CE]. URL: https://arxiv.org/abs/2205.08321.

[23] Ioannis Seventekidis, Andreas Pappas, and Dimitrios Theocharis. "A CNN-based framework for damage detection in truss structures trained on finite-element simulations". In: *Mechanical Systems and Signal Processing* 157 (2021), p. 107735. DOI: 10.1016/j.ymssp.2021.107735.

[24] Wonjong Lee, Sehoon Kim, and Hyunsoo Park. "Sim-to-real damage localization in steel frames using physics-updated digital twins and deep learning". In: *Scientific Reports* 13 (2023), p. 18694. DOI: 10.1038/s41598-023-46141-9.

[25] Sithija Jayawickrema, Eranga Perera, and Farhan Fernando. "Synthetic data–driven structural health monitoring of reinforced concrete beams". In: *Metrology* 5.3 (2025), p. 40. DOI: 10.3390/metrology5030040.

[26] Shanglian Zhou, Carlos Canchila, and Wei Song. "Deep learning-based crack segmentation for civil infrastructure: data types, architectures, and benchmarked performance". In: *Automation in Construction* 146 (2023), p. 104678. DOI: 10.1016/j.autcon.2022.104678.

[27] Yunwoo Lee et al. "Structural damage detection using deep learning and FE model updating techniques". In: *Scientific Reports* 13.18694 (2023). DOI: 10.1038/s41598-023-46141-9.

[28] Liang Liang et al. "A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis". In: *Journal of The Royal Society Interface* 15.138 (2018), p. 20170844. DOI: 10.1098/rsif.2017.0844.

[29] Shujian Zhu, Xudong Zhou, et al. "Non-linear buckling load prediction of imperfect reticulated shells using machine learning techniques". In: *Thin-Walled Structures* 163 (2021), p. 107769. DOI: 10.1016/j.tws.2021.107769.

[30] Yuan-Tung Chou et al. "StructGNN: An efficient graph neural network framework for static structural analysis". In: *Computers & Structures* 299 (2024), p. 107385. DOI: 10.1016/j.compstruc.2024.107385.

[31] Hernan J. Logarzo, German Capuano, and Julian J. Rimoli. "Smart constitutive laws: Inelastic homogenization through machine learning". In: *Computer Meth-*

ods in Applied Mechanics and Engineering 373 (2021), p. 113482. DOI: 10.1016/j.cma.2020.113482.

[32] *Truss metal bridges — sisgeo.com*. https://sisgeo.com/geotechnical-monitoring-applications/bridge-monitoring/truss-metal-bridges/. [Accessed 15-08-2025].

[33] J. Spencer W. *Fundamental Structural Analysis*. New York, NY: Springer Science & Business Media, 1988. ISBN: 978-1-4757-2006-8. DOI: 10.1007/978-1-4757-2006-8.

[34] João Saraiva Esteves Pacheco De Almeida. *Stabilité des constructions*. Course syllabus, Academic year 2025–2026. LGCIV1023; 5 ECTS; taught in French at Louvain-la-Neuve. 2025. URL: https://uclouvain.be/cours-2025-lgciv1023.

[35] NBN. *Les Eurocodes : normes de construction uniformes en Europe*. https://www.nbn.be/fr/themes/eurocodes. Accessed: 2025-04-27. 2025.

[36] Gang Liu et al. "Displacement Measurement Based on UAV Images Using SURF-Enhanced Camera Calibration Algorithm". In: *Remote Sensing* 14.23 (2022). ISSN: 2072-4292. URL: https://www.mdpi.com/2072-4292/14/23/6008.

[37] Sainab Feroz and Saleh Abu Dabous. "UAV-Based Remote Sensing Applications for Bridge Condition Assessment". In: *Remote Sensing* 13.9 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13091809. URL: https://www.mdpi.com/2072-4292/13/9/1809.

[38] Service d'Études sur les Transports, les Routes et leurs Aménagements (Sétra). *Guide technique – Épreuves de chargement des ponts*. Tech. rep. Volume 1: Catégorisation, méthodologie, instrumentation, exploitation. Seyssinet-Pariset, France: Sétra, Ministère de l'Équipement, 2004.

