

Dissertation Thesis



Czech
Technical
University
in Prague

F2

Faculty of Mechanical Engineering
Department of Instrumentation and Control Engineering

Hybrid Modeling of Mechanical Digital Twin by Finite Element Method and Graph Neural Networks

Marek Ciklamini

Supervisor: prof. Ing. Tomáš Vyhlídal, Ph.D.
Supervisor–specialist: Ing. Matouš Cejnek, Ph.D.
Field of study: Systems and Control Engineering
March 2024

Acknowledgements

Děkuji především mé podporující rodině a taktéž ČVUT, že mi je tak dobrou *alma mater*.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu. V Praze, 27. March 2024

Abstract

This dissertation proposes a hybrid modelling approach for the design of a digital twin of a mechanical structure. The concept integrates the connection of the finite element method and the graph neural network. The advantages of the physical-based method of accurately simulating complex physical and structural behaviour are extended by the possibility of effective data acquisition and, thus, expand the compact understanding of the given mechanical structure. The work aims to answer whether regressors based on graph neural networks can effectively build a digital twin. Studies supporting this methodology are presented in this work to suggest a perspective on how challenges relate to establishing digital twins. Designed experiments on the training of a regressor and its validation are addressed to ensure the accuracy and generality of the hybrid model as a whole mechano-digital framework.

Keywords: Digital Twin, Hybrid Modelling, Finite Element Method, Graph Neural Networks, Supervised Learning

Supervisor: prof. Ing. Tomáš Vyhlídal, Ph.D.
Ústav přístrojové a řídicí techniky
Technická 1902/4
Praha

Abstrakt

Tato disertační práce navrhuje hybridní modelovací přístup pro návrh digitálního dvojčete mechanické struktury. Koncepce integruje propojení metody konečných prvků s grafovou neuronovou sítí. Výhody první metody přesně simulovat složité fyzikální strukturální chování je rozšířeno o možnost efektivní datové akvizice, která rozšiřuje kompaktní porozumění o dané mechanické struktuře. Hlavním cílem práce je odpovědět na otázku, zdali regresor grafových neuronových sítí může být efektivním nástrojem pro stavbu digitálního dvojčete. Studie podporující tuto metodologii jsou představeny v této práci a slouží tak k navrhnutí pohledu, jak mohou být řešeny výzvy související s vytvořením digitálního dvojčete. Dále je pomocí navržených experimentů ověřena možnost trénování a validace regresoru tak, aby byla zajištěna přesnost a obecnost hybridního modelu jakožto celku mechanicko-digitální struktury.

Klíčová slova: Digitální dvojče, Hybridní modelování, Metoda konečných prvků, Grafové neuronové Sítě, Učení s učitelem

Překlad názvu: Digitální dvojče mechanického Systému - Hybridní modelování digitálního dvojčete metodou konečných prvků a grafových neuronových sítí — Hybridní modelování Digitálního Dvojčete Mechanické Struktury Metodou Konečných Prvků a Grafových Neuronových Sítí

Contents

List of Notations	1	1.5.3 GNNs for the use as DD	23
Introduction	3	1.6 Distinct Possible Applications Based on GNN	26
Part I			
Hybrid Modelling of Digital Twins			
1 Status Quo of Digital Twin Concept	7	1.7 Conclusion on The State-of-The-Art	30
1.1 Representative DT applications .	10	1.8 Problem statement	31
1.2 Challenges of DT	12	2 Hypothesis and Goals of Dissertation Thesis	33
1.3 Hybrid Modelling	13	2.1 Formulation of a hypothesis	33
1.3.1 Possible Use Cases	14	2.2 Thesis Objectives	34
1.4 Physical-Based Modelling	15	3 Methods for Hybrid Modelling of Digital Twin	35
1.4.1 Overview of Solving PDEs for a PM	16	3.1 Physical-based Modelling via Finite Element Technique	36
1.4.2 Fields Excerpts of PM	17	3.1.1 Poission Problem	37
1.5 Data-Driven Modelling	19	3.1.2 Variational Formulation	38
1.5.1 Fundamenatal Methods to Create Regressor for DD	20	3.2 Graph Theory and Poisson Problem	40
1.5.2 Balancing on DD and PM Modelling	22	3.3 Graph Extraction from Physical Based Model	42
		3.3.1 Graph Extraction from Nodal Structure of an Element	43

5.1.3 Scaling of a Dataset	76	6.2 Ground Truth Evaluation of Digital Twins	91
5.1.4 Data Splitting	76	6.2.1 GT Evaluation: Beam 2D . . .	92
5.1.5 Loss Functions and Optimizer	76	6.2.2 GT Evaluation: Beam 3D . .	94
5.2 Graph Reduction	77	6.2.3 GT Evaluation: Fibonacci's spring	96
5.3 GF dataset: Mesh-based Graph Dataset	77	6.2.4 GT Evaluation: Plane	98
5.4 Use Case: Beam 2D	79	6.3 Model Diagnostic of DTs	100
5.4.1 Geometry Material and Mesh	80	6.3.1 MD Evaluation: Beam 2D .	101
5.4.2 External Load and Boundary Conditions	80	6.3.2 MD Evaluation: Beam 3D .	103
5.4.3 Data acquired for Data-Driven Models	80	6.3.3 MD Evaluation: Fibonacci's spring	105
5.5 Use Case: Beam 3D	81	6.3.4 MD Evaluation: Plane	107
5.6 use case: Fibonacci's spring	82	6.4 Regressor Training Evaluation .	109
5.7 Use case: Half Airplane - Symmetrically Cut	83	6.4.1 Training Observation: Beam 2D	110
5.8 Summary of GF Dataset	84	6.4.2 Training Observation: Beam 3D	112
6 Results of Baseline for DTs	87	6.4.3 Training Observation: Fibonacci's spring	114
6.1 Accuracy and Time training of Regressor	89	6.4.4 Training Observation: Plane	116

6.5 Graph Reduction Experiment .	118
6.6 Results Discussion	119
7 Conclusion	123
7.1 Outlook of possible applications	125

Appendices

A Index	129
B List of Author Publications	131
Thesis Related	131
Other Publications	132
C Bibliography	133
D Graph Reduction: Algorithm and Tables	139

Figures

1.1 Simulation waves [38]	8	3.1 Illustrative example of physical model Ω based on tetrahedral elements building specific FEM. . .	36
1.2 Living heart [41]	10	3.2 Illustrative example of directed graph \mathcal{G} extracted from structure Ω	36
1.3 Procedure planning [42]	10	3.3 Accompanying example of two-dimensional beam for Poisson problem description	37
1.4 Intelligent Engine [43]	11	3.4 Stress distribution on accompanying descriptive example of two-dimensional beam	40
1.5 Venn diagram to depict union of Physical- based and Data-Driven modelling techniques	13	3.5 Tetra element represents as as undirected graph \mathcal{G} by adjacency matrix A	47
1.6 Visualization data of Ice Cube experiment for detection of neutrino particles [58].	27	3.6 Tetra element represented as directed graph A	47
1.7 From CT scan to the virtual representation of biological system [59].	27	3.7 Process of DT creation from FEM to Graph representation with further graph reduction process; from left: mechanical model, FEM, fully connected graph, reduced graph (shortest path), reduced graph (closed path)	49
1.8 Graph Neural Lasso for Dynamic Network Regression [49].	28	3.8 Data workflow of DT	57
1.9 Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network [48].	29	3.9 Data workflow from Physical model distilling information	58
1.10 Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network [5].	30	5.1 framework MLR: Multi-linear input/output regression	74

5.2 framework FN: Feed-forward neural network with two hidden layers	74	6.1 GFdataset: Boxplot of RMAXSE to depict accuracy of a regressor for DT implementation	89
5.3 framework GCN: two graph convolutional layers	75	6.2 GFdataset: Boxplot of overall training time required to establish regressor for specific DT	90
5.4 Framework SAGE: two graph SAGE layers	75	6.3 GT visualisation for the MLR framework of Beam2D DT for specific data sample	92
5.5 GFdataset: Graphs of structures extracted from FE models \mathcal{D} [69] .	77	6.4 GT visualisation for the FN framework of Beam2D DT for specific data sample	92
5.6 GF dataset: Visualized each FE model from \mathcal{D} in their final converged state	78	6.5 GT visualisation for the GCN framework of Beam2D DT for specific data sample	93
5.7 The workflow presented on Beam2D example: upper left: FE discretisation with the highlight of boundary condition and load applied; bottom left: Maximal Principal distribution at last frame; right: drawn bi-directional heterogeneous graph created based on FE model mesh.	79	6.6 GT visualisation for the SAGE framework of Beam2D DT for specific data sample	93
5.8 Beam3D: On the left is FE discretization, the right demonstrates the distribution of Maximal Principal stress at the final frame of simulation	81	6.7 GT visualisation for the MLR framework of Beam3D DT for specific data sample	94
5.9 Fibonacci Spring: FE discretisation and Maximal Principal distribution at last frame	82	6.8 GT visualisation for the FN framework of Beam3D DT for specific data sample	94
5.10 Plane: FE discretization with boundary conditions and Maximal Principal distribution at last frame	83	6.9 GT visualisation for the GCN framework of Beam3D DT for specific data sample	95
		6.10 GT visualisation for the SAGE framework of Beam3D DT for specific data sample	95

6.11 GT visualisation for the MLR framework of Fibonacci's spring DT for specific data sample	96	6.22 MD for DT regressor based on SAGE applied on Beam2D	102
6.12 GT visualisation for the FN framework of Fibonacci's spring DT for specific data sample	96	6.23 MD for DT regressor based on MLR applied on Beam3D	103
6.13 GT visualisation for the GCN framework of Fibonacci's spring DT for specific data sample	97	6.24 MD for DT regressor based on FN applied on Beam3D	103
6.14 GT visualisation for the SAGE framework of Fibonacci's spring DT for specific data sample	97	6.25 MD for DT regressor based on GCN applied on Beam3D	104
6.15 GT visualisation for the MLR framework of the Plane DT for specific data sample	98	6.26 MD for DT regressor based on SAGE applied on Beam3D	104
6.16 GT visualisation for the FN framework of the Plane DT for specific data sample	98	6.27 MD for DT regressor based on MLR applied on Fibonacci's spring	105
6.17 GT visualisation for the GCN framework of the Plane DT for specific data sample	99	6.28 MD for DT regressor based on FN applied on Fibonacci's spring	105
6.18 GT visualisation for the SAGE framework of the Plane DT for specific data sample	99	6.29 MD for DT regressor based on GCN applied on Fibonacci's spring	106
6.19 MD for DT regressor based on MLR applied on Beam2D	101	6.30 MD for DT-based regressor on SAGE applied on Fibonacci's spring	106
6.20 MD for DT regressor based on FN applied on Beam2D	101	6.31 MD for DT regressor based on MLR applied on Plane	107
6.21 MD for DT regressor based on GCN applied on Beam2D	102	6.32 MD for DT regressor based on FN applied on Plane	107
		6.33 MD for DT regressor based on GCN applied on Plane	108

6.34 MD for DT regressor based on SAGE applied on Plane	108	6.45 Experiments training history of GCN regressor on Fibonacci' spring dataset	115
6.35 Experiments training history of MLR regressor on Beam2D dataset	110	6.46 Experiments training history of SAGE regressor on Fibonacci' spring dataset	115
6.36 Experiments training history of FN regressor on Beam2D dataset	110	6.47 Experiments training history of MLR regressor on Plane dataset .	116
6.37 Experiments training history of GCN regressor on Beam2D dataset	111	6.48 Experiments training history of FN regressor on Plane dataset . . .	116
6.38 Experiments training history of SAGE regressor on Beam2D dataset	111	6.49 Experiments training history of GCN regressor on Plane dataset .	117
6.39 Experiments training history of MLR regressor on Beam3D dataset	112	6.50 Experiments training history of SAGE regressor on Plane dataset	117
6.40 Experiments training history of FN regressor on Beam3D dataset	112	6.51 Boxplots for prediction accuracy $\hat{y}(\sigma)$ of the smaller nodes count datasets: Beam 2D, Beam 3D . .	118
6.41 Experiments training history of GCN regressor on Beam3D dataset	113	6.52 Boxplots for prediction accuracy $\hat{y}(\sigma)$ of the higher nodes count datasets: Fibonacci Spring, Plane	118
6.42 Experiments training history of SAGE regressor on Beam3D dataset	113		
6.43 Experiments training history of MLR regressor on Fibonacci' spring dataset	114		
6.44 Experiments training history of FN regressor on Fibonacci' spring dataset	114		

Tables

5.1 FE models summary by its statistics of structure features	84
5.2 Graphs statistics	85
6.1 Summary Table of RMAXSE on validation sets for GF dataset	89
6.2 The training time required for an individual model in seconds	90
D.1 Table of results - proposed techniques compared to benchmark.	140
D.2 Summary table for GCNframework listing the proposed reduction strategies and their respective performance on presented datasets	141
D.3 Summary table for SAGE framework listing the proposed reduction strategies and their respective performance on presented datasets	142



List of Notations

Symbol	Meaning
DT	Digital Twin
HM	Hybrid model / technique
PM	Physical-based model / technique
DD	Data-driven model / technique
GNN	Graph Neural Network
Γ	Geometry PM
Ω	Solid Physical Based Model
\mathcal{E}	Material elasticity parameter of PM
ν	Material Poisson's constant of material PM
e	a finite element piece (certain typography) of FEM structure
n	node of finite element or graph
σ_n	Maximal Principal stress at dedicated node
δ	simulation step for calculation PM
\mathcal{D}	Dataset generated via FE
\mathcal{G}	Graph
\mathcal{G}'	Reduced \mathcal{G}
v	vertice (edge) of \mathcal{G}
A	Adjacency matrix of \mathcal{G}
D	Degree matrix of \mathcal{G}
L	Laplacian matrix of \mathcal{G}
\mathcal{D}	Dataset to train a regressor
LR	Linear regression
MLR	Multi-input-output linear regressor framework
$FFNN$	Feed-Forward Neural Networks
FN	Regressor based on multi-layer neural feed-forward neural network
GCN	Regressor defined by multiple graph convolutional operational neural neural network
$SAGE$	Regressor based on SAGE aggregation layers for graph neural neural network
\mathcal{L}	loss function for regressor training
e	error value between target and prediction
MD	Model Diagnostic
GT	Ground Truth




Introduction

This thesis explores the concept of mechanical digital twins, focusing on their status quo and advancements in physical modelling, data-driven modelling, and the emerging field of hybrid modelling. Digital twins are virtual replicas of physical objects, systems, or processes that enable real-time monitoring, analysis, and optimization. They have gained significant attention recently due to their potential in various industries, including manufacturing, healthcare, smart cities, etc.

The first part of this thesis will examine the current state of digital twins, examining their definition, key components, and underlying technologies. It will also provide an overview of the challenges and opportunities associated with their implementation.

The initial part of the thesis will also specifically investigate advancements in digital twin modelling techniques, focusing on physical modelling, data-driven modelling, and integrating both approaches through hybrid modelling. Physical modelling involves creating a digital replica of the physical system using mathematical equations, while data-driven modelling leverages large datasets to develop predictive models. Hybrid modelling combines the strengths of both approaches, allowing for a more accurate and comprehensive representation of the physical system.

The second part of this thesis focuses on the numerical demonstration of mechanical digital twins, explicitly investigating the process of distilling data-driven models from datasets obtained through physical-based models. This experimental approach aims to compare different frameworks and establish a



baseline for effectively extracting data-driven models from the information generated by physical models.

In this part of the thesis, a series of experiments will be conducted to compare various frameworks for distilling data-driven models from datasets obtained from physical-based models. The physical-based models serve as the baseline for generating the datasets, which are then used to develop data-driven models. These data-driven models are evaluated and compared in terms of their accuracy, efficiency, and suitability for representing the behaviour of the mechanical system.

This research aims to provide insights into the effectiveness of different frameworks for extracting data-driven models from physical-based models by conducting these experiments. The findings will contribute to understanding how to leverage the advantages of physical and data-driven modelling in developing hybrid models for mechanical digital twins.

The results of this study will be beneficial for researchers and practitioners in mechanical engineering and digital twins as the guidebook for selecting appropriate frameworks for distilling data. By analyzing and comparing these modelling approaches, this research aims to provide insights into the strengths and limitations of each method and identify the potential synergies that can be achieved through hybrid modelling. The findings of this study will contribute to the existing knowledge in the field of digital twins and offer valuable guidance for practitioners and researchers interested in leveraging digital twins for enhanced system understanding and optimization.



Part I

Hybrid Modelling of Digital Twins



Chapter 1

Status Quo of Digital Twin Concept

The Digital Twin technique (DT) concept can be defined as an adaptive model of a physical asset. A digital representation of behaving like a real-time mechanical structure can be built with a combination of various mathematical modelling techniques. Therefore, an original mechanical system is additionally altered by the benefits of state-of-the-art technologies. Multi-physical-based solvers, cybernetics of big data, artificial intelligence, and augmented and virtual reality are well-known technologies today. Yet, the mutual collaboration within DT is probably not evident in many fields.

The subject of DT is relatively new in comparison to other simulation methods; for instance, the number of already published articles is around fifteen thousand per two years (2020, 2023) only in the Science Direct database. Below are further representative papers to build a brief picture of the research vision.

DT might be defined as another milestone in the Evolution of simulations of the physical world, where a combination with current technologies is expected. The milestone is well surveyed at [38] and is expected to be a successor of the Product Lifecycle Management (PLM) tool, which can be understood as an approach to managing databases of specific mathematical models of a product as depicted at Fig.1.1.

The most likely pioneering publication [39] mentioning the DT concept yet again only as a sub-group of PLM also suggests further definitions of individual components to be aggregated in a DT model. Although the article

The following article [40] provides a more comprehensive overview of DT-connected aspects. The complete definition describes the whole concept to apply an entire DT concept as a valuable tool for making the right decisions throughout various parties in an organisation using specific DT. Discussed and highlighted are fields that have already started using the approach to some extent, for instance, in industry, healthcare, meteorology, and even education with individualised and adaptive student plans.

Additionally, in this article, DT is understood as an adaptive model, surprisingly enhanced as a metamodel of complex physical systems with a specific organisation and labelled as a state-of-the-art asset. Interestingly, a brief introduction of one possible view of assessing DT as a component of Industry 4.0 is made with pipelines to connect technologies such as the Internet of Things to create advanced applications. The suggested method of DT usage could be easily applied since multi-functional sensors are already available and are expected to be even more affordable due to the effect of technology on demand. This sharp image also leads to the involvement of other technologies, starting with Tensor Processing Units (TPU), used widely in machine learning applications, or even further used with Quantum Processing Units (QPU), where the price of those technologies is also expected to decrease due to the Moor law. The mentioned units have a primary high potential to make the required calculation needed for DT imitation of a physical asset faster, and the response of a model should be immediately available, for example, if connected with a 5G telecommunication network.

Logically, too many open questions remain to be solved for this fresh scientific sub-field, for instance, a question concerning cyber safety, since leakage of data from a DT might be at least expected. Another vital issue is the pre-processing of data heading to the DT model, and this is from the practical aspect of false predictions of a system by a designed model, which might lead to fatal errors. Making a complete list of unresolved topics affecting the DT concept would be fruitful.

1.1 Representative DT applications

The first best application to start with is a project named "The Living Heart" [41] from Dassault, which shows how DT could be helpful in healthcare as illustrated by Fig.1.2 and Fig.1.3. The application does not necessarily aim to build the DT model of the human heart explicitly. Still, the approach of usage is intuitively well suited to be an example of how a finite element model as DTP can be a helpful tool to make a goal-oriented decision in succeeding with planning and then executing the procedure of cardiostimulator implantation for an individual patient. Another example within the DDI context can be the calibration of a model ventricle to correctly describe parameters of a new material imitating biological tissue of the heart [42], which might have the potential for further usage as a DDA component. The approaches mentioned above can already be distinguished as the correct connection between components of DT architecture.

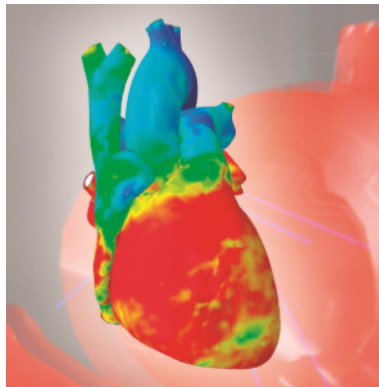


Figure 1.2: Living heart [41]

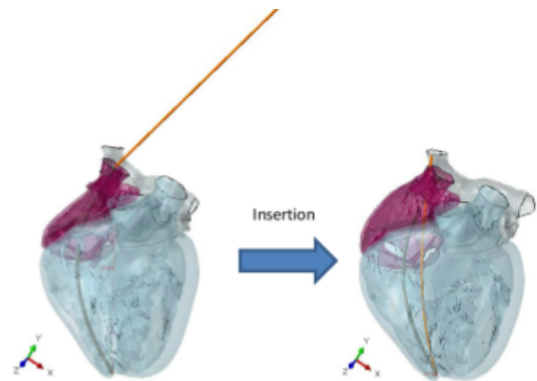


Figure 1.3: Procedure planning [42]

The company Rolls Royce is usually identified as a pioneer in bringing DT application to the real world with their product of jet engines used in aeroplanes [43]. The "Intelligent engine" tool is probably the first industry application to predict engine conditions. The state of an engine is monitored in real-time and evaluated based on available data received from developed models of other engines and, therefore, an agile decision of subsequent actions can be made, for example, whether a plane has to be guided from air to service check immediately with an object collision as illustrated in Fig.1.4. This DT product imitation of a physical engine can be used for other purposes, such as maintenance planning, estimation of reliability, or predicting the engine state under a specific weather forecast for a planned air route.



Figure 1.4: Intelligent Engine [43]

In the Industry 4.0 domain, great effort is usually put into making a production process more effective economically and beyond. Suggested article [46] describes an approach to solving this aspect by the method of artificial intelligence, by Reinforcement Learning (RL), with agents designed to converge to a solution of ideal autonomous industry management. The core of the idea is to connect the supply chain and warehouse into one architecture to create an adaptive system within this industry environment. Prerequisites for this application are defined as two groups of agents, with a fixed policy of learning and an agile policy. The first group of agents is learning based on separate data from the environment, which can be natural or synthetic. The next group of agents understands the architecture simultaneously from both data batches acting within the same system and expects stochastic events to be considered; therefore, the dynamically changed policy function is required.

1.3 Hybrid Modelling

The Hybrid Modelling technique (HM) can be described as a union of physical-based and data-driven modelling (there can also be a perception of modelling between multiple mathematical approaches) as illustrated in Fig.1.5. Furthermore, this approach might be understood as a fundamental pillar for building a DT of a specific system, for instance, mimicking structural mechanic behaviour.

The basic proposition upon the argument of feasible adaptation of HM technique to build a DT is expected when the following method can enhance a primary driving modelling approach describing observed specific complex structure.

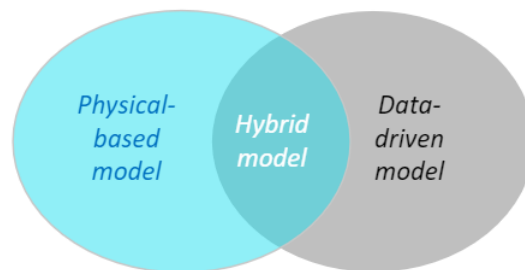


Figure 1.5: Venn diagram to depict union of Physical- based and Data-Driven modelling techniques

To illustrate how potential architecture based on different modelling techniques might look alike, let us consider an example of a prototype aeroplane. The plane's wings are measured at multiple locations by sensors, and the main objective is to investigate if the design of the entire aircraft is in safe conditions. Unfortunately, the decision of risks cannot be made on a physical-based model since the technique is mainly computationally demanding. Therefore, the delayed results for further plane control would not be valuable at a specific flight moment. The solution might be to build a model based on data transferred from the knowledge of a large physical model, and this distilled data-driven model is then utilized to control the loop.

In fundamental perception, HM combines two or more different models, such as a regressor based on neural networks or another, to create a more robust and accurate model. As a result, the combined model is often more precise than the individual models alone. In addition, hybrid models usually use data from multiple sources and leverage the strengths of both models to provide better predictions.

network to produce a more representative estimate to improve the composite properties. This model is trained on a data set where the experimental measurements complement the results from FEM simulations to account for the different energy absorption behaviour in differently rotated fibres of the composite.