[39] R. A. Fisher. "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.7 (1936), pp. 179–188.

[40] *Optuna - A hyperparameter optimization framework — optuna.org*. https://optuna.org/. [Accessed 26-06-2025].

[41] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML]. URL: https://arxiv.org/abs/1807.02811.

[42] Simon Vandenhende et al. "Revisiting Multi-Task Learning in the Deep Learning Era". In: *CoRR* abs/2004.13379 (2020). arXiv: 2004.13379. URL: https://arxiv.org/abs/2004.13379.

[43] Common Crawl. *Common Crawl: Open Repository of Web Crawl Data*. https://commoncrawl.org/. Accessed: 2025-03-19.

[44] Data Commons. *Data Commons: Integrating Open Datasets into a Unified Knowledge Graph*. https://datacommons.org/. Accessed: 2025-03-19.

[45] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848. URL: https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf.

[46] Sattar Dorafshan, Robert J. Thomas, and Marc Maguire. "SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks". In: *Data in Brief* 21 (2018), pp. 1664–1668. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2018.11.015. URL: https://www.sciencedirect.com/science/article/pii/S2352340918314082.

[47] Martin Mundt et al. *Meta-learning Convolutional Neural Architectures for Multi-target Concrete Defect Classification with the COncrete DEfect BRidge IMage Dataset*. 2019. arXiv: 1904.08486 [cs.CV]. URL: https://arxiv.org/abs/1904.08486.

[48] CCNY Robotics Lab. *Concrete Crack and Spalling Detection using Deep Neural Network*. https://github.com/CCNYRoboticsLab/concreteIn_inpection_VGGF. Accessed: 2025-03-19.

[49] Yahui Liu et al. "DeepCrack: A Deep Hierarchical Feature Learning Architecture for Crack Segmentation". In: *Neurocomputing* 338 (2019), pp. 139–153. DOI: 10.1016/j.neucom.2019.01.036.

[50] Rama. *A testing machine performing a bending test of a concrete beam*. Accessed: 2025-03-19. 2006. URL: https://commons.wikimedia.org/wiki/File:Trois-points-p1040189.jpg.

[51] D. Boerema. "A framework for synthetic dataset generation in pixel-wise crack segmentation". In: *Proceedings of the 1st Interdisciplinary Workshop on Computer Vision Technologies for the Built Environment*. Groningen, Netherlands, 2024. URL: https://research.hanze.nl/en/activities/a-framework-for-synthetic-dataset-generation-in-pixel-wise-crack-.

[52] *Eurocode 0: Basis of Structural Design*. EN 1990. Brussels, Belgium: European Committee for Standardization (CEN), 2002.

[53] *Eurocode 1: Actions on Structures*. EN 1991. Brussels, Belgium: European Committee for Standardization (CEN), 2002.

[54] Soumik Das. *CURSE OF DIMENSIONALITY — soumiksanku08*. https://medium.com/@soumiksanku08/curse-of-dimensionality-293d0d16fe2a. [Accessed 15-08-2025].

[55] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a

Computer Code". In: *Technometrics* 21.2 (1979), pp. 239–245. ISSN: 00401706. URL: http://www.jstor.org/stable/1268522 (visited on 2025-05-08).

[56] I.A. Antonov and V.M. Saleev. "An economic method of computing LPT-sequences". In: *USSR Computational Mathematics and Mathematical Physics* 19.1 (1979), pp. 252–256. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(79)90085-5. URL: https://www.sciencedirect.com/science/article/pii/0041555379900855.

[57] *Open System for Earthquake Engineering Simulation - Home Page*. https://opensees.berkeley.edu/. [Accessed 08-05-2025].

[58] *GitHub - zhuminjie/OpenSeesPy: OpenSeesPy versions, doc, and pip — github.com*. https://github.com/zhuminjie/OpenSeesPy. [Accessed 08-05-2025].

[59] *The HDF5 Library & File Format - The HDF Group - ensuring long-term access and usability of HDF data and supporting users of HDF technologies*. https://www.hdfgroup.org/solutions/hdf5/. [Accessed 01-08-2025].

This page intentionally left blank.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**