■ Dynamic System - Surrogate

Although the issue of DD was discussed in the previous section, here is an additional example. The reported approach of metamodeling a dynamic system using surrogate models [45] should be mentioned in this chapter. This paper describes a damped discrete dynamical system to discover a suitable DD concept using surrogate models (SM). The exciting aspect is the comparison of SM, such as a Gaussian process emulator, on a system that varies its stiffness and mass. Other data quality factors, such as sampling rate, are also included, which is crucial for developing a DT and subsequent behaviour.

The examples mentioned above collectively convey a strong sense of establishment within fields such as machine learning for physics, instilling confidence in the well-founded nature of the overall concept [4].

■ 1.4 Physical-Based Modelling

Physical-based technique models (PM) are an essential component of techniques aimed at creating DT through hybrid approaches that combine data-driven and physics-based methods. By incorporating fundamental physical principles, PM models provide a deeper insight into the system's behaviour being modelled, allowing for accurate predictions of its response to various stimuli. This might be particularly important in complex systems where more than data-driven models (further Section 1.5) are needed to capture the underlying physics. Integrating data-driven and PM models in a hybrid approach makes it possible to create more accurate and robust DTs that can be used for various applications, such as optimization, control, and decision-making. Furthermore, developing and using PM models can lead to new insights and discoveries in the field, driving innovation and progress in various industries.

PM models are established through analytical reconstruction of physical phenomena, ranging from simple linear relationships to more complex mathematical tools such as Partial Differential Equations (PDEs) that describe the

modelled physical system. However, PDEs often require enormous computing power to solve a problem precisely and capture the observed system. This computational capacity can be a disadvantage in some applications, whether the solution is implicit or explicit. Nevertheless, PM models based on PDEs can still provide valuable insights into the behaviour of physical systems and serve as a basis for developing numerical models. By combining analytical and numerical techniques, powerful models can be created to understand and predict physical systems' behaviour.

1.4.1 Overview of Solving PDEs for a PM

There are several methods to solve partial differential equations (PDEs) numerically [37; 2], including:

- **Finite Difference Method (FDM):** This method involves approximating the derivatives in the PDE using finite differences and then solving the resulting system of algebraic equations. FDM is a straightforward and efficient method for solving PDEs, but it is limited by its accuracy and its ability to handle complex geometries.
- **Finite Element Method (FEM):** This method involves dividing the domain of the problem into small finite elements and then approximating the solution within each element using a polynomial function. The equations governing the behaviour of each element are then assembled into a system of algebraic equations and solved numerically. FEM is a versatile method that can handle complex geometries and boundary conditions, but it can be computationally expensive and requires significant expertise to implement.
- **Finite Volume Method (FVM):** This method involves dividing the domain of the problem into small finite volumes and then applying a conservation law to each volume. The equations governing the behaviour of each volume are then assembled into a system of algebraic equations and solved numerically. FVM is particularly useful for solving problems involving fluid dynamics and heat transfer, and it can handle complex geometries and boundary conditions.
- **Spectral Methods:** This method involves approximating the solution to the PDE using a set of basis functions, such as Fourier series or Chebyshev polynomials. The coefficients of the basis functions are then determined by minimizing the residual error in the PDE. Spectral methods are highly accurate and efficient, but they can be limited by their ability to handle complex geometries and boundary conditions.

Naturally, each method has its strengths and weaknesses, and the choice of method depends on the problem being solved and the resources available for computation.

1.4.2 Fields Excerpts of PM

Physical modelling has so many approaches, and the publications that have been written would be too overwhelming for this thesis to touch even the tip of the iceberg of the PM topic. Therefore, only a few of the standard techniques of PM are listed:

Structural Mechanics

Poisson's law, in relation to Hooke's law, describes the behaviour of materials under deformation essential for structural mechanics [1]. For example, Poisson's law describes a material's lateral contraction or expansion occurring when it is stretched or compressed. In contrast, Hooke's law describes the linear relationship between stress and strain in an elastic material.

Poisson's law states that when a material is stretched or compressed in one direction, it will experience a corresponding contraction or expansion in the perpendicular directions. The ratio of the transverse strain to the longitudinal strain is called the Poisson's ratio, ν , and it is a material property that can be used to describe the deformation behaviour of materials.

A simple example of a material bar being stretched in one direction shows the relationship between Poisson's ratio and Hooke's law. When the bar is stretched, it experiences a longitudinal strain (change in length per unit length) in the direction of the applied force. At the same time, the material experiences a transverse strain (change in width per unit width) in the perpendicular direction.

The relationship between the longitudinal and transverse strains can be expressed as: $\epsilon_L = -\nu\epsilon_T$ where ϵ_L is the longitudinal strain, ϵ_T is the transverse strain, and ν is the Poisson's ratio. By substituting this expression into Hooke's law, we get: $\sigma = E\epsilon_L = -\nu E\epsilon_T$. This shows that Poisson's ratio is related to Young's modulus, which can be used to predict the behaviour of materials under different loading conditions. In summary, Poisson's law can be used in conjunction with Hooke's law to describe the deformation

PM can also design and optimize systems, allowing engineers to explore different configurations and operating conditions before building physical prototypes. This can reduce the risk of costly design errors and shorten the design cycle.

Moreover, system simulations can help develop control strategies for complex systems, allowing engineers to optimize the system's performance and stability under a wide range of operating conditions.

Overall, system simulations based on PM can provide a cost-effective and efficient means of exploring the behaviour of complex systems, optimizing their design and operation, and developing control strategies to achieve desired performance objectives.

This section presents a comprehensive introduction to PM as a powerful tool for understanding, designing, and optimizing complex systems by simulating their behaviour based on known physical laws and relationships.

The selected representation is based on three main approaches to physical modelling: Hookean, MBS, and system simulations. Hookean models are based on the linear relationship between mechanical stress and strain and can be used to predict the behaviour of materials under different loading conditions. On the other hand, MBS describes the behaviour of interconnected rigid or flexible bodies and is often used in designing mechanical systems such as vehicles and robots. Finally, system simulations based on physical-based modelling can provide a cost-effective means of exploring the behaviour of complex systems, optimizing their design and operation, and developing control strategies to achieve desired performance objectives.

By understanding the principles of physical-based modelling and its different approaches, tools might provide valuable insight into the behaviour of complex systems, optimize their designs, and improve their performance.

1.5 Data-Driven Modelling

Contrary to the previous section, there is a powerful technique for describing observed systems via fine-tuned complex mathematical and statistical models where knowledge of a system, in contrast to physical modelling, is not required for the first steps. Is that appropriate? This question could lead to another dissertation thesis. Let us assume that the Data-Driven model (DD) is crucial

model estimates the coefficients for each feature to minimise the sum of squared errors between the predicted and actual target values.

- **Polynomial regression:** Polynomial regression extends linear regression by incorporating polynomial terms of the input features. It allows for capturing non-linear relationships between the features and the target variable. By including higher-order terms, the model can capture more complex patterns in the data.
- **Surrogate modelling, also known as response surface modelling,** is a technique used in engineering, optimisation, and simulation to create a computationally efficient approximation (surrogate) of a complex and computationally expensive model or system. It aims to capture the relationship between the inputs and outputs of the original model/system using a more straightforward and faster-to-evaluate surrogate model. The surrogate model is trained using a set of input-output data obtained from the original model/system. The data points typically cover a range of input values and corresponding output values. The surrogate model can be of various types, including regression models (e.g., linear regression, polynomial regression), Gaussian processes, support vector machines, or artificial neural networks.
- **Ensemble methods:** Ensemble methods combine multiple individual regressors to create a more robust and accurate model. Techniques like random forests and gradient boosting are popular ensemble methods for regression. Random forests use an ensemble of decision trees, while gradient boosting combines multiple weak regressors sequentially to form a strong regressor.
- **Neural Networks:** In recent years, using neural networks for multiple input-output regression has gained significant popularity in DD modelling. This technique has become a favourite among researchers and practitioners due to the neural networks' remarkable ability to capture complex non-linear relationships within the data. With their interconnected layers of neurons, neural networks excel at learning intricate patterns and dependencies in multi-dimensional datasets. These models approximate sophisticated mappings between input features and multiple output variables by iteratively optimising their internal parameters through advanced optimisation algorithms. As a result, neural networks have emerged as a favoured choice for regression tasks involving multiple variables, solidifying their prominence in recent years.

By combining these techniques and iteratively refining the model, a regressor well-suited for DD modelling can be created. It can then capture the patterns and relationships in the data and make accurate predictions.

- Can be more accurate and reliable than DD models under certain conditions.
- Can provide insights into the system behaviour that can be used to guide experimental design and further study.
- Can extrapolate well beyond the range of data used to build the model, which can be useful in predicting behaviour under different conditions.

Cons of PB modelling:

- Can be difficult and time-consuming to develop and implement, especially for complex systems.
- Can require detailed knowledge of the modelled system and the physical principles involved.
- Can be sensitive to uncertainties in the input parameters and assumptions used in the model.
- Can be computationally expensive to simulate, especially for large-scale systems. It may not be able to capture all of the relevant physics and may require simplifying assumptions that can limit its accuracy and applicability.

■ 1.5.3 GNNs for the use as DD

The architecture of the latest type of neural network is representative of the DD modelling technique. What is quite interesting is Graph Neural Network (GNN) might be understood as a more corresponding architecture similar to the biological brain. The view stems from the lay notion that the brain does not have an input and output layer of neurons, as with Feed-Forward Neural Networks (FFNN), but has regions that are variously activated at any given time. Specifically, specific clusters of neurons are activated by different stimuli. Another comparison with FNs is that the biological brain is not made up of neat layers of neurons but is more like a graph arrangement, where specific neurons are connected to neighbouring ones.

GNN uses a node and edge modelling approach to model data instances. Relationships between neighbouring nodes are modelled using edges, which may or may not have a specified direction of information transfer. A graph can be used to describe many systems, from social networks, for example, to determining voting preferences or defining a map of a country to estimating the weather, but obviously to be used for FEM connectivity. Nodes already

model be computed at different levels of accuracy, thus choosing a compromise to the computational effort. The overall approach is demonstrated on a traditional partial differential equation problem for physical prediction in robotics to learn and predict image scenes from new perspectives.

How powerful a tool GNN is is the aim of the paper [52], where GNN is mentioned as an efficient tool for graph description. The neighbourhood clustering scheme is highlighted here. This consists of a vector representation of a given node, which is recursively computed to cluster and transform the vectors of neighbouring nodes. It is pointed out that many variants of GNN have already been proposed to achieve results in node and graph classification. Despite this revolutionary description of graph learning, there is still a limited understanding of GNN features and their limitations. This fact is considered, and a theoretical approach is presented to analyze and evaluate the performance of GNNs for different graph structures.

Interestingly, the results highlight the discriminative performance of popular GNN variants such as Graph Convolutional Networks and architecture called GraphSAGE [54] and demonstrate that this type of GNN cannot learn more straightforward graph representation other than the one they are primarily built on. The paper presents a simple architecture developed for this purpose, which the authors believe is the unique topology among GNNs. This claim is based on the testing performed using the Weisfeiler-Lehman graph isomorphism test [56; 54]. Thus, it is empirically verified that the proposed architecture achieved high performance among other types of GNNs.

The GNN transforms the properties of each neighbouring edge into a vector representation of that edge. Similarly, each edge representation is used to predict its value. This standard procedure implicitly assumes that edge labels are independently conditioned on their neighbourhoods. This assumption is essential in a paper dealing with residual correlation [6], and real-world examples are presented to show that this assumption is valid. By focusing on regression problems, conditional independence is discovered to be a limiting predictive power in several assumptions, which should not be surprising, according to the author, for the chosen traditional graph methods based on learning with or without a teacher, such as propagating edge labelling in the opposite direction using explicit modelling. According to the paper, there is a correlation between the predictions' output and

At this point, the authors note a problem with interpretability and efficiency and recommend improving GNN architecture by including correlation structure in the regression residuals. It is partially modelled using a multivariate Gaussian distribution applied in the residuals played by the GNN. The parameters are then estimated by maximizing the marginal likelihood

of the observed edges. The authors say this approach is more accurate than the selected baseline architectures. The most significant advantage is that the learned parameters can be further interpreted as an amplified correlation between neighbouring edges.

Another contribution may be the developed linear time algorithm for low variance, which estimates unbiased parameters so that they can be incorporated into more extensive and complex GNN structures.

As may be evident at first glance, GNN is still evolving, even fundamentally, where testing processes that have been around for decades in classical deep learning networks are only being standardized. Standardization as an essential aspect is addressed in a paper on performance measurement [57], which proposes the primary datasets (dataset) on which it is recommended to test future architectures of these new neural networks. For example, the chemical dataset ZINC [30], which contains real-world molecular graphs, is a tool to test the effectiveness and identify the most appropriate architecture for graph classification or regression tasks. It is proposed to use the well-known dataset MNIST [29] and CIFAR [31] for graph-constructed images using superpixels.

1.6 Distinct Possible Applications Based on GNN

For example, a geometric representation suitable for demonstrating graph structure could be an "ice cube". However, this is not just any ice cube but the South Pole Neutrino Observatory, whose primary goal is to study the high-energy neutrinos produced by particle accelerators that produce high-energy cosmic rays. The paper [58] applies GNN to classify more than 5k signals detecting Cherenkov radiation as the response of high-energy neutrinos acting on Arctic ice. Experimentation with GNN is compared against the physical modelling of this problem, where the GNN method can classify the submitted signals up to 3 times better. At the same time, the method can evaluate up to 6 times the signal volume of the standard evaluation approach.

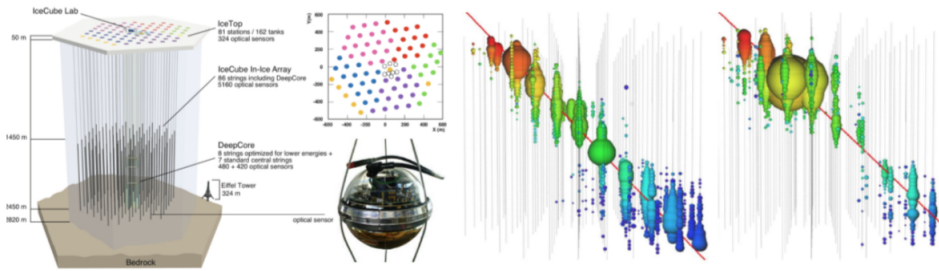


Figure 1.6: Visualization data of Ice Cube experiment for detection of neutrino particles [58].

It would be a pity not to complement the work with another human system, as a representative of DTI 1, namely a publication dealing with constructing a model of the human lung based on CT scans [59]. Graph refinement based on the obtained sub-graphs to complete the overall graph can be applied to many problems. In the presented work, tree or sub-tree-shaped structures are extracted from the image data to derive a graph representation of the volumetric data, which is further refined.

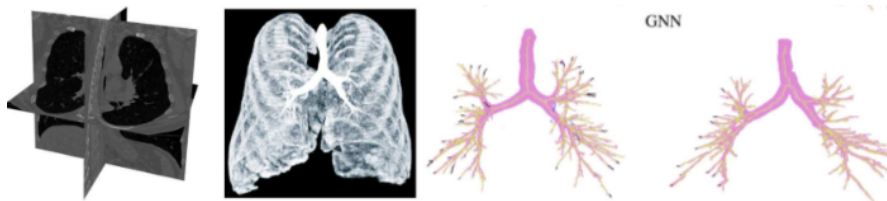


Figure 1.7: From CT scan to the virtual representation of biological system [59].

One of the most exciting publications is a paper dealing with the regression problem of dynamically changing GNN patterns [49]. Here, the issue of nodes being variously connected over time is addressed by using a unique neuron, the gated diffusive unit as depicted on Fig.1.8. The unit has a time stamp involved and can temporally describe the inputs to neighbouring nodes of GNN. The proposed approach is only at the theoretical level so far and has yet to be applied to a real example.

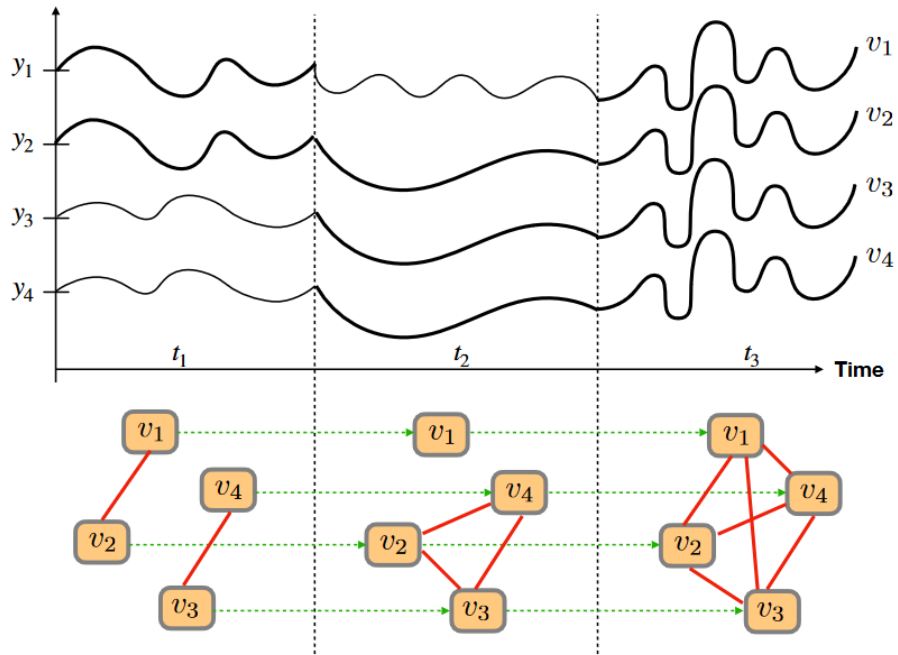


Figure 1.8: Graph Neural Lasso for Dynamic Network Regression [49].

There is no better application to conclude a brief GNN overview than combining DT with reinforcement learning techniques. Dynamic scheduling of flexible production based on Petri nets [48] is broadly understood by the authors of this paper. The goal here is to propose a policy for scheduling the operation of an intelligent plant using a combination of the mentioned DD techniques demonstrated by Fig.1.9.

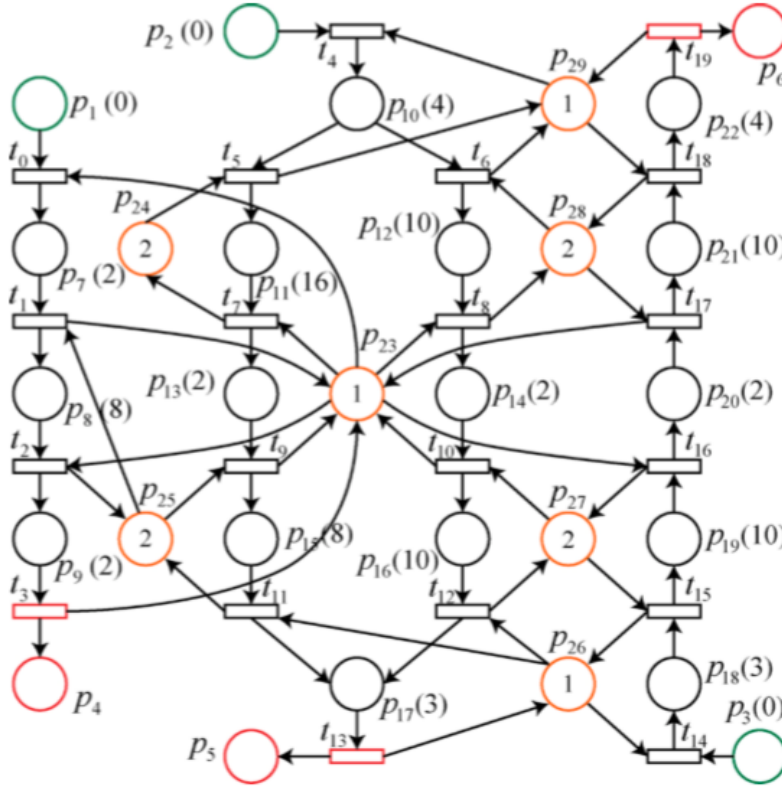


Figure 1.9: Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network [48].

Self-optimizing GNN [5] is included in the application so that a Petri Net can be used to describe what and how resources are shared in production. What is surprising about this approach is that aspects such as whether flexible production paths can be used and even if the aspect of stochastic material delivery for production is included. The proposed architecture using a Petri Net for their characterization of the Bipartite (even) graph is an elegant solution, as the manufacturing unit is modelled using a sequential topology, and no unnecessary large sponge convolutional network is applied, as often seen in similar applications. The authors do not propose any new technique for reinforcement learning to optimize the strategy as demonstrated at Fig.1.10. Instead, the classical deep Q net is used here, where PSs are defined as a Markov decision process.

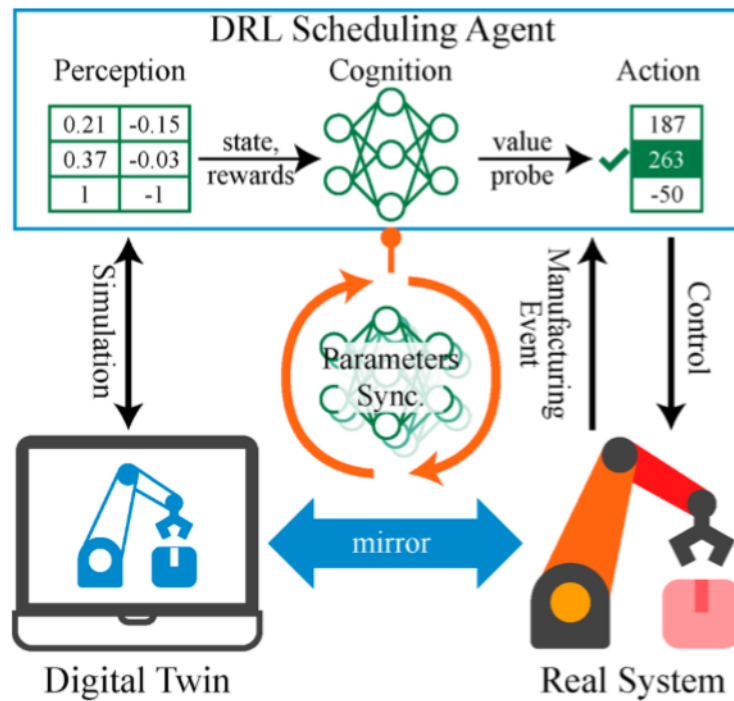


Figure 1.10: Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network [5].

1.7 Conclusion on The State-of-The-Art

In the previous three sections, the main blocks for finding inspiration for the thesis have been described. The examples of real-world applications described in each block were presented to the reader to outline the primary intention as much as possible, namely in compiling all three topics into one DD design methodology. The creation of a DT using a PM, for example, with the help of FEM and, in particular, the use of this information to build graphical neural networks.

In the first part, the main purpose was to determine the status quo and, therefore, examine the broader definition of the DD concept. The definition is still unclear, but the core of the matter is clear from the beginning.

Virtual imitation of a physical system with the help of some mathematical approaches has been an essential tool in the development of various products for several decades. However, the DD concept is only understood as another milestone in the field of simulation rather than an entirely new scientific field, as it might seem at first glance. This chapter also identified other

characteristics a DD should have, which are summarised at the end of this chapter. The following part of the state-of-the-art was conceived as a reservoir of inspiration for linking different mathematical modelling approaches. Combining PM and DD techniques is expected to provide advantages with both methods. Thanks to PM, a fast DD should be obtained from the PM technique, which might be used further and retrospectively interpretably.

The final section of the study describes GNNs to provide an introduction and brief link to the following chapter, which deals with the basis for the entire idea of the dissertation thesis. GNNs are presented as a new type of milestone in the field of neural networks and, therefore, in the DD technique.

Publications are selected so that the reader can derive an initial insight into this new subfield of artificial intelligence. It is implicitly highlighted that even though a GNN is still taking shape, some challenges for standard FFNN were efficiently handled by GNN.

1.8 Problem statement

The central hypothesis of this dissertation thesis is to answer whether GNN can be applied as an effective DT modelling technique to store and evaluate information from the physical description of a mechanical system. The main goal is to achieve similiarity of experimental data of a structural mechanic system such that the resulting interpretability for the user will be similar to that of an extracted FEM.

It is also assumed that a GNN is a suitable architecture compared to classical deep-learning neural networks. The assumption is based on the fact that GNN can use the information of nodes and the relationships between them using a 3D model initialized and essential in FEM.

The mechanical stress results of the loaded system in FEM are suitable for creating a dataset for training GNN, which will be faster for load classification as a result of specific nodes of the mechanical system.

Chapter 2

Hypothesis and Goals of Dissertation Thesis

2.1 Formulation of a hypothesis

The central hypothesis of this dissertation thesis is to answer whether GNN can be applied as an effective DT modelling technique to store and evaluate information from the PM-based description of a mechanical system. The main goal is to achieve similarity of experimental data of a structural mechanic system so that the resulting interpretability for the user will be similar to extracted FEM.

It is also assumed that a GNN is a more suitable architecture than classical deep-learning neural networks. This is because GNN can use the information of nodes and the relationships between them using a 3D model initialized in FEM. The mechanical stress results of the loaded system in FEM are suitable for creating a dataset for training GNN, which will be faster for load classification as a result of specific nodes of the mechanical system.

2.2 Thesis Objectives

For the dissertation, it is proposed to investigate and clarify the following issues to develop a methodology applicable to establishing a DT of a mechanical system by combining various modelling techniques.

1. How to design a methodology for predicting a particular mechanical system for authentic and accurate operation and overall establish a Digital Twin architecture of a mechanical system with the combination of FEM and GNN techniques?
2. How to extract information from an FE model representing a mechanical system and compile it to ensure that the graphs required to train a particular regressor are properly defined?
3. How to train regressors optimally so that they can be used to perform regression tasks on nodes of graphs reflecting a PM, and what FEM data will be chosen to build the training dataset?
4. How can the overall DT model be diagnosed to avoid false system predictions based on a GNN regressor so that DT mimics data that will potentially be taken from sensors in the regular operation of the physical asset?



Chapter 3

Methods for Hybrid Modelling of Digital Twin

To address the problem of predicting physical behaviour and characteristics of complex systems, several computational methods have been required to define them. These methods range from mathematical modelling to numerical simulations and machine learning algorithms. In the following lines, a combination of FE analysis as the tool for physical modelling and neural graph networks will be employed to model the behaviour of a physical system subject to the Poisson boundary. Specifically, the variational problem formulation is to derive a set of partial differential equations that describe the system's behaviour and then discretise these equations using finite element analysis to solve them numerically. Additionally, extracting a graph representation of the mechanical system will be described with the aim of further use in GNNs. The objective is to learn the relationship between the mechanical system's behaviour and its underlying structure. Combining these methods aims to develop a more comprehensive understanding of the physical system and provide accurate predictions of its behaviour.

The optimal strategy to do so, the following structure, was identified. To appropriately set the overview of methods required to have the background on building a DD model with a hybrid approach, the focus is targeted on three main topics.

Firstly, a description of Poisson's problem for finite element variational formulation, which is a fundamental method for modelling physical systems, is taking place. Then, this topic will be enhanced by incorporating graph theory and DD modelling, focusing on GNNs. Despite the apparent incoherence

between these topics at first glance, the chapter is organized to provide a proper technique for building a DD model by combining a PM approach with stored knowledge of the mechanical system. This aims to comprehensively understand the system's behaviour and provide accurate predictions for its performance.

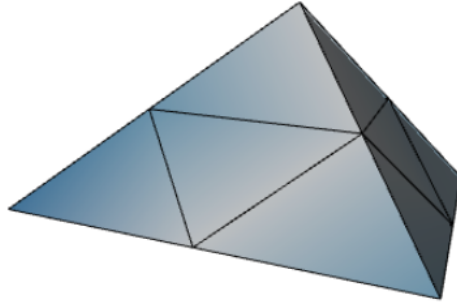


Figure 3.1: Illustrative example of physical model Ω based on tetrahedral elements building specific FEM.

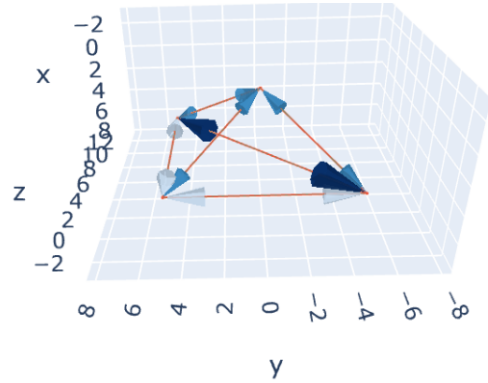


Figure 3.2: Illustrative example of directed graph \mathcal{G} extracted from structure Ω .

The following paragraphs are devoted to the topic of designing DT by the usage of GNNs with a combination of FEM to store knowledge of the PM mechanical systems. For this purpose, it is necessary to define the whole mechanical system in FEM software. Then, this representation Fig.3.1 can be transferred to the graph as illustrated with Fig.3.2 and further to a neural network. Further utilisation targets training a GNN to closely imitate a real system's behaviour.

The first three subsections aim to describe the essential topics of DT, GNN and FEM briefly. Those are necessary ingredients for the final compilation of a hybrid mechanical system model. The following section then describes how to pre-process and create GNN based on the geometry extracted from the FEM model representation.

3.1 Physical-based Modelling via Finite Element Technique

Physical-based modelling via FEM technique has proven to be an effective method for modelling complex physical systems. In this technique, the physical system is divided into small elements, and a set of partial differential equations is derived to describe the behaviour of each element. These equations are then discretized and solved numerically using the FEM, which

involves approximating the solution over the entire domain of the physical system. This approach allows for capturing the system's behaviour with high accuracy, considering the effects of various physical phenomena such as stresses, strains, and temperature variations. The primary keys are mathematical tools in the form of Poisson's and Boundary condition problems.

3.1.1 Poisson Problem

The Poisson problem is a fundamental mathematical equation that can be applied to specific physical phenomena. They are named after the French mathematician Siméon Denis Poisson. The most common type of Poisson equation is the Laplace equation, which describes the behaviour of a static electric field. However, the equation can also be used to model a variety of physical contexts, including solid mechanics, heat conduction, fluid flow, and diffusion.

For instance, the Poisson problem with fixed boundary conditions U and specific solid geometry Γ , may be introduced on the two-dimensional square beam with sides of length and width a, b , and the material parameters consist of Young's modulus E and Poisson's constant ν . The beam is subject to a concentrated load F on the opposite side to boundary conditions, together as the entire condition $\Phi(U, F)$ per Fig.3.3. The equations governing small elastic deformations of an Ω geometrical structure can be written as:

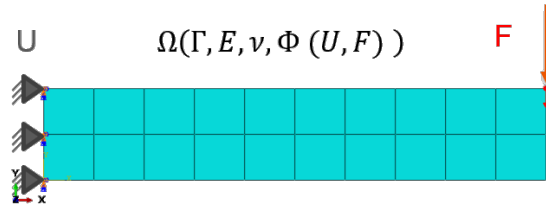


Figure 3.3: Accompanying example of two-dimensional beam for Poisson problem description

$$-\nabla \cdot \sigma = F \in \Omega \tag{3.1}$$

$$\sigma = \lambda \text{tr}(\varepsilon) I + 2\mu \varepsilon \tag{3.2}$$

$$\varepsilon = \frac{1}{2} (\nabla u + (\nabla u)^T) \quad (3.3)$$

Where σ is the stress tensor, f is the body force per unit volume, λ and μ are the elasticity parameters of the material in Ω , I is the identity tensor, tr is the trace operator of a tensor, ε is the symmetric unitary strain tensor (symmetric gradient), and u (or enhanced $u(x, y)$) is the displacement vector field. In this problem, isotropic elastic conditions are assumed [1; 2].

If we combine two previous 3.2, 3.3 equations than we obtaine:

$$\sigma = \lambda (\nabla \cdot u) I + \mu (\nabla u + (\nabla u)^T) \quad (3.4)$$

Note that equations 3.1-3.3 can be easily transformed into a single vector of PDEs for u , which is the PDE for the unknown u (Navier's equation) [2]. However, it is convenient to keep the equations separate, as above, in deriving the variational formulation.

3.1.2 Variational Formulation

The variational formulation of 3.1-3.3 consists of taking the inner product of 3.1 and a test vector function v belonging to \hat{V} , where \hat{V} is a vector-valued test function space. Then, integrating over the domain Ω the following equation is obtained:

$$- \int_{\Omega} (\nabla \cdot \sigma) \cdot v dx = \int_{\Omega} f \cdot v dx \quad (3.5)$$

The gradient of σ contains second-order derivatives of the primary unknown u , which is integrated into this term by parts (although there is a similarity with integration by parts, what is done is to apply Green's theorem, which is why integral over the boundary appears):

$$- \int_{\Omega} (\nabla \cdot \sigma) \cdot v dx = \int_{\Omega} \sigma : \nabla v dx - \int_{\partial\Omega} (\sigma \cdot n) ds \quad (3.6)$$

The symbol $:$ represents the inner product between tensors (the sum of the element-wise product), and n is the unit normal pointing outward on the boundary. The quantity σn is known as the traction or stress tensor on the boundary and is often prescribed as a boundary condition. Here, it is assumed that it is specified on a part of the boundary as T . On the remaining part of the boundary, we assume that the displacement values are given as a Dirichlet boundary condition[2; 3]. Then is obtained:

$$\int_{\Omega} \sigma : \nabla v dx = \int_{\Omega} f \cdot v dx + \int_{\partial\Omega} T \cdot v ds \quad (3.7)$$

Inserting the expression 3.4 for sigma gives the variational form with u as the unknown. Note that the boundary integral on the remaining part disappears due to the Dirichlet condition. The variational formulation can be summarized as follows: find u belonging to V so that:

$$a(u, v) = L(v) \quad \forall v \in \hat{V} \quad (3.8)$$

Where,

$$a(u, v) = \int_{\Omega} \sigma : \nabla v dx \quad (3.9)$$

$$\sigma(u) = \lambda(\nabla \cdot u) I + \mu(\nabla u + (\nabla u)^T) \quad (3.10)$$

$$L(v) = \int_{\Omega} f \cdot v dx + \int_{\partial\Omega} T \cdot v ds \quad (3.11)$$

Suppose the gradient is expressed of v as a sum of its parts. In that case, only the symmetric part will survive the product between sigma and the gradient of v because sigma is a symmetric tensor. Therefore, replacing the gradient of u with the symmetric gradient of u 's strain tensor, ϵ , yields the following slightly different variational form:

$$a(u, v) = \int_{\Omega} \sigma(u) : \epsilon(v) dx \quad (3.12)$$

Where epsilon of v is the symmetric part of the gradient of v :

$$\epsilon(v) = \frac{1}{2}(\nabla v + (\nabla v)^T) \quad (3.13)$$

As all the necessary information is in place, the stress requirements for the structure can be calculated. For instance, Von Mises's stress is expressed as:

$$\sigma = \sqrt{\frac{3}{2} s : s} \tag{3.14}$$

where s is the deviatoric tensor

$$s = \sigma - \frac{1}{3} \text{tr}(\sigma) I \tag{3.15}$$

The deviatoric tensor is the basis for deriving other stress values, such as the maximal principle stress visualized at Fig.3.4, shear stress, and hydrostatic stress [3].

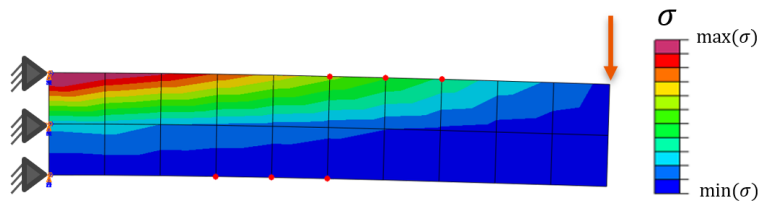


Figure 3.4: Stress distribution on accompanying descriptive example of two-dimensional beam

The mathematical tool discussed here is a robust framework for analyzing structural mechanics and defining stress tensors. By leveraging an understanding of known physical phenomena, such as dynamics and transient temperature variations, these further phenomena might be seamless and easily adapted into the analysis. This versatility allows for a more accurate and realistic structural performance assessment, paving the way for enhanced design, optimization, and reliability in various engineering applications.

3.2 Graph Theory and Poisson Problem

Graph theory might be for someone exciting branch of mathematics that studies applying graphs to various problems in many fields, including mathematics, computer science, physics, chemistry, and economics. In the field, a graph is a collection of objects (called vertices or nodes) connected by lines (called edges). Hence FEM has, at first glance, a structure similar to graphs; it is worth introducing to this theory. Furthermore, this foundation will be helpful to the section dealing with graph neural networks.

A tool for mutual interaction must be identified to transfer the Poisson problem from a mechanical engineering context to a graph theory. The problem can be represented using a graph. Specifically, it can be created as a graph where the nodes represent the vertices of a specific element from a finite mesh of structure Ω , and the edges represent the connections between the nodes. Furthermore, the applicability to other geometrical shape topologies of the element (Voxel, Hexahedral, Pentagonal, Hexagonal) might be quickly adopted.

As the graph at least minimum consists of source and target nodes N connected by edges V (at pure graph theory used nodes U and vertices V notation, to make synergy between domains, the notation is therefore slightly modified), where each node connects only one, the proper way to extract a suitable graph \mathcal{G}

$$\mathcal{G} = (N, V) \tag{3.16}$$

To solve the Poisson problem using graph theory and graph neural networks, the graph Laplacian operator is a useful matrix that describes the graph's connectivity. The graph Laplacian can be used to define a discrete version of the Poisson equation on the graph:

$$\mathbf{L}\mathbf{u} = -\mathbf{f} \tag{3.17}$$

where \mathbf{L} is the graph Laplacian matrix, \mathbf{u} is the vector of nodal displacements, and \mathbf{f} is the vector of nodal loads. The boundary conditions can be incorporated into the graph Laplacian matrix by modifying the appropriate entries.

To solve this equation using graph neural networks, We can use a neural network to learn the nodal stress (strain, displacement), given the graph Laplacian matrix and the nodal loads. The neural network can be trained using supervised learning, where examples of known nodal values and corresponding loads are provided, and the network learns to generalize to new examples.

This approach has the advantage of allowing us to solve Poisson's problem on irregular meshes or graphs, which may be more efficient or easier to work with than a traditional FEM. Additionally, graph neural networks can learn to generalize to new meshes or graphs, making them useful for solving Poisson's problems in various settings.

$$L_{norm} = \frac{L_0}{\|L_0\|} = L_0^T \cdot \|L_0\|^{-1} \quad (3.18)$$

The graph Laplacian version of the Poisson equation in a tetrahedral domain represented as a graph can be written as:

$$\mathbf{L}\mathbf{u} = -\mathbf{f} \quad (3.19)$$

where \mathbf{L} is the graph Laplacian matrix (more at 3.4.4), \mathbf{u} is the vector of nodal displacements, and \mathbf{f} is the vector of nodal loads.

The boundary conditions can be incorporated into the graph Laplacian matrix by modifying the appropriate entries to reflect the fixed displacement at the boundary nodes.

The goal of using a graph neural network is to learn the mapping from the nodal loads to the nodal displacements. This can be represented using a neural network f as:

$$\mathbf{u} = f(\mathbf{L}, \mathbf{f}), \quad (3.20)$$

where f is a neural network that takes as input the graph Laplacian matrix \mathbf{L} and the nodal loads \mathbf{f} , and outputs the nodal displacements \mathbf{u} . The neural network can be trained using supervised learning, where there are examples of known nodal values (displacements, stresses, strains) and corresponding loads. The network is then trained to learn and generalize to new examples.

■ 3.3 Graph Extraction from Physical Based Model

Once the physically based model Ω is established, its geometric and domain description is documented via a protocol called the usual input file. Such a protocol has consistently described each node n_i and element e_i from a spatial perspective and mutual connectivity between elements.

$$f : \Omega(\Gamma(e, n), E, \nu, F) \rightarrow \mathcal{G}(u, v) \quad (3.21)$$

Two main approaches can be identified for graph extraction strategies suitable for storing and transporting information from a physically based model built using the FEM technique. Identified possible appropriate strategies for graph extraction from input files are:

- Graph extraction based on nodal structure presented in [72]
- Graph extraction from element centroids [23]

■ 3.3.1 Graph Extraction from Nodal Structure of an Element

The primary goal of this strategy is to use the information available to each element from the mesh, which is a discretisation of the entire geometric structure. The main idea is to use each node of the element as a node of the graph, with the links between graph nodes being established based on edges for the specified element.

Typically, the protocol has written the node information on a row-by-row basis. A row starts with a numerically labelled specific node and continues for its spatial position within the structure.

```

** Structure (Omega)
*Node
  1,          0.,          0.
  2,          5.5,          0.
  3,          11.,          0.
  4,          16.5,         0.
  5,          22.,          0.
  6,          27.5,         0.
  7,          33.,          0.

```

Listing 3.1: Excerpt of Node section from input file

Similarly, all structure-defining elements are written to their respective input file section. The section begins by specifying element typology, which usually differs from software (in the following listing, the CPS4 is the 2D element used in Abaqus terminology). Still, from the logical perspective, element shape is the primary information for graph creation. Here, for each element with a label, the set of nodes that make up the element is specified and is written to the element's row.

```

*Element, type=quadralirela element
  1,  1,  2, 13, 12
  2,  2,  3, 14, 13
  3,  3,  4, 15, 14
  4,  4,  5, 16, 15
  5,  5,  6, 17, 16

```

Listing 3.2: Excerpt of Element section from input file

The algorithm intended to provide the graph is crucial in providing all necessary compilation from physically based structure defined via its nodes and elements. For the graph regression on the nodes, as for the task objective, the direction of the edges is not essential; the key is only on the connectivity between all the nodes within an element, which is the only necessary aspect to be considered. For this reason, the algorithm is designed, for example, for a structure consisting of a tetra-element as follows

Algorithm 1 Graph extraction from structure based on tetrahedral element

Ensure: $G = Graph$

```

1: for  $nd_i$  in  $nodes$  do
2:    $src \leftarrow nd_i$  ▷ source node
3:    $src_{pos} \leftarrow [src[x], src[y], src[z]]$ 
4:    $G \leftarrow \mathbf{addNode}(src_{pos})$ 
5: end for
6: for  $i$  in  $elements$  do
7:    $e_i \leftarrow nodes_i$ 
8:    $e_{ai} \leftarrow \mathit{reshape}(e_i, 2, 2).T$ 
9:    $e_{bi} \leftarrow \mathit{reshape}(e_i, 2, 2)$ 
10:   $e_{ci} \leftarrow \mathit{roll}(e_i, 1)$ 
11:   $G \leftarrow \mathbf{addEdges}(e_{ai})$ 
12:   $G \leftarrow \mathbf{addEdges}(e_{bi})$ 
13:   $G \leftarrow \mathbf{addEdges}(e_{ci})$ 
14: end for
15: return  $G$ 

```

Furthermore, the algorithm used to generate the graph for the DT based on the tetra-element can also be applied similarly to the hexahedral typology. In a hexahedral element, each node is connected to the other three close neighbourhood nodes within the element. This can be represented in a graph format similar to the tetra-element. Therefore, the algorithm can be extended

to consider the hexahedral (sometimes as qu) typology by considering the connectivity between the nodes within each hexahedral element.

Algorithm 2 Graph extraction from structure based on hexagonal element

Ensure: $G = Graph$

```

1: for  $nd_i$  in  $nodes$  do
2:    $src \leftarrow nd_i$  ▷ source node
3:    $src_{pos} \leftarrow [src[x], src[y], src[z]]$ 
4:    $G \leftarrow \mathbf{addNode}(src_{pos})$ 
5: end for
6: for  $i$  in  $elements$  do
7:    $e_i \leftarrow nodes_i$ 
8:    $e_{ai} \leftarrow \mathit{reshape}(e_i, 2, 4).T$ 
9:    $e_{bi} \leftarrow \mathit{reshape}(e_{i(0:4)}, 2, 4)$ 
10:   $e_{ci} \leftarrow \mathit{reshape}(e_{i(4:end)}, 2, 4)$ 
11:   $e_{di} \leftarrow \mathit{roll}(e_{bi}, 1)$ 
12:   $e_{ei} \leftarrow \mathit{roll}(e_{ci}, 1)$ 
13:   $G \leftarrow \mathbf{addEdges}(e_{ai})$ 
14:   $G \leftarrow \mathbf{addEdges}(e_{di}, e_{ei})$ 
15: end for
16: return  $G$ 

```

Quadrilateral elements are commonly used in two-dimensional FEM simulations, while tetrahedral and hexahedral elements are used in three-dimensional FEM simulations. However, hexahedral elements are often preferred over quadrilateral elements when simulating structures with more complex geometries and require more accurate behaviour modelling. Another difference is quadrilateral elements can only deform in a plane, while hexahedral elements can deform in three dimensions. This means that hexahedral elements can provide more accurate simulations of three-dimensional structures, while quadrilateral elements are better suited for analysing two-dimensional structures [47]. Since the next chapter dealing with numerical demonstration considers one use case- beam 2D, consisting of quadrilateral elements- the following algorithm has been established similarly to previous typologies suited for 3D structures.

Algorithm 3 Graph extraction from structure based on quadrilateral element

Ensure: $G = \text{Graph}$

```

1: for  $nd_i$  in  $nodes$  do
2:    $src \leftarrow nd_i$  ▷ source node
3:    $src_{pos} \leftarrow [src[x], src[y]]$ 
4:    $G \leftarrow \text{addNode}(src_{pos})$ 
5: end for
6: for  $i$  in  $elements$  do
7:    $e_i \leftarrow nodes_i$ 
8:    $e_{ai} \leftarrow roll(e_i, 1)$ 
9:    $G \leftarrow \text{addEdges}(e_{ai})$ 
10: end for
11: return  $G$ 

```

The other typologies of elements used for discretising the structure can be easily adapted in the same way as presented for the three main ones mentioned above. Since the graphs are derived, the structure representation of Ω can be further stored in an adjacency list or matrix.

3.4 Initial Tools From Graph Theory To Establish Synergy With Finite Element Technique

As mentioned in the previous section, a graph is at least minimally composed of source and target nodes N connected by edges V , where each node connects only one, the correct way to extract a suitable graph \mathcal{G} with connections written in the adjacency matrix A [61].

Graph preparation based on the FE model can be approached from multiple perspectives. One way is to store information such as material properties and element types as scalars to differentiate connections. Contact or joint definitions can also be prescribed on vertices, further enhancing the graph representation. This consistent and reliable approach ensures a robust representation of the FE model and facilitates accurate analysis in subsequent experiments.

$$\mathcal{G} = (N, V, A) \tag{3.22}$$

3.4.1 Adjacency Matrix

The adjacency matrix A defines the mutual relationship between specific nodes. For example, the Directional heterogeneous graph as the fundamental input to GNN has its directions defined via this matrix. The adjacency matrix of a graph represents the connections between nodes in the graph. It is a binary matrix where the element in row i and column j is one if there is an edge connecting node i and node j , and 0 otherwise. The adjacency matrix can be represented as A . Adjacency matrix, for example, of tetra element per Fig.3.5, would look as follows for the undirected element graph.

$$A_{tetra} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

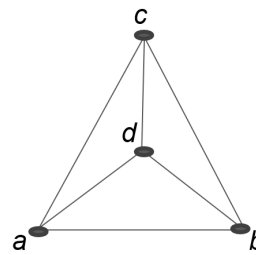


Figure 3.5: Tetra element represents as an undirected graph \mathcal{G} by adjacency matrix A

The directional graph of the accompanying Fig.3.6 has the A_{tetra} then expressed as follows:

$$A_{tetra} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}. \quad (3.23)$$

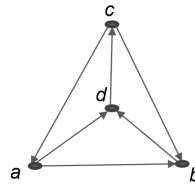


Figure 3.6: Tetra element represented as directed graph A

3.4.2 Adjacency List

An adjacency list is a collection of unordered lists used to represent a finite graph. Each unordered list within an adjacency list describes the set of neighbours of a particular vertex in the graph. The adjacency list is the optimal way to store graphs for further processes [24; 61].

3.4.3 Degree Matrix

A degree matrix D is a valuable tool in neural graph networks because it contains information about the local structure of the graph. The degree matrix is a diagonal matrix where the i -th diagonal entry is the degree of the i -th node in the graph, i.e., the number of edges incident to the node. The degree matrix can be used in several ways, which is further important in GNN usage. One common approach is normalising the adjacency matrix by the degree matrix to obtain a row-normalised adjacency matrix. This normalisation helps to account for the varying degrees of nodes in the graph and prevents nodes with higher degrees from dominating the representation of the graph. Another way to use the degree matrix in GNNs is to incorporate it into the node embeddings. Specifically, the degree matrix can be used as a diagonal weight matrix to scale the node embeddings before feeding them into the GNN. This scaling can help to balance the importance of low-degree and high-degree nodes in the representation of the graph.

The expression of the previous tetra-directed graph example is then for Out degree (D^{out}) and In degree D^{in} matrixes expressed as:

$$D^{out} = \begin{vmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{vmatrix} \quad (3.24) \quad D^{in} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3.25)$$

3.4.4 Laplacian Matrix

The Laplacian matrix L of a graph is derived from the adjacency, and the degree matrix represents the graph's structure. The Laplacian matrix is a positive semi-definite matrix that reflects the connectivity and topology of the graph [60].

$$L = D - A \quad (3.26)$$

The Laplacian matrix is often used in GNNs [63] to encode the graph structure and capture information about node relationships. It is also used as a regularisation term to smooth the signals across the graph.

In summary, the adjacency matrix represents the connections between nodes in a graph, while the Laplacian matrix, which is derived from the adjacency matrix, represents the graph's structure. The Laplacian matrix is often used in GNNs to encode the graph structure and capture information about node relationships.

3.5 Graphs Reduction

State of the art in Graph Reductions is constantly evolving. Recent advances have focused on developing algorithms for graph reduction that are more efficient and accurate than traditional methods. These advances include techniques such as graph simplification, graph sparsification, graph clustering, and graph summarisation. In addition, researchers have been exploring machine learning and deep learning techniques to improve graph reduction algorithms further elaborated at [24].

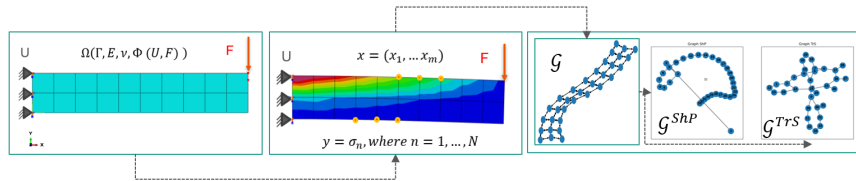


Figure 3.7: Process of DT creation from FEM to Graph representation with further graph reduction process; from left: mechanical model, FEM, fully connected graph, reduced graph (shortest path), reduced graph (closed path)

There are many different types of graph reductions, and the choice of reduction technique depends on the problem being solved. Some common types of graph reductions include:

- Vertex elimination: Removing a vertex and all its incident edges while preserving the connectivity of the remaining graph.
- Edge contraction: Replacing two adjacent vertices and their incident edges with a single vertex while preserving the connectivity of the remaining graph.
- Subgraph extraction: Identifying a smaller subgraph within a larger graph with certain desired properties and removing all other vertices and edges.
- Simplification: Removing vertices and edges that do not affect certain properties of interest, such as the existence of a Hamiltonian cycle or the chromatic number of the graph.

Graph reduction can be a powerful tool for simplifying complex graphs and making them easier to incorporate into DT. However, it's important to choose the right reduction technique for the problem at hand and to ensure that the reduction doesn't inadvertently change the properties of interest.

For FEM extracted graph, the edge reduction type of task is taken place to have all nodes connected at least once into a graph. The graph structure incorporation can be adapted to graph reduction using graph simplification, graph sparsification, graph clustering, and graph summarisation algorithms. These techniques can be used to reduce the complexity of FEM and improve the accuracy of their results. In addition, machine learning and deep learning techniques can be applied further to enhance the accuracy and efficiency of FEM techniques. The following main methods were identified as suitable for further elaboration:

- Shortest Path (\mathcal{G}^{ShP}),
- Solution for Traveling Salesman (\mathcal{G}^{TrS}),
- Random Laplacian Reduction (\mathcal{G}^{L1}),
- Weighted Laplacian Reduction (\mathcal{G}^{WL1}),
- Weighted Laplacian Reduction II. (\mathcal{G}^{WL2}).

■ Shortest Path

The shortest path connectivity is introduced to streamline the structural information. This involves assigning weights to the edges of the graph \mathcal{G}^{ShP} based on the final step of the FEM simulation. The weights are determined considering the stress, strain, or other relevant parameters from the simulation results. This ensures that the graph weights reflect the structural characteristics observed in the FEM simulation's concluding phase. The implementation of the Dijkstra method in the NetworkX library [20] is used to calculate the shortest weighted path between two nodes in a graph. The graphical overview of graph creation and pruning is shown in Fig.3.7.

■ Traveling Salesman Problem Solution for Graph reduction

This approach is based on reducing only the edges in the graphs while keeping all the nodes. Therefore, this reduced graph is cyclic, and its creation is similar to solving a well-known travelling salesman problem (TSP). This problem is generally defined as finding the shortest possible route that visits each node of the original graph and returns to the starting node. In the context of graph reduction, the TSP solution aims to identify a subset of edges from the original graph that form a cycle traversing all nodes with minimal total edge weight.

The TSP can be solved in nondeterministic polynomial time, which makes it an NP-hard problem. Therefore, searching for the best solution by brute-force testing all solutions is not feasible. Thus, it is possible to obtain only an approximate solution. This study uses the default algorithm for the directed TSP solution (Threshold accepting [18]) implemented in the NetworkX library. The algorithm optimises the route by minimising the total distance travelled or the cost incurred when visiting each node exactly once.

It is important to note that, in the context of FEMs, the values associated with the edges often represent information regarding the mechanical stress distribution, such as the maximum principal stress. These values are derived from the elements of the FEM and provide crucial insight into the structural behaviour of the system.

Once the TSP solution is obtained, the selected subset of edges forms the reduced graph, which retains the original graph's essential connectivity while reducing computational complexity. With stress distribution information encoded in its edges, this reduced graph can then be used for various applications, such as optimisation, network analysis, or machine learning tasks.

■ Spectral Reduction

The third method in our investigation involves the application of the Laplacian Matrix for Dimensionality Reduction inspired by [16]. By leveraging the eigenvalues of the Laplacian Matrix, we aim to distil the inherent structures of the FEM graph. This reduction not only simplifies the computational complexity but also retains the essential characteristics of the structural interconnections. Our exploration of Laplacian-based reduction is guided by the notion that critical structural information can be efficiently preserved within a lower-dimensional representation. Additionally, We coarsen graph edges v by a similar approach [19] where a threshold γ is used to distinguish the importance of an edge weight.

In the spectral reduction approach, we first calculate the eigenvalues and eigenvectors of the Laplacian matrix, which is derived from the graph's adjacency matrix. The Laplacian matrix is Hermitian [60], allowing us to compute its eigenvalues and eigenvectors efficiently. These eigenvectors capture essential structural characteristics of the graph. We then sort the eigenvectors based on their corresponding eigenvalues and select the largest ones, representing the graph's dominant modes. The number of eigenvectors to retain is determined by a reduction factor, which specifies the desired size

of the reduced graph. By discarding the smallest eigenvectors, we effectively reduce the dimensionality of the graph while preserving its essential structural features.

■ Weighted Spectral Reduction

In our approach to spectral reduction with edge weights, we utilise pruning techniques to refine the reduced graph while preserving essential structural features. In our approach to spectral reduction with edge weights, we utilise pruning techniques to refine the reduced graph while preserving crucial structural features. Our pruning method does not require iteration of the training line in [17], and We incorporate an Inclusion of mechanical stress values $\Sigma \in v_i(\sigma)$ as edge weights provide valuable information on the system's load distribution and stress concentrations. This is compared to the approach using edge pruning with saliency matrix [14] or pruning all the nodes and edges as it is at [15], which is, for our purposes, not feasible. By setting a threshold parameter γ , we establish criteria for pruning the graph, removing edges with insignificant weight contributions. This pruning ensures that only the most influential edges are retained in the reduced graph, leading to a more focused representation of the structural dynamics. Furthermore, considering the eigenvalues and eigenvectors derived from the Hermitian matrix representation of the weighted graph, we identify the most significant structural modes and prioritise their preservation during the reduction process. Through this combined approach of spectral reduction and pruning, we achieve a balance between computational efficiency and structural precision, enabling efficient analysis and simulation of complex mechanical systems. It is introduced and compiled to pseudo-algorithm 4 for better clarity under the suggested approach.

■ 3.6 Graph Neural Networks

As mentioned in the first part of the thesis, Graph Neural Network (GNN) is the youngest Artificial Neural Network technique. It is characterised by the need to have and see specific patterns that a model with a particular architecture could learn, in this case, in the form of graphs. Since the previous sections cover all the necessary background, the transition with fundamentals to this exciting topic is uncovered by a brief description of tasks that GNN can reliably handle. The most common techniques for defining a GNN will then be presented.

■ 3.6.1 Tasks Covered by GNNs

- graph classification: probably the most likely in the chemistry industry to find a new potential required chemical compound
- graph node classification: for instance, in sociology, a voting preference in the public sector
- graph link-edge prediction: similar to the previous
- regression on nodes or edges: the task well suited to create a digital twin distilled from PM knowledge

■ 3.6.2 Message Passing

Message passing is a fundamental concept in GNNs that propagates information between nodes in a graph. In message passing, each node sends a message to its neighbouring nodes, which then update their states based on the received messages. This process is typically repeated multiple times, allowing information to propagate throughout the Graph. Message passing in GNNs typically requires the following processes:

1. Node Embedding: Represent each node in the Graph as a vector of features.
2. Aggregation: Compute a representation of the neighbourhood of each node by aggregating the node embeddings of its neighbours.
3. Update: Update the node embedding for each node based on the aggregated neighbourhoods of its neighbours.
4. Readout: Compute a graph-level representation from the updated node embeddings.

Message passing [50] [51] is similar to forward propagation, well known in classical feed-forward neural networks. Probably the most initial model, to begin with, is linear regression adapted to multi-input/output tasks,

$$Y_{IDS}^{\hat{}} = \mathbf{W} \cdot \mathbf{X}_{IN} + \mathbf{b} \quad (3.27)$$

where $Y_{IDS}^{\hat{}}$ is the predicted target vector of dataset nodes, \mathbf{W} is matrix of weights, input features from dataset \mathbf{X}_{IN} (further, only \mathbf{X}_0) and bias vector b .

Both techniques involve propagating information through a network of nodes. In forward propagation, the information is propagated from input to output nodes. In message passing, the information is propagated between nodes in a graph structure, where the nodes can have different relationships. The basic equation for message passing is for the first hidden layer of GNN, then ($i = 0$) can be rewritten as follows:

$$H_1 = \sigma \left(\mathbf{W}_0^T \cdot \mathbf{X} + \mathbf{b}_0 \right) \quad (3.28)$$

, where the chosen activation function for the further experiment is Rectified linear unit $\sigma = \max(0, X)$ or others (sigmoid, tangent hyperbolic).

3.6.3 Fundamental Layers for GNN

- Graph convolutional Layer Graph Convolutional Neural Network [8] can be understood as the enhancement of FFNN by the graph-structured data described with consideration of adjacency matrix A . The forward pass for the first hidden layer is then

$$H_1 = \hat{D}^{-1/2} \cdot \hat{A} \cdot \hat{D}^{-1/2} \cdot \mathbf{X} \cdot \mathbf{W}_0 + \mathbf{b}_0 \quad (3.29)$$

- Graph Sample and Aggregated Embeddings Layer

The SAGE layer is a type of GNN layer that aggregates information from a node's neighbourhood to generate a node representation.

The Sage layer first computes the node embeddings of the neighbouring nodes, then aggregates them to generate a new node embedding. The aggregated embedding is then used to update the target node embedding. This process is repeated until all nodes in the Graph have been updated. As a type of GNN layer, the sage layer [9; 10] aggregates information from a node's neighbourhood to generate a node representation. The forward pass is expressed

$$H_1 = [AGG(\mathbf{X}) || \mathbf{X}] \mathbf{W}_0 + \mathbf{b}_0 \quad (3.30)$$

, where AGG is a function aggregating neighbourhood nodes with a certain aggregation method (sum, mean, min, max).

- The Chebyshev layer [11] is characterised by its utilisation of Chebyshev polynomials as basis functions for convolutional operations. It applies spectral filtering to capture local and non-local dependencies in the input data while learning filter weights through training.

$$H_1 = \sigma \left(\sum_{k=0}^{K-1} \sum_{j=0}^{J-1} \mathbf{T}_k \cdot \mathbf{W}_{kj} \cdot \mathbf{X}_{IN} \cdot \mathbf{T}_j^T + \mathbf{b} \right) \quad (3.31)$$

where σ is the activation function applied element-wise to the output, and $\mathbf{T}_k \mathbf{W}_j$ are the Chebyshev polynomials of degree k and j respectively.

The main difference between Chebyshev, Graph Convolutional Layers, and SAGE is how they process data. Chebyshev layers use polynomial approximation to perform convolutions on graph data, Graph Convolutional Layers use graph-structured filters to capture local neighbourhood information, and SAGE (or GraphSage) uses aggregated neighbour features to learn node-level features.

3.7 Graph Reduction to Optimise Size of Regressor

Once the regressor is trained and fully diagnosed, an additional tuning step involves reducing the graph size [60], thereby reducing the time required for training [66]. The proposed research [70] harnesses graph theory utilising graphs extracted for the following foundations of training GNNs. This approach combines white box modelling with machine learning, as demonstrated in various domains such as medical imaging and physics-performed machine learning [64]. However, the increasing scale and complexity of engineering projects pose challenges in computational intensity associated with detailed FEMs. Addressing this challenge, the research focuses on graph reduction to streamline computational complexity and lay the groundwork for efficient GNN training [65]. At the methodology's core lies the integration of GNNs, transcending traditional simulation boundaries by learning intricate relationships embedded in structural graphs and predicting behaviours beyond conventional modelling. The synergy between graph reduction and GNNs is pivotal in achieving an accurate DT, empowering neural networks to predict with unprecedented accuracy, optimising computational efficiency, and establishing a robust framework for real-time predictive modelling.

3.8 Summary for HM via FEM and GNN Methods

A few key factors can impact the performance of a GNN model with a simple architecture compared to a FFNN. Here are a few examples:

- Data complexity: If the data is more complex or has more underlying patterns and relationships, a GNN model may be more effective at capturing these patterns and performing well on the task. On the other hand, if the data is relatively simple, an FFNN may be sufficient.
- Amount of data: The amount of data available for training can also impact the performance of the two models. In general, more data can help to improve the performance of both GNNs and FFNNs, but the

relative benefit of additional data may be more significant for GNNs due to their ability to capture complex patterns in the data.

- **Model architecture:** The specific architecture of the GNN and FFNN models can also impact their performance. For example, a GNN model with a more complex or profound architecture may perform better than a simpler GNN model but may also be more challenging to train and require more data.
- **Task complexity:** The complexity of the task being performed can also impact the performance of the two models. If the task is relatively simple, an FFNN may be sufficient, but if the task is more complex and requires the ability to capture more complex patterns in the data, a GNN may be more effective.

A number of factors can impact the performance of a GNN model with a simple architecture when compared to an FFNN. These include the complexity of the data, the amount of data available, the model architecture, and the complexity of the task being performed.

■ 3.9 Dataset Acquisition

Accurately distilling information from simulations of PM presented, particularly those established via the FE, is of utmost importance. These models are invaluable tools for understanding complex phenomena, predicting outcomes, and making informed decisions. However, their reliability hinges on a rigorous validation process. This involves subjecting the model to knowledgeable critical inspection by experts in the respective field and supporting it with real-world laboratory experiments. By adhering to this validation framework, we can ensure that the distilled knowledge from the model aligns with empirical evidence, enhancing its accuracy, robustness, and overall credibility. This, in turn, empowers us to unravel the intricacies of the natural world, optimize designs, mitigate risks, and ultimately make well-informed decisions for a better future. This is visualized by demonstrative Fig.3.8.

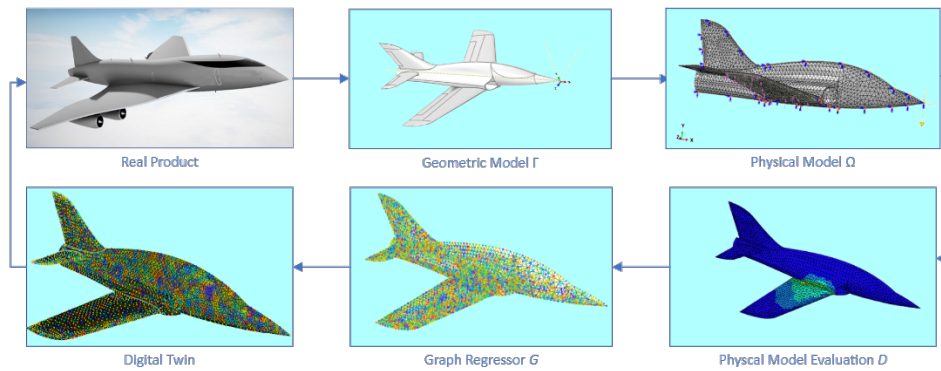


Figure 3.8: Data workflow of DT

The accompanying image illustrates the workflow of a DT for an authentic product, highlighting the integration of a regressor trained on a physical-based model. The workflow begins with developing a physical-based model that captures the real product’s behaviour and characteristics. This model is the foundation for training the regressor, which learns the relationships between input variables and output predictions. Once the regressor is trained, the DT enters an operational phase, where it can make predictions and simulate the behaviour of the real product in a virtual environment. However, a closed loop is established to ensure the DT’s accuracy and reliability. This loop incorporates the expectation of real-time measurements from the physical product.

Assuming the regressor remains current, the DT can continuously compare its predictions with real-time measurements. This enables the system to adjust, validate the model’s accuracy, and refine its predictions over time. In addition, the closed-loop feedback mechanism ensures that the DT remains aligned with the physical product’s actual behaviour, enhancing its effectiveness as a virtual replica. The depicted workflow showcases the integration of a regressor trained on a physical-based model within a DT framework. This enables the DT to leverage real-time measurements and maintain its accuracy and reliability, providing valuable insights and predictive capabilities for optimizing the real product’s performance and maintenance.

This closed-loop feedback mechanism plays a crucial role in maintaining the DT’s fidelity. It allows for continuous validation and refinement of the physical-based model, ensuring it remains aligned with the real product’s behaviour as it evolves and changes over time. This, in turn, enables more accurate predictions, better decision-making, and improved optimization of the physical system.

Recent articles exploring this topic [12], [13] have delved into various aspects of closed-loop DT workflows, including techniques for sensor integration, data processing and analysis, model updating and adaptation, and the overall impact on system performance and maintenance. These articles provide valuable insights into the advancements and challenges associated with implementing closed-loop feedback in DT systems, contributing to this promising technology's ongoing development and refinement.

Once the FEM is established, its geometric and domain description is documented through a computational environment, which defines the spatial configuration of each node n_i within a particular element e_i , as well as the mutual connectivity between the elements. The graph model \mathcal{G} can be obtained as follows:

$$f : \Omega(\Gamma(e, n), E, \nu, F) \rightarrow \mathcal{G}_{FEM}(N, V, \mathbf{X}(\mathbf{F}), \mathbf{y}(\sigma)) \quad (3.32)$$

All nodes and edges in the resulting graph adhere to the logic of the original FEM. The nodes acquire data from the converged FEM, with input parameters X capturing reaction forces F_i of the selected nodes $n_i \in \mathcal{S}$ simulating distributed sensors. The target output \mathbf{y} represents structural mechanical stress for all nodes σ in light of Fig.3.9. This approach were introduced at [72].

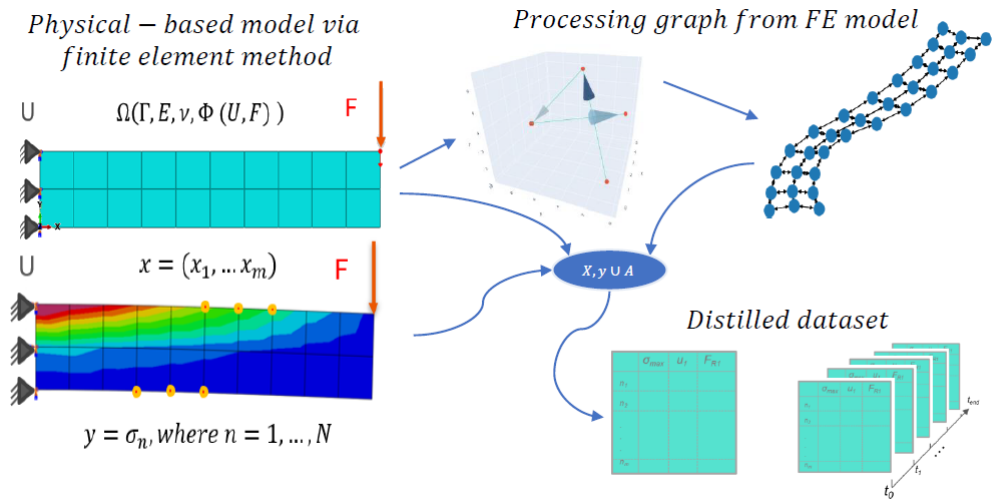


Figure 3.9: Data workflow from Physical model distilling information

■ 3.9.1 Synthetic Data

DTn is from the primary intrinsic goal understood as a tool giving a user immediate information. Still, for establishing DT, the detailed information is based on assumptions, premises, hypotheses, and overall requirements, and obviously, the distilled model is imprecise from the beginning. This might be solved by building information models [33].

Usually, the accuracy of the regressor is strongly influenced by the data density from the physically based model. The data obtained are, on the one hand, pseudo-accurate, but on the other hand, the noise is not depicted in any form. Therefore, the raised assumption of noise depicted in the natural mechanical system is welcomed, along with the additional minor aspect that the synthetically increased dataset density might help improve a regressor's robustness. The inspiration to distil further data-free knowledge can be gained at [32; 34].

The data mainly used to control the mechanical system is usually very expensive to acquire, as it can be obtained from actual measurement processes via sensors or physics-based models. Therefore, the dataset created is very demanding from a cost of training computational time perspective compared to applications such as image object detection and language models.

■ 3.9.2 Synthetic Data: Strategies to Enhance Knowledge Base of DT

One approach to creating synthetic data that is interpolated between existing values in a dataset is to use a "generative modelling" technique. Generative models are machine learning algorithms that learn to generate new data similar to a given dataset.

One popular type of generative model is a Variational AutoEncoder (VAE) [25]. A VAE consists of two parts: an "encoder" that maps the input data to a lower-dimensional "latent" space and a "decoder" that maps points in the latent space back to the original data space. During training, the VAE learns to encode the input data into a distribution in the latent space and then sample from that distribution to generate new, synthetic data. To use a VAE to create synthetic data that is interpolated between existing values in a dataset would first train the VAE on the existing dataset. Then, to generate new synthetic samples, it would choose two or more points in the latent space corresponding to nearby data points in the original dataset and interpolate between them by sampling from the distribution.

One potential challenge with this approach is that the VAE may not learn to generate samples that are exactly in between existing values in the dataset but rather samples that are close to current values in some sense (e.g., in terms of their overall distribution or statistical properties). To mitigate this issue, it can be used a more complex generative model, such as a Generative Adversarial Network" (GAN), specifically designed to generate high-quality, realistic samples.

A variant of the standard VAE architecture called Graph VAE or Graph Autoencoder (GAE) can adapt VAEs for graph node regression tasks. GAEs extend the standard VAE architecture to work with graph-structured data by encoding the graph structure and node features into the latent space and decoding the latent representation back into the original graph structure and node features [26].

The high-level overview of how it can adapt VAEs for graph node regression stays in the definition of an encoder network that takes as input the graph structure and node features, outputs the mean and standard deviation of a Gaussian distribution in the latent space, and then samples a point from the Gaussian distribution using the reparameterization trick [25]. After defining a decoder network that takes the sampled point from the latent space as input and outputs a predicted graph structure and node features, the whole architecture is set and prepared to create synthetic data.

By combining the VAE and the trained regressor, the VAEs might have leveraged the ability to generate synthetic data and the regressor's capability to fine-tune and refine the interpolated samples [35].

3.10 Loss Functions for Neural Networks

The evaluation metric is inevitably the primary aspect to focus on when training an artificial neural network, and for GNN, the following rule is also applicable. Loss function \mathcal{L} is used to optimize the hyperparameters of a framework with the objective of zero gradients.

The task of preparation of a DT of a mechanical system has the characteristic of a multiple input-output regression task on nodes of the graph. Since the graph \mathcal{G} can vary from a very small to an enormous number of nodes, the right strategy might be where a slight variation of errors on the targeted nodes is expected.

For example, the importance of selecting a metric for a model with optimal accuracy in a task where the primary task is to classify images from the Internet (e.g., MNIST) is probably not as crucial as in the DT of a crane that imitates a physical asset in operation with expensive goods in docks, where the main purpose is to operate such a crane with a high expectation of safety standards.

The first cohesive way anyone would probably start with is preferably standard metrics suitable for regressions tasks collected from [67; 62; 63]:

- Mean Absolute Error (MAE):

$$\mathcal{L}_{\text{MAE}}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N \text{abs}(\hat{y}_i - y_i)^2$$
- Root Mean Squared Error (RMSE):

$$\mathcal{L}_{\text{RMS}}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N \sqrt{(\hat{y}_i - y_i)^2}$$
- Mean Absolute Percentage Error (MAPE):

$$\mathcal{L}_{\text{MAPE}}(y, \hat{y}) = \frac{100\%}{N} \sum_{i=1}^N \text{abs}\left(\frac{y_i - \hat{y}_i}{y_i}\right)$$
- R-Squared (R^2):

$$\mathcal{L}_{R^2}(y, \hat{y}) = \frac{\sum_{i=1}^N (\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^N (\bar{y}_i - \hat{y}_i)^2}$$
- Mean Squared Error (MSE):

$$\mathcal{L}_{\text{MSE}}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2$$
- Mean Absolute Percentage Error (MAPE):

$$\mathcal{L}_{\text{MAPE}}(y, \hat{y}) = \frac{100\%}{N} \sum_{i=0}^{N-1} \frac{|y_i - \hat{y}_i|}{|y_i|}$$
- Mean Squared Logarithmic Error (MSLE):

$$\mathcal{L}_{\text{MSLE}}(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2$$

Moreover, a custom metric should be reconsidered when contemplating whether a standard metric is appropriate for a DT. One of the most relevant scenarios is that the critical variable, for example, mechanical stress, as in the example of the crane mentioned above, is most likely to be considered, and to obtain more conservative predictions, the overall metric should take this aspect into account. Therefore, the following suggested metric considering maximal error and loss \mathcal{L} function is then:

- Root Max Square Error (RMAXSE)

$$\mathcal{L}_{RMAXSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N \sqrt{\max(\hat{y}_i - y_i)^2} \quad (3.33)$$

■ 3.10.1 Optimizers

Since the main loss function and metric might be well set, the following step is to recognize a suitable optimizer. The most commonly used optimizers for training GNNs are:

- Stochastic Gradient Descent (SGD): This is a famous optimizer for training GNNs. It updates the weights of the network in the opposite direction of the gradient of the loss function concerning the weights. SGD can suffer from local minima, slow convergence, and other issues.
- Adam: This optimizer is an extension of SGD that uses adaptive learning rates. It adjusts the learning rate for each weight based on the history of gradients for that weight. Adam is known for its fast convergence and good performance on various tasks.
- Adagrad: Adagrad adapts the learning rate of each weight based on the historical gradient of that weight. It performs well on sparse datasets and is often used in natural language processing tasks.
- Adadelta: This optimizer is similar to Adagrad but uses a moving window of gradients to adapt the learning rate. Adadelta can handle sparse datasets well and is less sensitive to the initial learning rate.
- RMSprop: This optimizer also adapts the learning rate based on the historical gradients but uses a moving average of the squared gradients instead of the sum. RMSprop can handle non-stationary objectives and is often used in deep learning.

These are just a few of the most commonly used optimizers for training GNNs. More can be found at [63; 62]. The choice of optimizer can depend on the specific task and the dataset being used.

■ 3.11 Evaluation of Regressor used as DT

All of the mentioned sections neatly defined key variational features to explore mosaicking the workflow to establish the DT of the particular mechanical system. Without hesitation, validating the DD model is one of the inevitable aspects and probably the most crucial one, which cannot be easily excluded without consequences on the behaviour of targeted DT.

■ 3.11.1 Ground Truth Validation

The central emphasis of Ground Truth (GT) evaluation of DT is the paramount importance of visualization for operators utilizing the DT in the decision-making process. By enabling operators to visually observe the ground truth representation of the trained regressor on the physical-based dataset, they can clearly understand the actual behaviour of the simulated structure. This visual insight becomes crucial for informed decision-making processes, as operators can confidently rely on accurate visualization to make critical assessments and take appropriate actions.

■ 3.11.2 Node Error

Evaluating the targets of a dataset composed of nodes from a FEM is crucial for obtaining a comprehensive understanding of the model's weaknesses. By scrutinizing the target values associated with each node, areas where the model may exhibit shortcomings or inaccuracies in capturing the system's actual behaviour can be identified. This evaluation allows us to gain insights into specific regions or components of the structure that may require further refinement or validation. Understanding the model's weaknesses empowers us to make informed decisions and focus our efforts on improving its performance in critical areas, ultimately enhancing the overall accuracy and reliability of the model.

Furthermore, evaluating target values offers a deeper understanding of the model's limitations and areas where uncertainties may arise. It enables us

to assess the model's sensitivity to various input parameters and boundary conditions. By observing deviations between the predicted targets and the actual values from the finite element model, we can identify discrepancies and potential sources of error. This awareness is essential for model refinement and improvement, as it directs our attention to specific aspects that require attention and further investigation.

Additionally, evaluating target values facilitates effective communication and collaboration between model creators and stakeholders of a physical asset. By visualizing the discrepancies and weaknesses in the model, decision-makers can grasp the limitations and potential risks associated with the digital twin. This shared understanding enables informed discussions and aids in making optimal decisions that consider the strengths and weaknesses of the model. Ultimately, by evaluating the targets of the dataset, we gain valuable insight into the performance of the regressor as DT

3.12 Validation of Regressor

When developing a DT based on GNNs, one of the crucial factors to consider is the validation of the regressors. Validation techniques such as quantile-quantile plots and Tukey-Anscombe diagrams can be employed as possible policies or strategies to assess the performance and accuracy of the regressors.

The quantile-quantile plot provides a graphical representation of the distribution of the predicted values compared to the expected values. Plotting the predicted values' quantiles against the expected values' quantiles, deviations from a straight line can indicate discrepancies or errors in the regressor's predictions. This technique enables a visual assessment of the model's performance and its ability to capture the underlying patterns in the data accurately.

The Tukey-Anscombe diagram is another valuable tool for validating regressors in a DT based on GNNs. It involves plotting the residuals, the differences between the predicted and actual values, against the expected values. This plot helps identify any systematic patterns or trends in the residuals, indicating potential biases or shortcomings in the model. By visually examining the scatter of the residuals, one can gain insights into the regressor's performance and reliability.

These validation techniques, namely the quantile-quantile plot and the

Tukey-Anscombe diagram, provide valuable means to assess the accuracy and quality of the regressors in the DT. By employing these policies or strategies during the development and deployment stages, one can ensure that the DT effectively represents the behaviour of the actual system.

■ 3.12.1 Homoskedacity and Heteroskedacity

Homoskedasticity is one of the critical assumptions under which the Ordinary Least Squares (OLS) gives an unbiased estimator, and the Gauss–Markov Theorem applies. Linear regression modelling typically tries to explain the occurrences with a single equation.

To interpret heteroskedasticity [36] can be used as a scatter plot of the residuals against the predicted values or the independent variable(s). Suppose the scatter plot shows a funnel shape, with the spread of the residuals increasing or decreasing as the predicted values or independent variable(s) increase. In that case, heteroskedasticity is likely present.

Another way to detect heteroskedasticity is to perform a formal test, such as the Breusch-Pagan [28] test or the White test [27]. These tests examine the relationship between the squared residuals and the independent variables to determine whether there is evidence of heteroskedasticity.

Heteroskedasticity can have essential implications for regression analysis. If it is present, the standard errors of the regression coefficients will be biased, which can lead to incorrect conclusions about the statistical significance of the coefficients. Additionally, heteroskedasticity can reduce the efficiency of the regression estimates, making it more difficult to detect significant effects of the independent variables on the dependent variable.

Techniques such as weighted least squares, robust standard errors, or generalized least squares regression may be used to address heteroskedasticity. These techniques account for heteroskedasticity by adjusting the standard errors of the regression coefficients, which can improve the accuracy of the regression estimates.

3.12.2 Non-linear Heteroskedasticity

If the scatter plot of the residuals against the predicted values shows a logarithmic shape, it suggests the presence of non-linear heteroskedasticity. This means that the residuals' variability is not constant across the range of values of the independent variable(s), and it increases (or decreases) non-linearly.

In this case, transforming the data to reduce the non-linear heteroskedasticity might be considered. One common approach is to apply a logarithmic transformation to the dependent variable, which can help stabilize the residuals' variance. Another method is transforming the independent variable(s) to reduce the non-linear heteroskedasticity.

However, it's important to note that non-linear heteroskedasticity can also be caused by misspecification of the functional form of the regression model [68], so it's essential to carefully examine the model specification and consider applicable alternative forms or nonlinear-models. Suppose the non-linear heteroskedasticity persists even after transforming the data or modifying the model. In that case, there is a need to use techniques such as generalized least squares regression or weighted least squares regression, which are designed to handle non-linear heteroskedasticity. These techniques can adjust the standard errors of the regression coefficients to account for the non-constant variance of the residuals, which can improve the accuracy of the regression estimates.

Non-linear heteroskedasticity in a multi-input, multi-output regression model can be interpreted in several ways depending on the specifics of the model and the data.

One possible interpretation is that the variance of the errors varies non-linearly with the values of one or more of the independent variables, indicating that the model is misspecified or that there are interactions between the independent variables that have yet to be accounted for. In this case, it may be necessary to re-specify the model or to include additional variables or interaction terms to account for the non-linear heteroskedasticity.

Another possible interpretation is that outliers or influential observations are driving the non-linear heteroskedasticity. Outliers can substantially affect the estimated regression coefficients and increase the variability of the residuals, which can lead to non-linear heteroskedasticity. In this case, it may be necessary to identify, remove, or downweight the influential observations

to improve the model fit and reduce the non-linear heteroskedasticity.

Finally, non-linear heteroskedasticity can also be caused by the presence of unobserved variables or unmeasured heterogeneity in the data. In this case, it may be necessary to use advanced statistical techniques, such as instrumental variables regression or panel data models, may be required to account for the unobserved variables and reduce the non-linear heteroskedasticity.

The interpretation of non-linear heteroskedasticity in a multi-input, multi-output regression model will depend on the specific details of the modelled structure and the data. It will require careful analysis and consideration of alternative specifications and modelling techniques but should mainly be avoided for DT.

3.13 Methodology Definition

In this section, a proposed methodology is presented, as illustrated abstractly in Fig.3.8 at section 3.9, for the invention of a DT based on a PM of a mechanical structure. The methodology involves leveraging principles from Poisson and Hook's laws to establish a foundation for DT construction. A systematic approach is outlined through this workflow, integrating physical modelling principles with digital techniques to develop a comprehensive representation of the mechanical system.

1. Identify physical phenomena to model and monitor actual observed products and collect and process data. Gather all the data related to the physical structure, including its dimensions, material properties, and any other relevant parameters.
2. Define geometrical model Γ to required spatial precision and build a virtual model using computer-aided design (CAD) software to create a virtual model of the mechanical structure. This model should be as accurate as necessary for the physical structure.
3. Creation of FEM Ω based on requirements of the first and second steps. Then, simulate the structure's mechanical behaviour. This can help identify potential issues with the design and ensure that the virtual model accurately represents the physical phenomenon of the observed structure.
4. Extract calculated physical attributes \mathcal{D} of converged model. These features could include stress, strain, displacement, and other mechanical properties.
5. Train and Validate Graph Regressor $f(\mathcal{G})$ by using the extracted features and training a GNN to learn the relationship between the features and the mechanical behaviour of the structure. Test the trained GNN on new data and evaluate its accuracy in predicting the structure's mechanical behaviour using tools.
6. Replacement of FEM by DT based on regressor of GNN once it has been successfully validated. Deploy the DT to monitor and predict the behaviour of the physical structure in real-time.
7. Optimize the Graph by reducing the DT complexity and continuously updating the GNN architecture. As new data becomes available, continuously update the DT to improve its accuracy and predictive capabilities. This could involve retraining the GNN using the new data or adjusting the monitoring equipment to collect more relevant data.

Part II

Experimental Demonstration



Chapter 4

Demonstration by Experiments

The dedicated part of the thesis aims to demonstrate the usage of distributed knowledge from a PM suitable for a digital twinning task.

One primary task is to have representative performers acquire important responses to see the state of the mechanical systems, for instance. A minor approach would be to enhance knowledge of DP by meta-knowledge of DI to achieve optimal design.

GNNs have emerged as powerful tools for modelling and analyzing complex systems. By representing data as graphs, GNNs can capture relationships and dependencies between entities, making them well-suited for various applications, including regression and classification tasks, as comprehensively emphasized in section 3.6. In this chapter, we present two case studies that showcase GNNs' potential in the context of finite element modelling.

The first case study focuses on benchmarking GNNs for regression on nodes using the GFdataset [69], the graph mesh-based dataset from FEMs. The exploration of the challenges of working with complex and heterogeneous data, such as the variability in the number of nodes and edges in each model and the need for feature engineering, is drawn. It discusses the design and implementation of the GNN architecture and the evaluation of its performance on various metrics. Through this case study, it is depicted as a demonstration of the effectiveness of GNNs in modelling complex and non-linear relationships in finite element models.



Chapter 5

Methods



5.1 Methods for Knowledge Distillation Experiments

The experiment design is accompanied by specific methods to gain a proof of concept for the ideas proposed in this thesis. These methods are carefully chosen to ensure the validity and reliability of the experimental results. Implementing these methods aims to demonstrate the feasibility and effectiveness of the suggested concepts in a practical setting. The experiment design and accompanying methods serve as crucial components in validating the proposed ideas and providing empirical evidence to support the claims put forth in this thesis.



5.1.1 Extraction of Graphs

The graphs are extracted for all future physical models by specifically choosing the full element nodes with single-direction connectivity. For more detailed information on the visualization process (see section 3.3).



5.1.2 Frameworks of Baseline of DD Models

First, the critical framework of multi-input-output linear regression (MLR) is utilized. This statistical modelling technique establishes a linear relationship

between multiple input variables and multiple output variables. By leveraging multi-input-output linear regression, the aim is to predict multiple output variables simultaneously based on a set of input variables. This framework offers interpretability and enables the estimation of coefficients representing the influence of each input variable on its corresponding output variable.

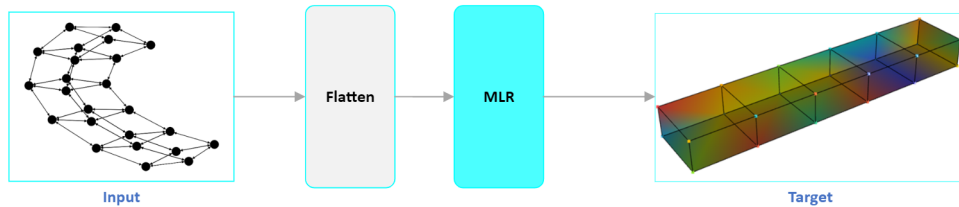


Figure 5.1: framework MLR: Multi-linear input/output regression

Secondly, feed-forward neural networks, a fundamental type of artificial neural network, are incorporated as a critical framework (FN). These networks are known for their ability to learn complex patterns and relationships in data, making them widely used in various fields, including image recognition, natural language processing, and regression tasks. By including feed-forward neural networks, the researcher aims to harness their powerful learning capabilities and assess their effectiveness in predicting the behaviour of mechanical structures within the proposed hybrid modelling approach.

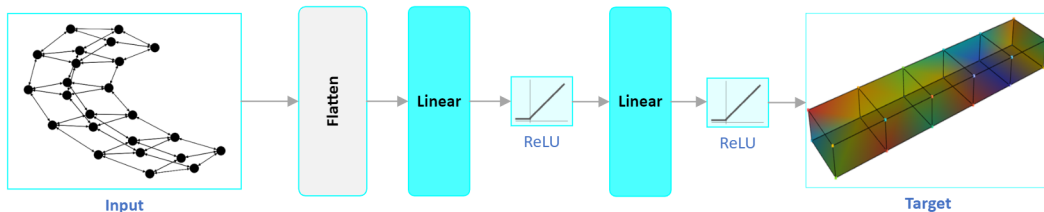


Figure 5.2: framework FN: Feed-forward neural network with two hidden layers

As illustrated in the preceding Figs. 5.1 and 5.2, it is evident that all frameworks, except for the graph-based ones, employ an initial operation of *Flatten* to streamline the data in preparation for training meticulously. This process involves isolating individual nodes within the graphs, each with their respective input and output attributes, thereby facilitating efficient training

Frameworks based on GNNs and variations on chosen layers are employed. GNNs are specifically designed to handle data with graph structures, making them well-suited for modelling interconnected components in mechanical systems. By incorporating GNNs into the framework, the ability to capture complex relationships and dependencies between nodes in the graph representation of the structure is established. Additionally, exploring different layer variations enhances the representation and information propagation within the GNN, contributing to more accurate and effective modelling of mechanical systems. The following framework (GCN) depicted at Fig.5.3 is composed solely of graph convolutional layers.

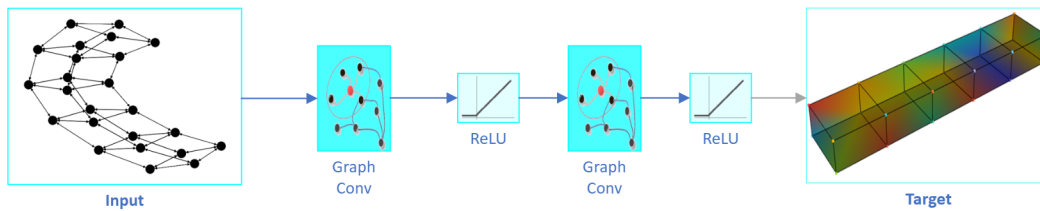


Figure 5.3: framework GCN: two graph convolutional layers

Finally, the hybrid modelling approach introduces SAGE layers as a standalone framework illustrated by Fig.5.4. SAGE layers, or GraphSAGE, are GNN layers that aggregate information from neighbouring nodes in a graph structure. By incorporating sage layers, it is possible to capture complex relationships and dependencies between nodes in the graph representation of the mechanical structure. Sage layers provide a flexible and effective approach for modelling graph.

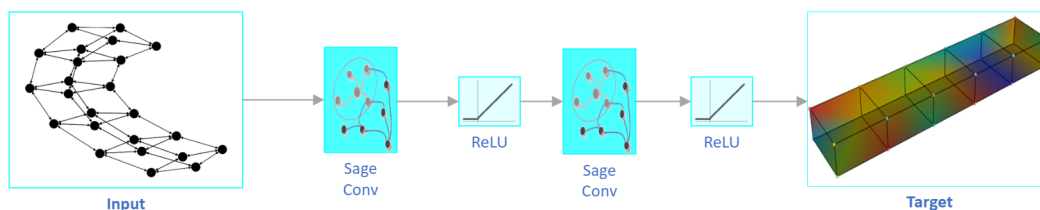


Figure 5.4: Framework SAGE: two graph SAGE layers

■ 5.1.3 Scaling of a Dataset

Due to the potential dangers and critical implications associated with scaling or transforming the dataset [73], further scaling is not considered in this study. While it is not a conventional approach for data normalization, such as MinMax or Standardization, a scaling method is only applied to the input variables (specifically, deformations written to each node), while the outputs remain unscaled. Consequently, the loss function may exhibit high initial values due to the varying orders of mechanical parameters with different units (e.g., mm, MPa, g). This decision acknowledges the potential consequences of an unbalanced unit.

■ 5.1.4 Data Splitting

As the dataset used in this study does not exhibit any time dependency, it allows for the straightforward random shuffling of the entire dataset and subsequent division into standard training and validation subsets. The process involves randomly rearranging the data points to eliminate any inherent order or sequence. By fractionally dividing the shuffled dataset, a portion is allocated for training purposes while another amount is set aside for validation. This standard splitting approach ensures a representative data distribution and facilitates practical evaluation and validation of the models used in the experiment.

■ 5.1.5 Loss Functions and Optimizer

The loss function used in this study is the root max square error \mathcal{L}_{RMASE} , which calculates the maximum magnitude of errors between the predicted and targeted values on the nodes of the structure, as discussed in the previous chapter. This loss function provides a robust measure of the most significant discrepancies between the predicted and desired values, allowing for a more comprehensive evaluation of the model's performance. The classical ADAM optimizer is also chosen as the optimization algorithm for this loss function. ADAM adapts the learning rate based on the gradients of the model parameters, ensuring efficient and effective optimization during the training process. The study aims to enhance the accuracy and convergence of the hybrid modelling approach for mechanical structures by utilising the root max square error loss function and the ADAM optimiser.

5.2 Graph Reduction

In Section 3.5, various methods for graph reduction were meticulously identified. Additionally, Algorithm 4 has been specifically designed and will be utilized as a pivotal component in our experimentation process, as further elaborated. These methods are deemed crucial in streamlining the complexity of graph-based models, enabling more efficient processing and analysis. By implementing these reduction techniques, efforts are made to enhance the performance and scalability of our models, ultimately contributing to advancements in graph-based data analysis and modelling.

5.3 GF dataset: Mesh-based Graph Dataset

Many data sets for supervised learning have been created throughout the machine learning era, and without a doubt, many others are in the evolution process. Graph neural network, the youngest artificial neural network technique, is also characterised by a need to have and see specific patterns that a model with a particular architecture might learn. The following lines describe the dataset consisting of graphs with nodes of information created based on simple FEM structural models (GF dataset) designed to explore the possibilities of a specific hybrid modelling technique suggested as a bridge connecting a FEM and a GNN into an effective tool for the purpose to creating a Digital Twin of a mechanical system. The GF Dataset visualised in Fig.5.5 was introduced at [69].

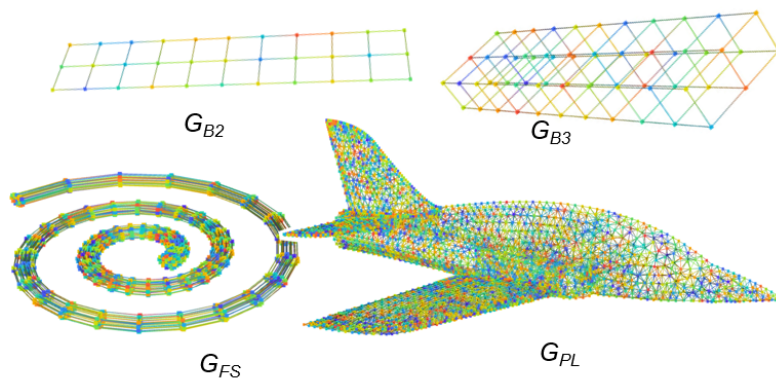


Figure 5.5: GFdataset: Graphs of structures extracted from FE models \mathcal{D} [69]

These compiled information of FEMs depicted at Fig.5.6 possess a distinct feature of precise geometry, with meticulously crafted meshes and specific boundary conditions. This precision allows for explicitly exploring feasible

strategies to build a data-driven model. To facilitate replicating a given geometrical structure, the repository includes the input file required to run or enhance the FE model. The repository aims to support reproducibility and further advancements in data-driven modelling for mechanical structures by providing access to these resources.

- \mathcal{D}_{b2} ... Beam2D: Encastred and loaded 2D beam.
- \mathcal{D}_{b3} ... Beam3D: analogy of Beam2D at 3D.
- \mathcal{D}_{fs} ... Fibonacci's Spring: analogy of 3D beam with introducing slightly complicated geometry.
- \mathcal{D}_{pl} ... Plane: Symmetrically cut geometry of RC plane loaded with pressure on the wing.

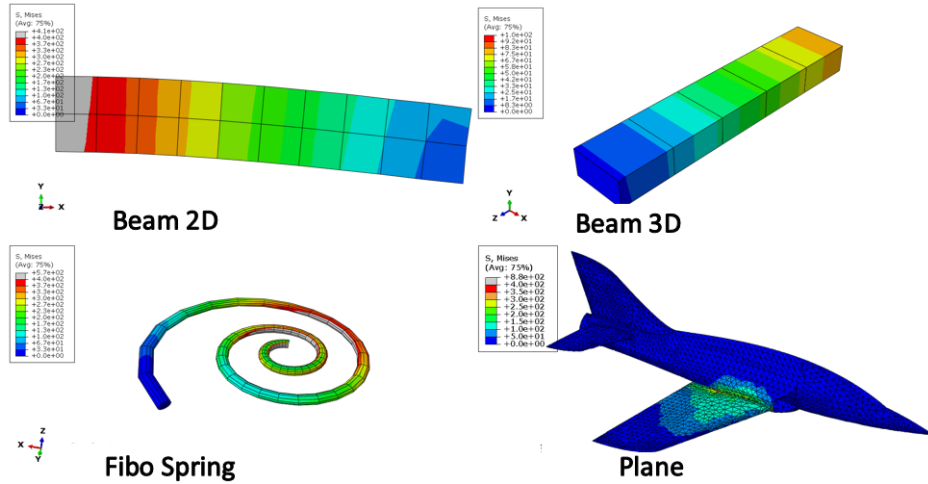


Figure 5.6: GF dataset: Visualized each FE model from \mathcal{D} in their final converged state

The main goal of introducing the GF dataset is to demonstrate tasks where the primary purpose is mainly to train various GNN architectures (not only GNN but other strategies to train ML models are possible) based on a sufficient amount of extracted data from a specific FE model mimicking structural mechanic problem. The collection of data subsets consists of the five compilations of generated data, gradually tackling complexity from the perspective of their geometrical structures.

The graph structures available via the GF dataset store information regarding mechanical variables on each node of the specified structure. The prepared data carrying the inputs and targets can be processed via GNN. The FE models created for this purpose are described in the following subsection.

The selected examples of physically driven models will not extol the virtues of finite element modelling practices. However, the intention is to show simple geometrical structures acting in a straightforward quasi-static environment to easily distinguish whether the chosen strategy of creating a model driven by extracted data has unique potential.

The presented dataset will be evaluated to create an initial benchmark. The experiment set will consist of four basic frameworks to emphasise a brief understanding of specific techniques' limitations. As mentioned earlier, MLR, FN architecture, and two different architectures of GNN are available frameworks competing in benchmark experiments for GF datasets.

5.4 Use Case: Beam 2D

A seemingly simple 2D Beam example will describe the sequence for building a specific data set. Finally, the following procedure in conjunction with Fig.5.7 is applied for the lately presented sub-datasets of the GF dataset group.

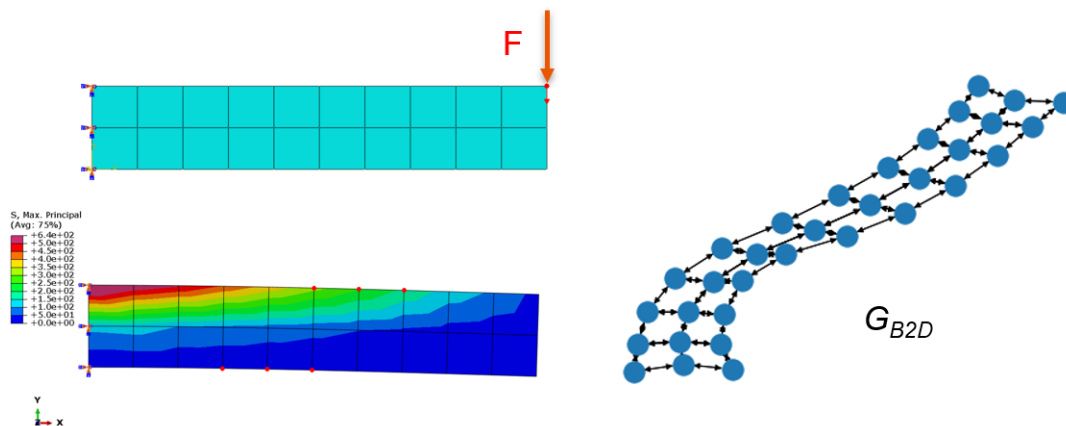


Figure 5.7: The workflow presented on Beam2D example: upper left: FE discretisation with the highlight of boundary condition and load applied; bottom left: Maximal Principal distribution at last frame; right: drawn bi-directional heterogeneous graph created based on FE model mesh.

"Hello Beam", as the commonly known structural mechanics example, is loaded and constrained on opposite sides in a usual manner. The left side of the Beam has nodes fixed, and on the opposite side, the concentrated force F is applied in the vertical direction heading down. The building sequence of the dataset procedure consists of conventional steps in FE modelling and additional data preparation of the data to embed the information gained from the simulation into the graph. The sequence is relatively smooth:

- Define geometry, mesh, material properties
- Identify boundary and load conditions
- Run and post-process analysis in a manner to create the basis for graph structure
- Compile FE simulation model information into the mesh-based graph and store data on nodes

■ 5.4.1 Geometry Material and Mesh

The two-dimensional beam structure used in this use case, Beam2D (B2), is constructed entirely from steel. The structure's geometry is partitioned into quadrilateral elements, which allows for a more detailed and accurate representation of the Beam's shape and behaviour. The FE models can effectively capture the steel beam's structural response and deformation characteristics by utilising quadrilateral elements.

$$\Omega_{B2}(E, \Gamma(e), E, \nu) \tag{5.1}$$

■ 5.4.2 External Load and Boundary Conditions

The structure is subjected to a concentrated force applied on one side, which linearly increases throughout the simulation. This loading condition allows for examining Beam's response to gradually intensifying forces. On the opposite side, the entire structure is fully encastred, meaning that all degrees of freedom of the side nodes are constrained. This constraint prevents displacement or rotation at the fixed end, providing a fixed boundary condition for the Beam. By imposing these loading and constraint conditions, the FE models can accurately simulate the behaviour of the Beam under realistic scenarios, enabling a comprehensive analysis of its structural response and deformation characteristics.

■ 5.4.3 Data acquired for Data-Driven Models

- Inputs: Reaction forces, represented as F , are measured at the constrained nodes using hypothetical force sensors placed at those specific locations. These sensors allow direct measurement of the forces exerted on the constrained nodes, providing valuable insights into the structural response and stability under the given loading conditions.

- **Targets:** The targets in this study refer to the values of maximal principal stress obtained at each node of the structure during the data acquisition process. These targets serve as valuable data points for understanding the stress distribution across the structure and can provide insights into areas of potential high-stress concentration.

5.5 Use Case: Beam 3D

The 3D beam in this context is an analogy to the previous example, where the structure is still represented by a beam but in a three-dimensional space. Like the previous example, the 3D beam is made of steel material and partitioned into hexahedral elements to accurately capture its geometry and behaviour. The loading conditions and constraints imposed on the beam are adjusted accordingly to reflect the three-dimensional nature of the structure. By utilizing this analogy, the research aims to extend the understanding gained from the 2D beam example to three-dimensional mechanical structures, allowing for a more comprehensive analysis and exploration of data-driven modelling techniques.

The suggested fundamental 3D model of the beam depicted in Fig.5.8 has only one goal: to be initially and quickly validated during the development of higher-order geometrical typologies, which are usually essential for digital twinning. The Beam3D is defined by the geometry Γ of 10x20 mm and by the steel material property in the elastic area. Constraining (fixed face on one side) and loading (force application on the opposite end of the beam, $F = 100N$) were applied identically to the previous example.

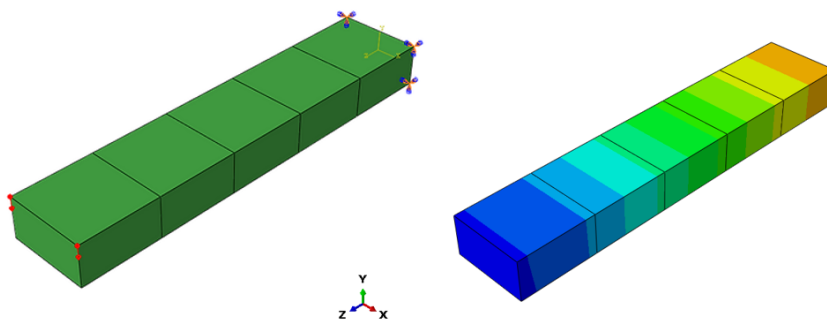


Figure 5.8: Beam3D: On the left is FE discretization, the right demonstrates the distribution of Maximal Principal stress at the final frame of simulation

5.6 use case: Fibonacci's spring

Moreover, the presented example involves a slightly more complex 3D topology inspired by the Fibonacci sequence visualised in Fig.5.9. This topology is applied with a small contribution to mesh discretisation, allowing for a more detailed representation of the structure. Specifically, the example focuses on the geometry of a torsional spring, which is constrained on the inner diameter and loaded on the opposite side. Further information regarding the geometry and preprocessing of this graph based on the finite element mesh can be found in the article dedicated to this topic. This article delves into how the graph representation is derived from the finite element mesh, providing valuable insights into the preprocessing steps involved in the data-driven modelling approach [?].

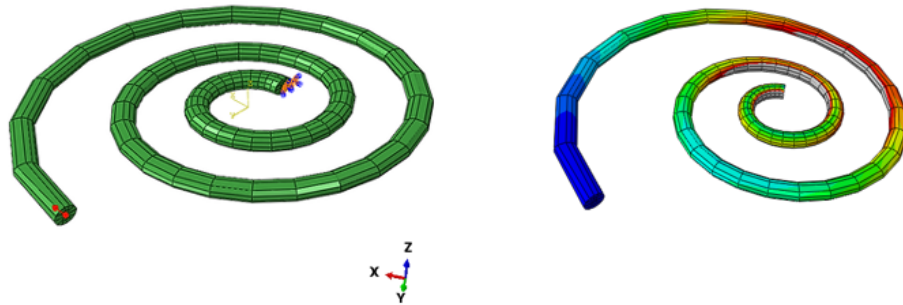


Figure 5.9: Fibonacci Spring: FE discretisation and Maximal Principal distribution at last frame

In the presented example, the inner end of the torsional spring is encastred, meaning it is fully constrained and unable to undergo any displacement or rotation. On the other hand, the opposite end of the spring is loaded similarly to the previous example, with a concentrated force applied at a specific centre node on the outer surface. This loading condition allows for the analysis of the torsional behaviour of the spring and the examination of its response to the applied force. By imposing these boundary conditions, the finite element model accurately simulates the mechanical behaviour of the spring, enabling a comprehensive study of its torsional characteristics and providing insights into its performance under various loading conditions.

5.7 Use case: Half Airplane - Symmetrically Cut

Lastly, the Plane dataset is an example, motivating further enhancements in real-world applications. This dataset showcases a half 3D geometry per Fig.5.10 representing the entire volume plane. It is important to note that the FEM employed in this example may not be in the form typically expected for complex and highly accurate aeroplane specifications. However, this example is well-suited for handling large graphs. By utilizing this dataset, researchers and practitioners can explore and develop optimal use-case applications, leveraging the advantages of graph-based modelling techniques. The presented example serves as a starting point for future advancements and improvements in graph-based modelling and analysis of complex structures.

The plane wing is loaded by a pressure applied from the opposite direction of gravity, while the influence of gravity itself is also considered. This combined loading scenario creates a complex and realistic loading condition that considers both the applied pressure and the effects of gravity. By studying the plane's response under this combined loading, valuable insights can be gained into its structural behaviour, including the interaction between the applied pressure and the gravitational forces. This use case provides a practical example for investigating the performance of structures subjected to multiple loading factors, which is crucial for designing robust and efficient structures in various engineering applications.

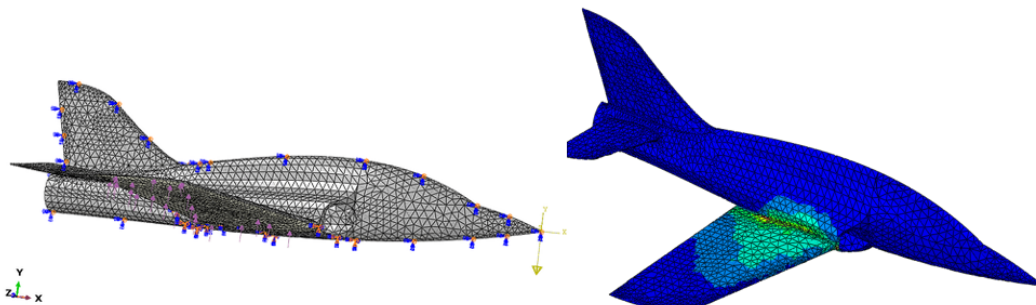


Figure 5.10: Plane: FE discretization with boundary conditions and Maximal Principal distribution at last frame

The main objective of this study is to measure the reaction forces at the end of the wing, which provides valuable information about the structural response to the applied loading. Additionally, a comprehensive stress map of the plane is desired, which will help accurately represent the distribution and magnitude of stresses throughout the structure. A more complete understanding of the structural behaviour can be achieved by capturing these reaction forces and obtaining a detailed maximal principal stress map. These measurements and stress analyses are crucial for validating the computational models' accuracy and improving the plane's design and performance.

5.8 Summary of GF Dataset

The presented use cases of physical models, aimed at investigating the behaviour of mechanical structures, are comprehensively summarized in a dedicated table. This summary table provides a concise overview of each use case, encompassing key aspects essential for further modelling and analysis. It includes pertinent information such as the specific geometry, loading conditions, objectives, and significant findings. The table is a valuable resource for researchers and practitioners studying similar mechanical systems by encapsulating these crucial details in a summarised format. It facilitates easy comparison and identification of commonalities or differences among the use cases, promoting a deeper understanding of structural behaviour and guiding future modelling endeavours. The characteristics captured in the table consist of the type of finite elements used to discretize the geometry (*Typeel.*), the number of nodes within the dataset (*No.Ns*), the number of elements (*No.El*), the maximal applied force on the structure ($\max(F)$), and the number of simulation frames (*Sim.Frames*).

FE mod-els char.	\mathcal{D}_{b2}	\mathcal{D}_{b3}	\mathcal{D}_{fs}	\mathcal{D}_{pl}
Type El.	Quad	Quad	Quad	Tet.
No. Ns	33	24	1524	4758
No. El.	12	8	321	5983
$\max(F)$	100 N	200 N	0.5 N	10 MPa
Sim. frames	500	500	200	200

Table 5.1: FE models summary by its statistics of structure features

In addition to the use case summary table, a separate table is also established to present the extracted graphs from the physical models. This table provides a comprehensive overview of the graph structures obtained from each model, including information such as the number of nodes, edges, and connectivity patterns. It is a valuable reference for researchers interested in graph-based modelling and analysis, allowing for easy comparison and selection of appropriate graph structures for specific applications. This table facilitates a deeper understanding of the data representation by capturing the essential characteristics of the extracted graphs. It aids in developing graph neural network models for further analysis and prediction tasks.

Graphs char.	\mathcal{D}_{b2}	\mathcal{D}_{b3}	\mathcal{D}_{fs}	\mathcal{D}_{pl}
G. Ns.	33	24	1426	4758
G. Es.	104	88	7518	48210
Feat. X	S, U	S, U	S, U, Le	S, U, Le
Feat. y	S, U	S, U	S, U, Le	S, U, Le

Table 5.2: Graphs statistics



Chapter 6

Results of Baseline for DTs

This section introduces results on the methodology developed in Section 3.13 of this thesis. The designed experiments evaluate this methodology using essential tools identified during its development. The overall evaluation of results is conducted using the GF dataset, providing a comprehensive representation of the methodology's effectiveness.

Firstly, the ranking of each framework for regressors is determined based on their performance across the validation datasets of subexperiments. This is visualized using the most suitable identified tool, the boxplot. Additionally, a similar evaluation of training time is conducted to understand the computational demand required to develop certain DTs.

The subsequent evaluation focuses on validating ground truth in DT models, which serves as a cornerstone for understanding DT behaviour. Through explicit visual representations of the trained regressor's performance on physical-based datasets, operators can gain profound insights into the simulated structure's actual behaviour. The overall performance is further assessed through Model Diagnostic, utilizing a quantile-quantile plot and Tukey-Anscombe plot with unscaled residuals to examine the model's behaviour across the entire operational range.

The final analysis involves examining the training curve histories for all experiments to depict the spread across the best-converged models. By scrutinizing the training curves over iterations or epochs, insights into the convergence patterns and stability of the models can be gleaned. This comprehensive examination allows a thorough understanding of the training process and its impact on model performance.

The methodology finishes with graph reduction, serving as both the initial point and the recurring loop in the DT model lifecycle. Graph reduction is demonstrated through experiments designed similarly to previous ones but with variations introduced by reduction techniques outlined in Section 3.5, altering the graph structure of specific datasets.

6.1 Accuracy and Time training of Regressor

Within this subsection of the results section, model ranking based on boxplot analysis was conducted. Boxplots proved invaluable as a visualisation tool for comparing model performance across multiple experiments. By examining the distribution of the RMAXMSE metric defined in section 3.10, which offers insights into the spread of model predictions on the validation set, the box plots clearly illustrated performance variations among different sub-datasets in accordance with Fig.6.1. Summary Tab. 6.1 provides a compiled view of the overall experiment.

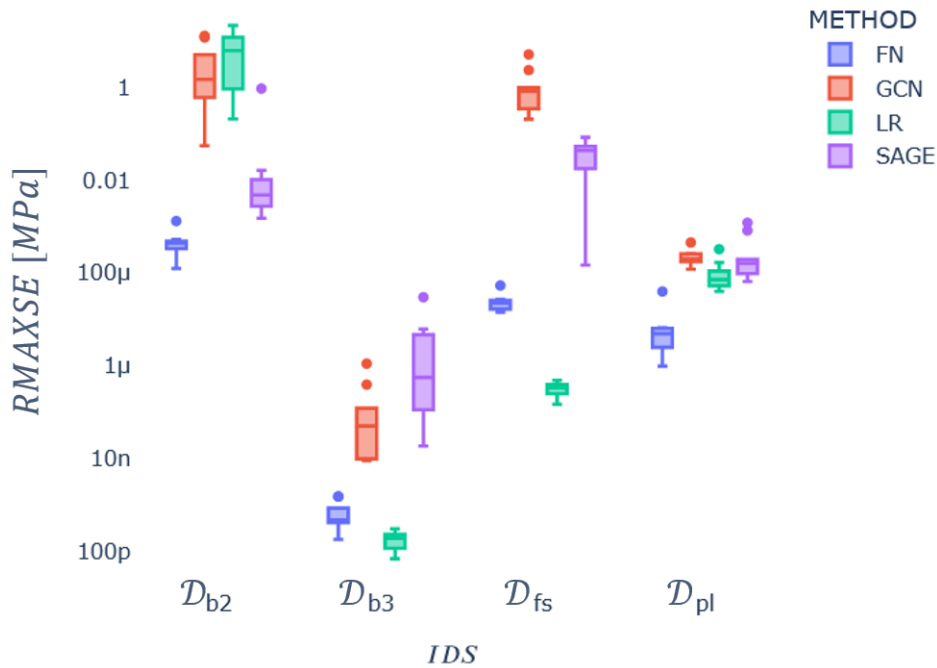


Figure 6.1: GFdataset: Boxplot of RMAXSE to depict accuracy of a regressor for DT implementation

Methods	\mathcal{D}_{b2}	\mathcal{D}_{b3}	\mathcal{D}_{fs}	\mathcal{D}_{pl}
LR	$8E+0 \pm 8E+0$	$2E-10 \pm 8E-11$	$3E-7 \pm 1E-7$	$1E-4 \pm 9E-5$
FN	$5E-4 \pm 3E-4$	$7E-10 \pm 5E-10$	$2E-5 \pm 1E-5$	$8E-6 \pm 1E-5$
GCN	$4E+0 \pm 5E+0$	$2E-7 \pm 3E-7$	$1E+0 \pm 2E+0$	$2E-4 \pm 1E-4$
SAGE	$1E-1 \pm 3E-1$	$5E-6 \pm 9E-6$	$4E-2 \pm 3E-2$	$3E-4 \pm 4E-4$

Table 6.1: Summary Table of RMAXSE on validation sets for GF dataset

Furthermore, similarly to the previous analysis, the training time required to develop a regressor from the set of frameworks was evaluated. Boxplots were again used as a valuable visualisation tool for comparing the training time across different frameworks per Fig.6.2. By examining the distribution of training times, insights were gained into each framework's efficiency in terms of computational resources required for model development. In addition

to the boxplot, a detailed examination of the resulting training ranges is provided by Tab. 6.2

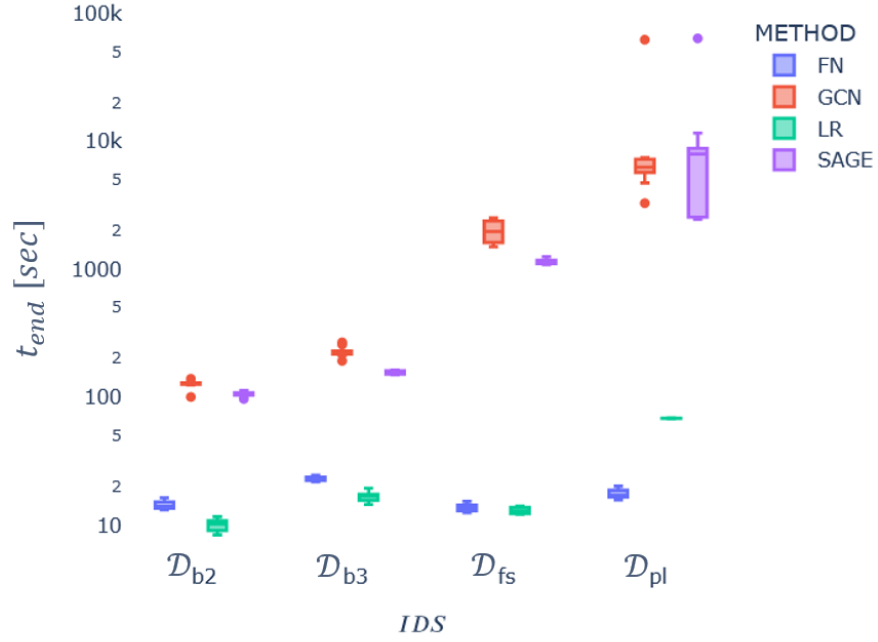


Figure 6.2: GFdataset: Boxplot of overall training time required to establish regressor for specific DT

Methods	\mathcal{D}_{b2}	\mathcal{D}_{b3}	\mathcal{D}_{fs}	\mathcal{D}_{pl}
LR	$1.0E1 \pm 1.1E0$	$1.7E1 \pm 1.5E0$	$1.3E1 \pm 7.4E-1$	$6.8E1 \pm 3.2E-1$
FN	$1.4E1 \pm 1.1E0$	$2.3E1 \pm 1.0E0$	$1.4E1 \pm 9.8E-1$	$1.8E1 \pm 1.5E0$
GCN	$1.3E2 \pm 1.0E1$	$2.3E2 \pm 2.2E1$	$2.0E3 \pm 4.1E2$	$1.2E4 \pm 1.8E4$
SAGE	$1.1E2 \pm 4.6E0$	$1.6E2 \pm 4.8E0$	$1.2E3 \pm 5.0E1$	$1.2E4 \pm 1.8E4$

Table 6.2: The training time required for an individual model in seconds

6.2 Ground Truth Evaluation of Digital Twins

The GF dataset's Ground Truth (GT) results are presented, where the GT and predicted values of graph regressors are visualised for all datasets. The paramount of evaluation is to visualise the regressor performance of prediction on a selected data point from a dataset referring to a particular load case of mechanical structure. The results provide insight into their efficacy since the regressors are potentially trained to be DT. Furthermore, by comparing the predicted values to the ground truth, the accuracy of each regressor can be assessed and determine which ones performed best. An additional minor aspect is to examine the variability in performance across the GF datasets, which sheds light on the generalizability of these models.

6.2.1 GT Evaluation: Beam 2D

The plot by Fig.6.3 illustrates the regressor MLR model's performance on the original geometry, displaying individual samples alongside their input, target output, model prediction, and associated error, characterised by $e \in (-0.6, 0.0) [MPa]$ for visualised data point.

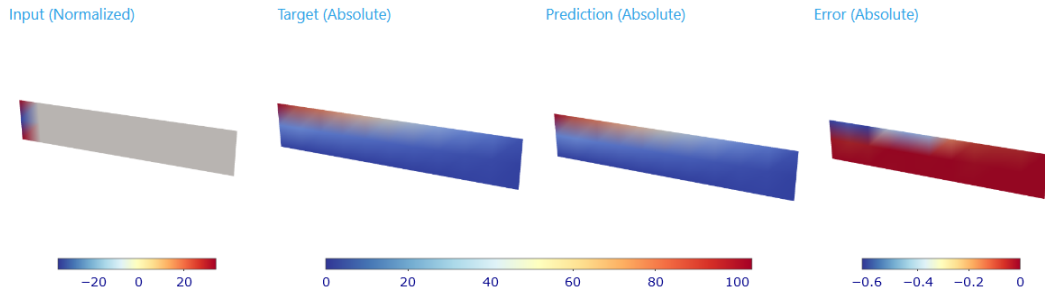


Figure 6.3: GT visualisation for the MLR framework of Beam2D DT for specific data sample

Visualising by Fig.6.4 the performance of the regressor FN model on the original geometric data, the plot showcases each sample along with its input, target output, model prediction, and error, denoted by $e \in (-0.6, 0.2) [MPa]$ for visualised sample.

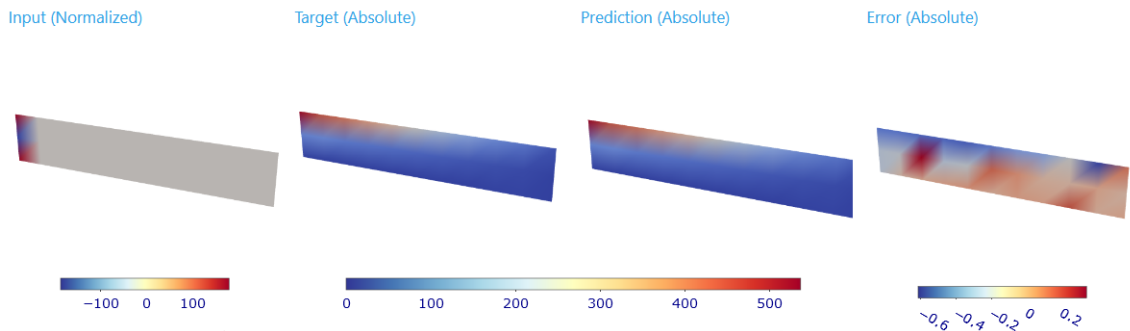


Figure 6.4: GT visualisation for the FN framework of Beam2D DT for specific data sample

Examining the regressor GCN model's performance on the original geometric dataset, the plot by Fig.6.5 presents individual samples alongside their input, target output, model prediction, and associated error, with $e \in (-0.15, 0.01) [MPa]$ for visualised sample.

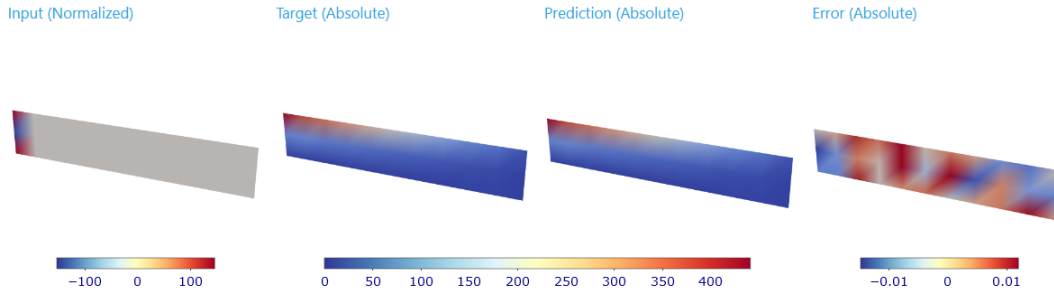


Figure 6.5: GT visualisation for the GCN framework of Beam2D DT for specific data sample

The plot provided by Fig.6.6 as a visual representation of the regressor SAGE model's performance on the original geometric data, depicting each sample's input, target output, model prediction, and error, where $e \in (-0.02, 0.05) [MPa]$ encapsulates the error range of data sample.

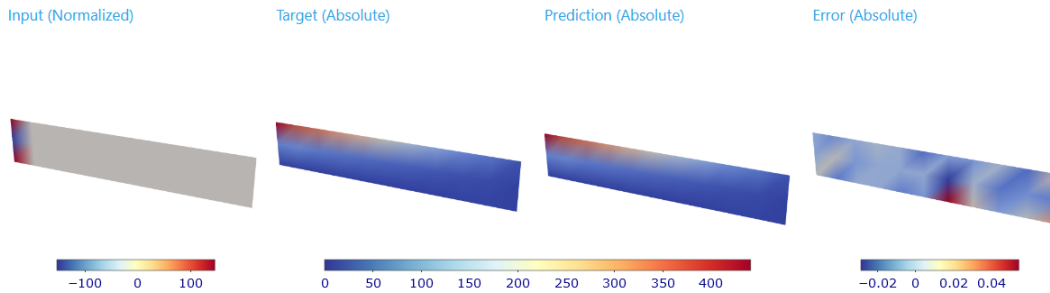


Figure 6.6: GT visualisation for the SAGE framework of Beam2D DT for specific data sample

6.2.2 GT Evaluation: Beam 3D

The plot by Fig.6.7 illustrates the regressor MLR model's performance on the original geometry, displaying individual samples alongside their input, target output, model prediction, and associated error, characterised by $e \in (-0.002, 0.002)$ [MPa] for visualised sample.

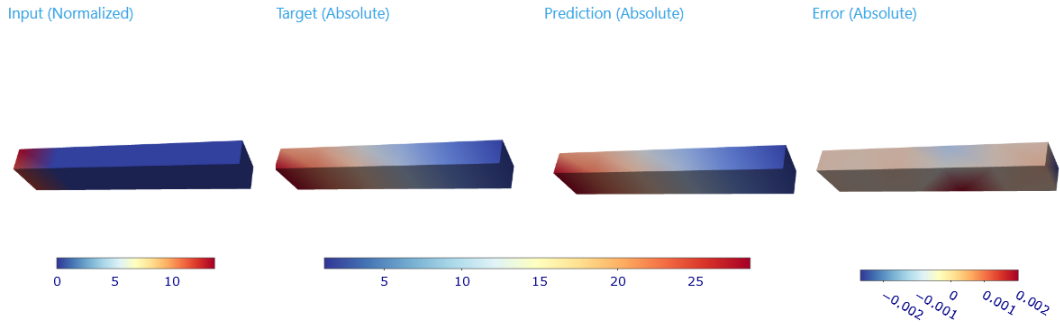


Figure 6.7: GT visualisation for the MLR framework of Beam3D DT for specific data sample

Visualising by Fig.6.8 the performance of the regressor FN model on the original geometric data, the plot showcases each sample along with its input, target output, model prediction, and error, denoted by $e \in (0., 0.0015)$ [MPa].

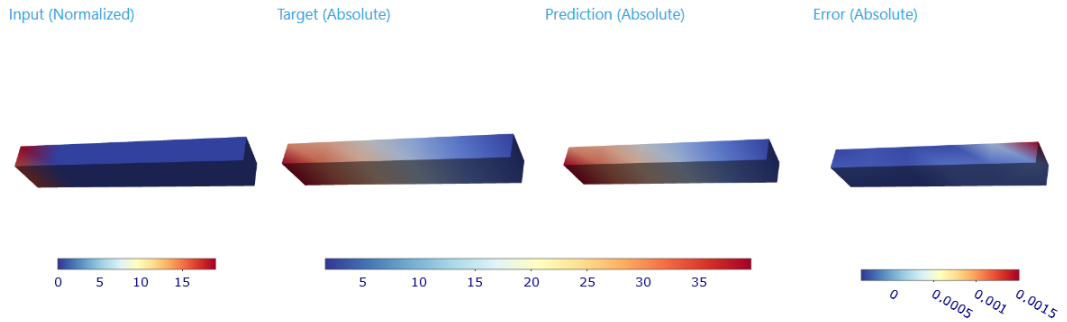


Figure 6.8: GT visualisation for the FN framework of Beam3D DT for specific data sample

Examining the regressor GCN model’s performance on the original geometric dataset, the plot by Fig.6.13 presents individual samples alongside their input, target output, model prediction, and associated error, with $e \in (-0.002, 0.002)$ [MPa] representing the error range.

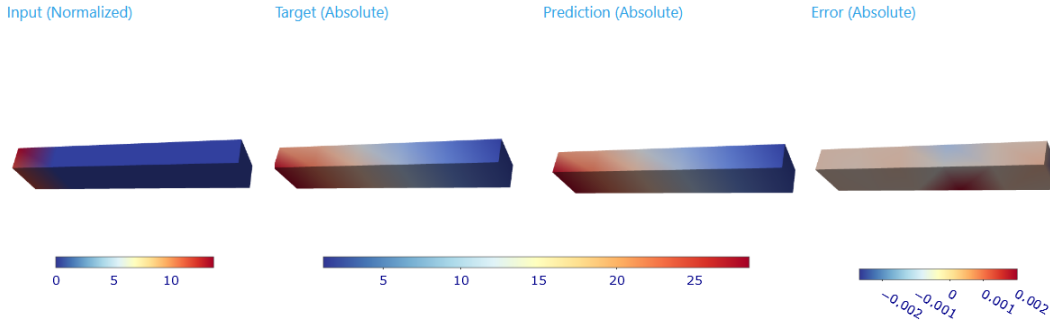


Figure 6.9: GT visualisation for the GCN framework of Beam3D DT for specific data sample

The plot provides a visual representation of the regressor SAGE model’s performance on the original geometric data, depicting each sample’s input, target output, model prediction, and error, where $e \in (-0.01, 0.015)$ [MPa] encapsulates the error range of the sample.

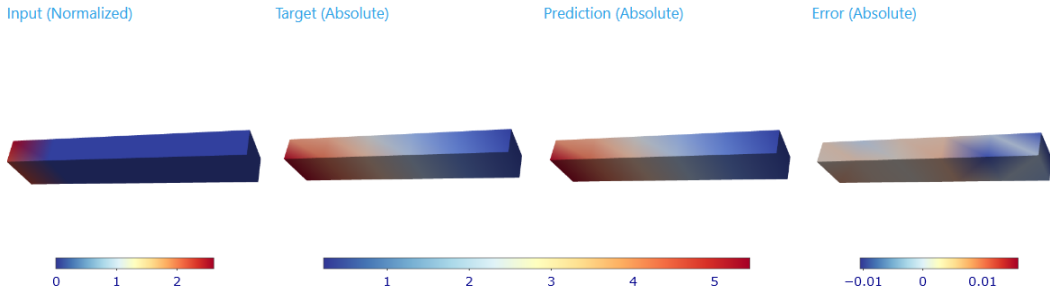


Figure 6.10: GT visualisation for the SAGE framework of Beam3D DT for specific data sample

6.2.3 GT Evaluation: Fibonacci's spring

The plot by Fig.6.11 illustrates the regressor MLR model's performance on the original geometry, displaying individual samples alongside their input, target output, model prediction, and associated error, characterised by $e \in (-0, 350)$ [MPa] for the sample visualised. Visualising by Fig.6.12 the

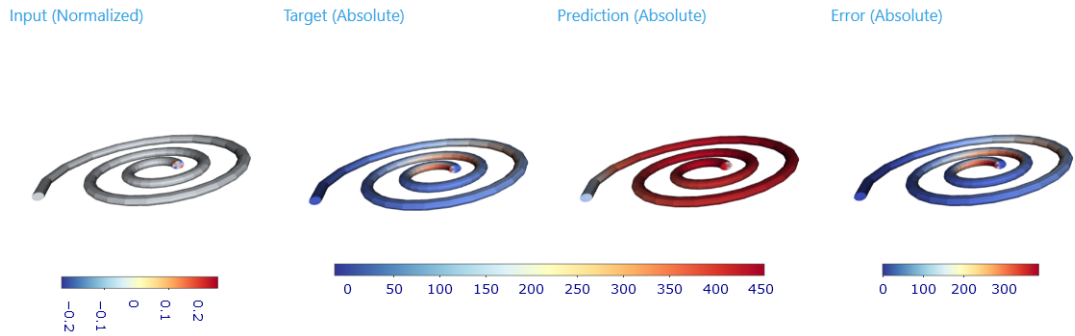


Figure 6.11: GT visualisation for the MLR framework of Fibonacci's spring DT for specific data sample

performance of the regressor FN model on the original geometric data, the plot showcases each sample along with its input, target output, model prediction, and error, denoted by $e \in (-0.45, 0.1)$ [MPa] for the sample visualised.

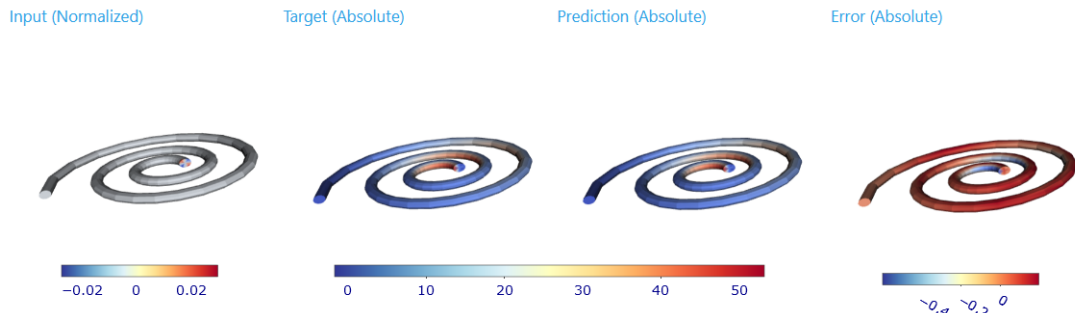


Figure 6.12: GT visualisation for the FN framework of Fibonacci's spring DT for specific data sample

Examining the regressor GCN model's performance on the original geometric dataset, the plot by Fig.6.13 presents individual samples alongside their input, target output, model prediction, and associated error, with $e \in (-0.5, 0.5)$ [MPa] representing the error range for the sample visualised.

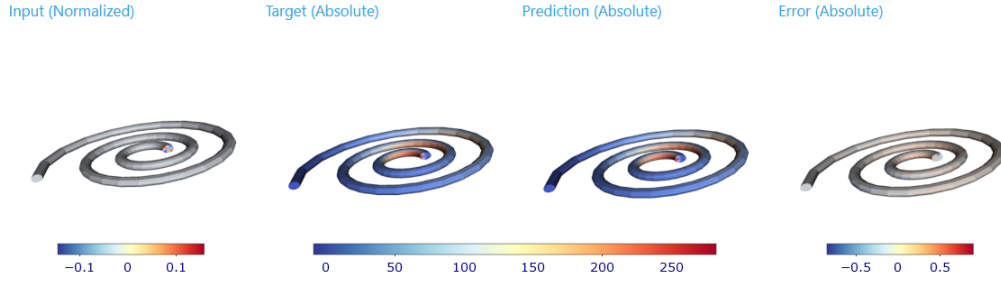


Figure 6.13: GT visualisation for the GCN framework of Fibonacci's spring DT for specific data sample

The plot provides a visual representation of the regressor SAGE model's performance on the original geometric data, depicting each sample's input, target output, model prediction, and error, where $e \in (-0.2, 0.2)$ [MPa] encapsulates the error range for the sample visualised.

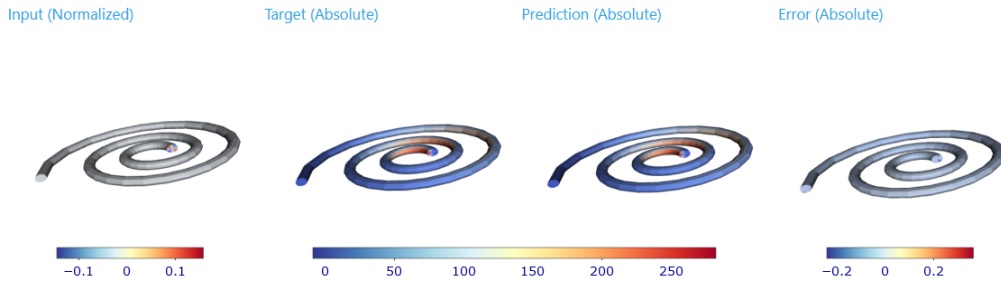


Figure 6.14: GT visualisation for the SAGE framework of Fibonacci's spring DT for specific data sample

6.2.4 GT Evaluation: Plane

The plot by Fig.6.15 illustrates the regressor MLR model's performance on the original geometry, displaying individual samples alongside their input, target output, model prediction, and associated error, characterised by $e \in (0, 650) [MPa]$

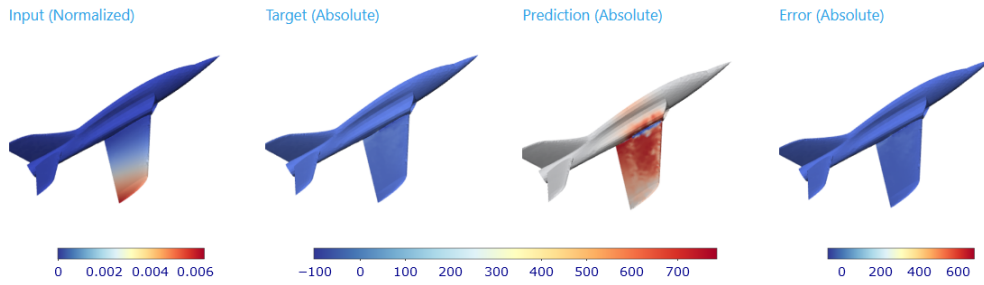


Figure 6.15: GT visualisation for the MLR framework of the Plane DT for specific data sample

Visualising by Fig.6.16 the performance of the regressor FN model on the original geometric data, the plot showcases each sample along with its input, target output, model prediction, and error, denoted by $e \in (-1.5, 0) [MPa]$ for the sample visualised.

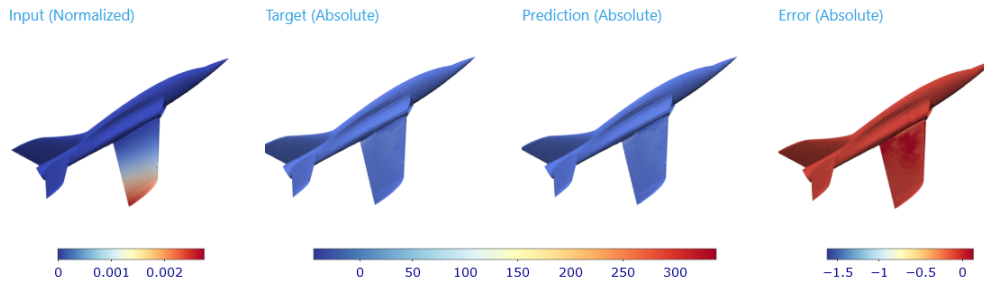


Figure 6.16: GT visualisation for the FN framework of the Plane DT for specific data sample

Examining the regressor GCN model's performance on the original geometric dataset, the plot by Fig.6.17 presents individual samples alongside their input, target output, model prediction, and associated error, with $e \in (-0.4, 0.05) [MPa]$ representing the error range for the sample visualised.

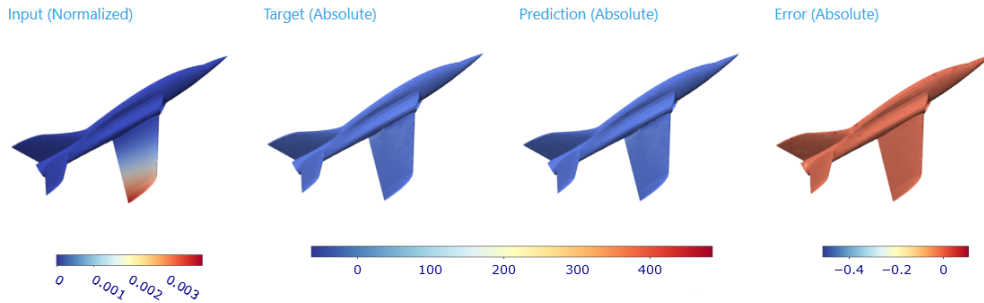


Figure 6.17: GT visualisation for the GCN framework of the Plane DT for specific data sample

The plot provides a visual representation of the regressor SAGE model's performance on the original geometric data, depicting each sample's input, target output, model prediction, and error, where $e \in (-0.05, 0.05) [MPa]$ encapsulates the error range for the sample visualised.

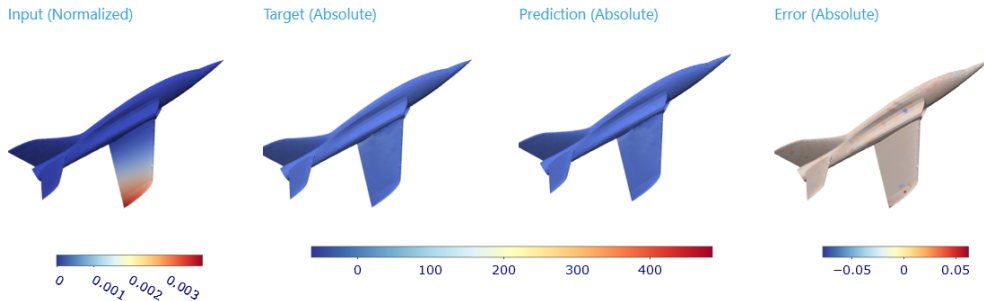


Figure 6.18: GT visualisation for the SAGE framework of the Plane DT for specific data sample

While visualising the ground truth and predicted values provides valuable insights into the performance of suggested regressors, this evaluation form may not be the most appropriate for documenting in a thesis. Furthermore, it is well-known that relying solely on ground truth evaluation can be problematic as it only provides information for a specific data point from the dataset. Therefore, it is essential to use multiple evaluation methods to assess these models' performance comprehensively. For discussion, the focus on visualising the ground truth and predicted values is for illustrative purposes only, and future work should explore other evaluation techniques to provide a more

comprehensive assessment of DT built with the hybrid approach.

■ 6.3 Model Diagnostic of DTs

The Model Diagnostic (MD) results of the GF dataset are presented, where the Model Diagnostic and predicted values of graph regressors are visualised for all datasets. The paramount evaluation is visualising the regressor performance of prediction on a selected validation data point from a dataset referring to a particular load case of mechanical structure. The results provide insight into their efficacy since the regressors are potentially trained to be DT. Furthermore, by comparing the predicted values y_{hat} to the Model Diagnostic, the accuracy of each regressor can be assessed, and the results can be determined as to which ones performed best. An additional minor aspect is to examine the variability in performance across the GF datasets, which sheds light on the generalizability of these models.

6.3.1 MD Evaluation: Beam 2D

MD Summary: DT regressor based on framework MLR of Beam2D shows per Fig.6.19 a required linear trend via Quantile-quantile plot (left). The right plot represents a combination of homoskedasticity and heteroskedasticity with a small relative error. DT would, for its simple framework, work properly.

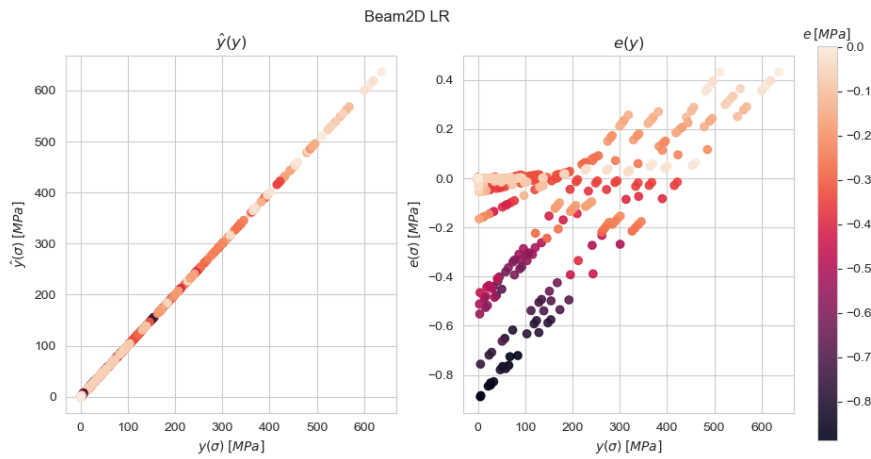


Figure 6.19: MD for DT regressor based on MLR applied on Beam2D

MD Summary: Fig.6.20 illustrating behaviour for DT regressor based on FN of Beam2D shows via Quantile-quantile plot (left) required linear trend depicting. The right plot represents V-shaped nonlinear heteroskedasticity. The prediction error is relatively reasonable.

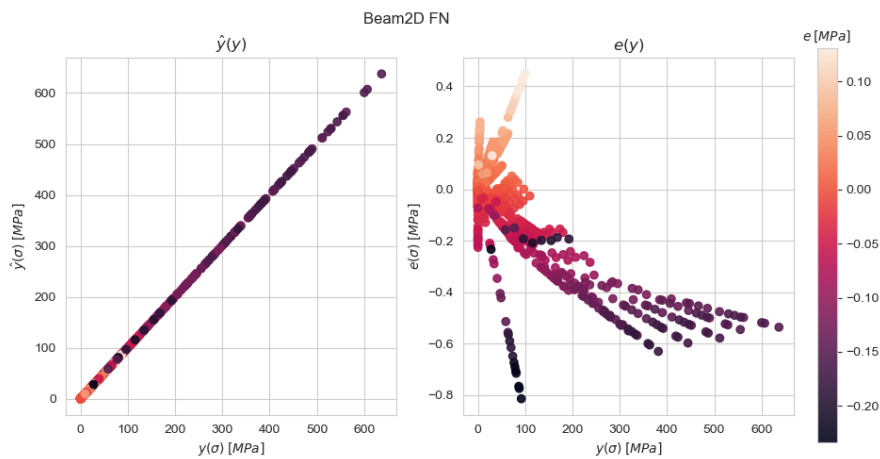


Figure 6.20: MD for DT regressor based on FN applied on Beam2D

MD Summary: DT regressor based on framework GCN of Beam2D depicted in Fig.6.21 shows via Quantile-quantile plot shows proper behaving of DT. Extreme is heteroskedasticity for small predicting values up to 100 MPa. The model prediction would be questionable for small values.

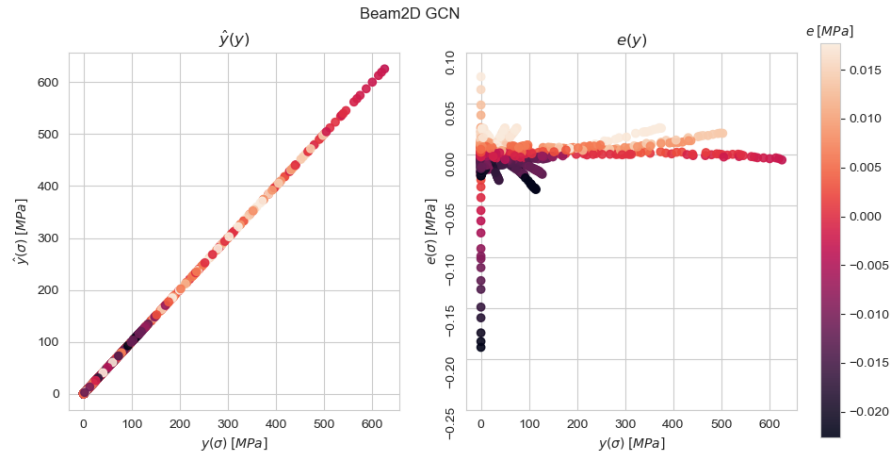


Figure 6.21: MD for DT regressor based on GCN applied on Beam2D

MD Summary: for DT regressor based on framework SAGE of Beam2D shows via Quantile-quantile plot by Fig.6.22 proper behaving of DT. Extreme is heteroskedasticity for small predicting values up to 100 MPa. A deployed model with a small prediction error would be the best framework choice for the DT of Beam2D.

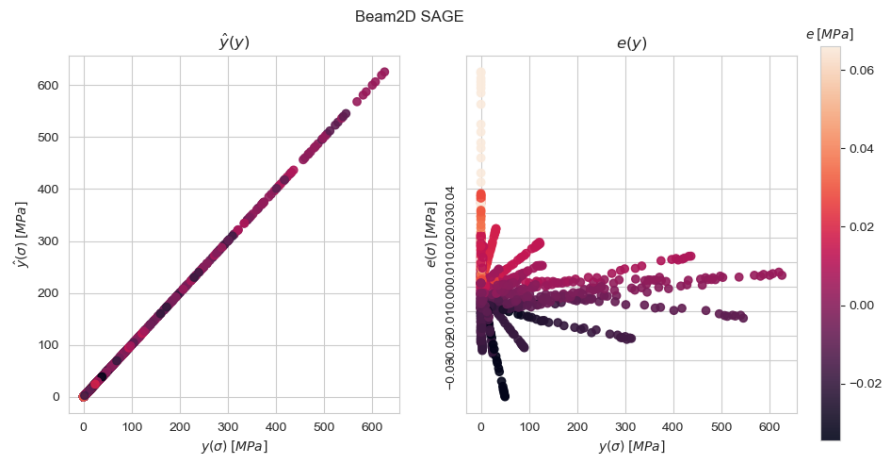


Figure 6.22: MD for DT regressor based on SAGE applied on Beam2D

6.3.2 MD Evaluation: Beam 3D

MD Summary: DT regressor based on framework MLR of Beam3D shows via Quantile-quantile plot shows per Fig.6.23 proper linear behaviour of DT. At the right plot, homoskedasticity is visible; therefore, the model would be the right fit for application.

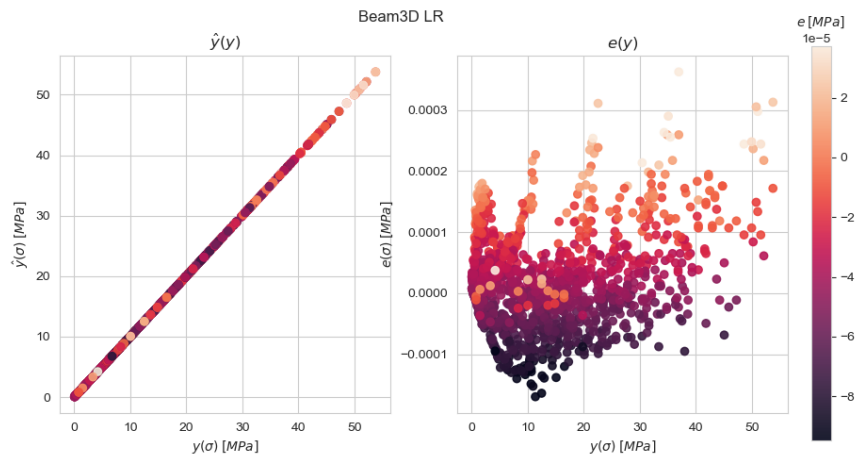


Figure 6.23: MD for DT regressor based on MLR applied on Beam3D

MD Summary: DT regressor based on framework FN of Beam3D shows by Fig.6.24 the proper linear behaviour of DT via Quantile-quantile plot. Extreme nonlinear heteroskedasticity is present, especially for small stresses. DT Model would behave properly with small error values across the whole operation range.

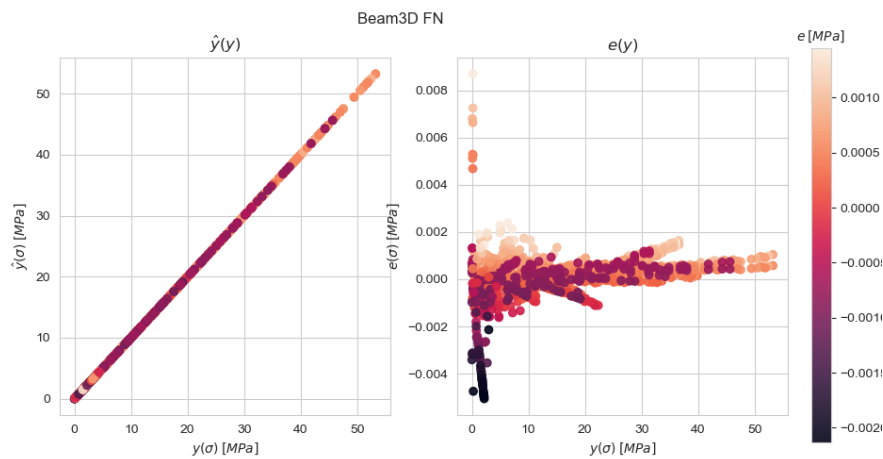


Figure 6.24: MD for DT regressor based on FN applied on Beam3D

MD Summary: DT regressor based on framework GCN of Beam3D shows per Fig.6.25 the expected linear behaviour. The right plot depicts heteroscedasticity with the shape of a funnel. DT Model would behave properly with small error values across the whole operation range.

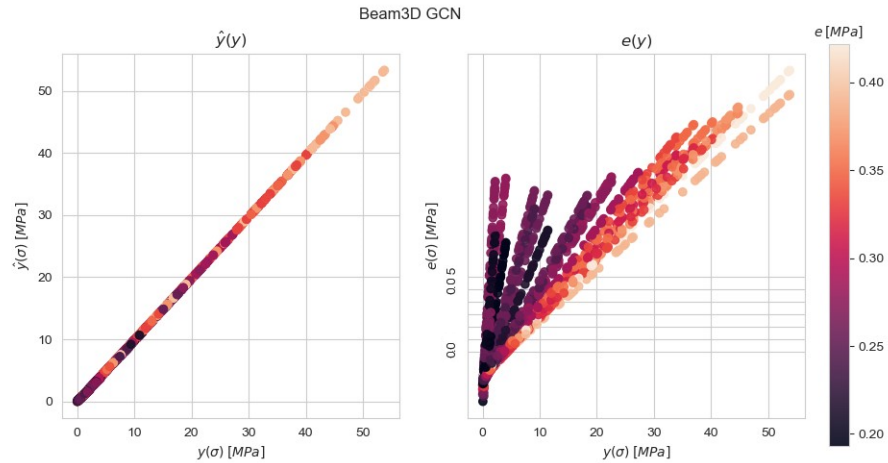


Figure 6.25: MD for DT regressor based on GCN applied on Beam3D

MD Summary: DT regressor based on framework SAGE of Beam3D shows by Fig.6.26 linear via Quantile-quantile plot. The right plot also depicts heteroscedasticity with the shape of an extreme funnel. DT Model would behave properly with small error values across the whole operation range.

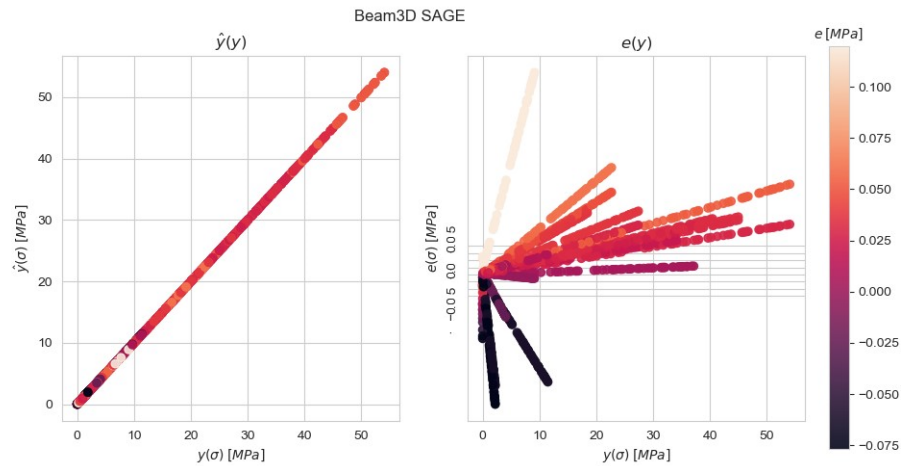


Figure 6.26: MD for DT regressor based on SAGE applied on Beam3D

6.3.3 MD Evaluation: Fibonacci's spring

MD Summary: DT regressor based on framework MLR of Fibonacci's spring depicted in Fig.6.27 shows via Quantile-quantile plot amazing nonlinear behaviour. The homoskedasticity is present, and there is small nonlinear behaviour at the beginning. DT deployed is for this inappropriate.

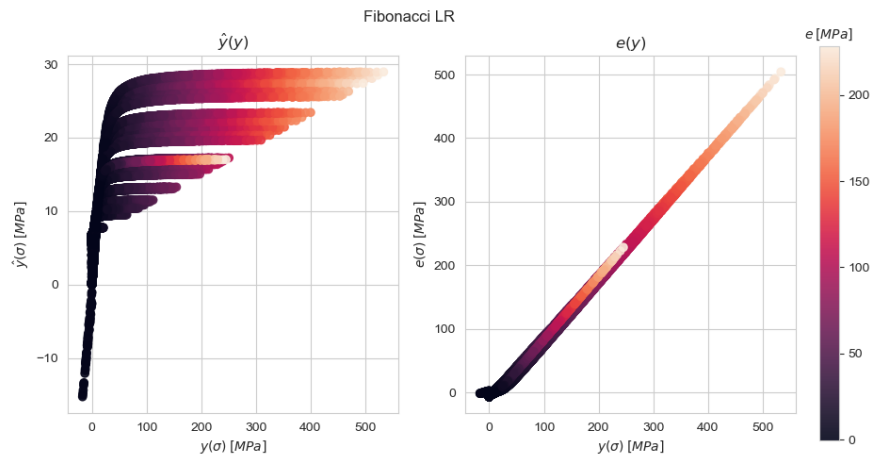


Figure 6.27: MD for DT regressor based on MLR applied on Fibonacci's spring

MD Summary: DT regressor based on FN of Fibonacci's spring per Fig.6.28 shows via Quantile-quantile plot (left) required linear trend depicting. The right plot represents nonlinear heteroskedasticity with reasonable error across the operating range.

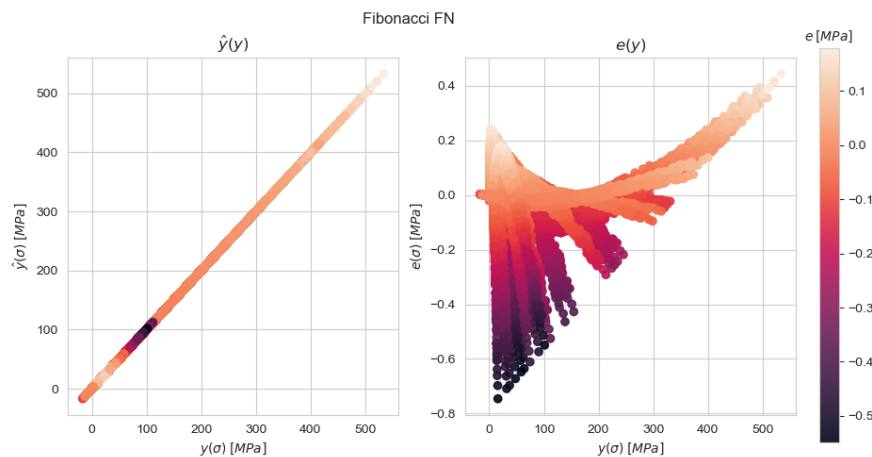


Figure 6.28: MD for DT regressor based on FN applied on Fibonacci's spring

MD Summary: DT regressor based on framework GCN of Fibonacci's spring shows via Quantile-quantile plot is expected to have the correct behaviour by Fig.6.29. Heteroskedasticity in the initial range and homoscedasticity in the rest are present. DT would not behave properly for small stress values.

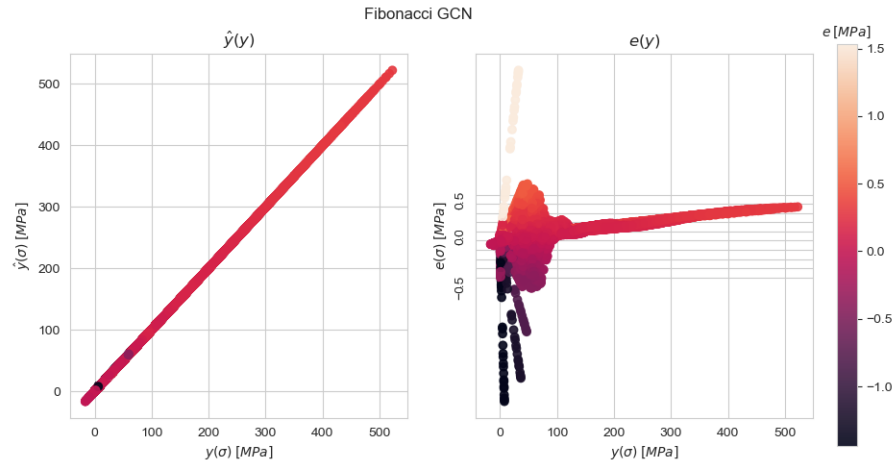


Figure 6.29: MD for DT regressor based on GCN applied on Fibonacci's spring

MD Summary: DT regressor based on framework SAGE of Fibonacci's spring shows via Quantile-quantile plot linear behaviour. The right plot depicts extreme heteroskedasticity in the initial range as indicated by Fig.6.30. DT would not behave properly for small stress values.

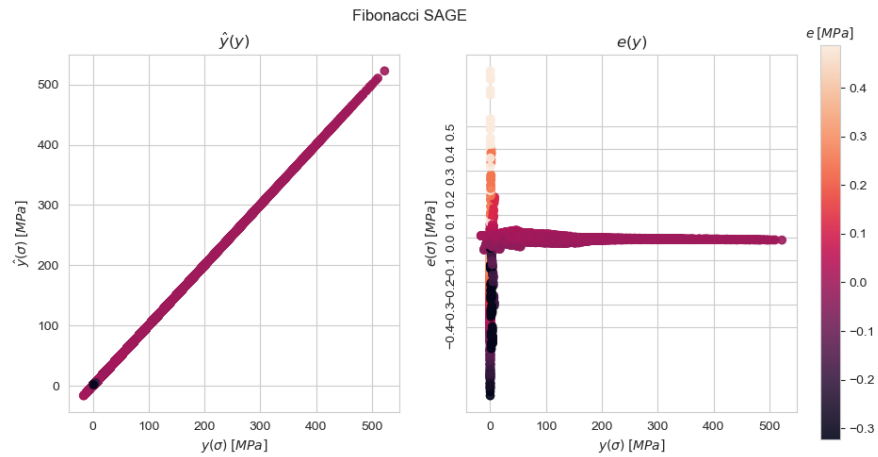


Figure 6.30: MD for DT-based regressor on SAGE applied on Fibonacci's spring

6.3.4 MD Evaluation: Plane

MD Summary: DT regressor based on framework MLR of Plane shows by Fig.6.31 inappropriate non-linearities via Quantile-quantile plot and a right plot showing extreme error. DT based on this regressor is qualified as not feasible deploy to mimic system.

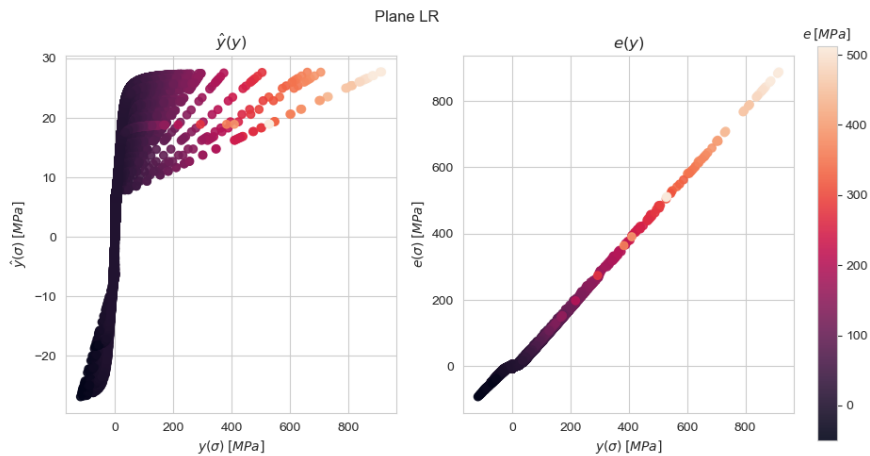


Figure 6.31: MD for DT regressor based on MLR applied on Plane

MD Summary: DT regressor based on FN for Plane DT shows by Fig.6.32 required linear via Quantile-quantile plot (left). The right plot represents nonlinear heteroskedasticity.

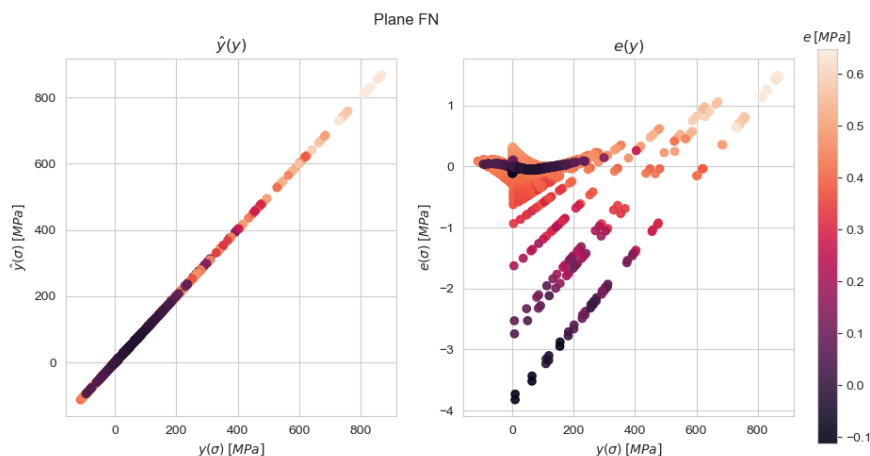


Figure 6.32: MD for DT regressor based on FN applied on Plane

MD Summary: DT based on framework GCN of Plane shows required linear behaviour via Quantile-quantile plot. The right plot depicts extreme absolute heteroskedasticity in the initial range as seen in Fig.6.33. DT would not behave properly for small stress values, but for its relatively small error, it is applicable.

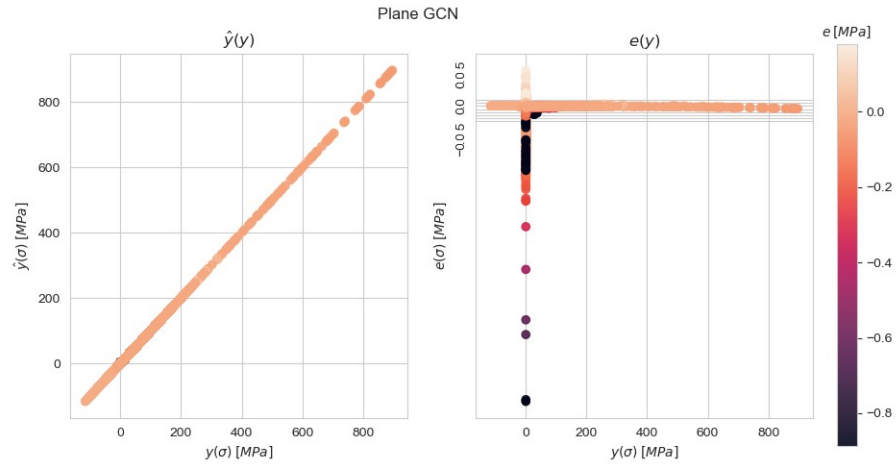


Figure 6.33: MD for DT regressor based on GCN applied on Plane

MD Summary: DT regressor based on framework SAGE of Plane shows with reference to Fig.6.34 proper expected linear behaviour via Quantile-quantile plot. The right plot indicates relatively insufficient prediction for small values of DT. A deployed model for its application would also be the best choice from overall small errors across the whole range.

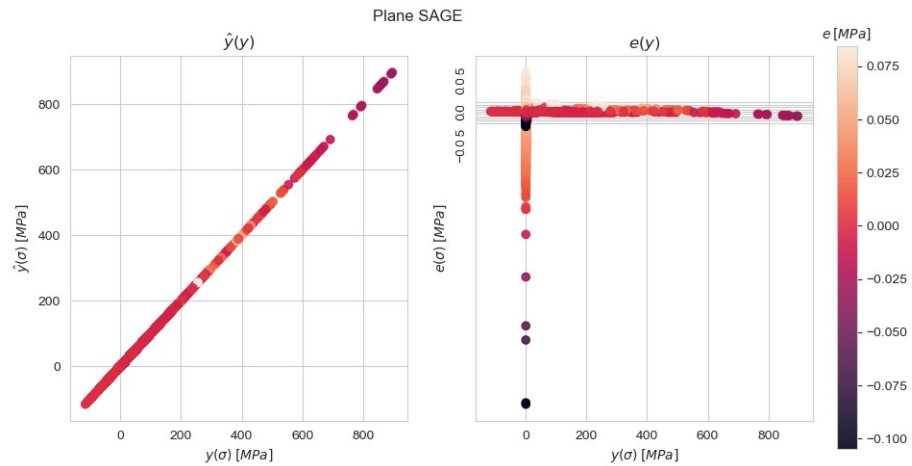


Figure 6.34: MD for DT regressor based on SAGE applied on Plane

While visualising the MD and predicted values provides valuable insights into the performance of suggested regressors, this evaluation form may not

be the most appropriate for evaluating the model's overall performance. Furthermore, relying solely on Model Diagnostic evaluation can be problematic as it only provides information for a specific data point from the dataset. Therefore, it is essential to use multiple evaluation methods to assess these models' performance comprehensively.

6.4 Regressor Training Evaluation

The subsequent section is dedicated to the inspection of regressor training. The best models achieved throughout the training epochs are probed during this inspection. The experiment consists of ten training sessions, and by closely examining the performance and behaviour of the trained regressors, valuable insights can be obtained regarding their strengths, weaknesses, and overall effectiveness. The inspection process involves analysing various aspects of the models, including their predictive capabilities regarding root mean square error, convergence patterns, and generalisation abilities. Through a thorough inspection of the trained regressors, a deeper understanding of their performance is gained, enabling informed decisions about their suitability for specific applications. This inspection phase is crucial in refining and optimising the regressor models, ultimately resulting in improved accuracy and reliability, particularly in predicting the target variable of maximal principal stress.

6.4.1 Training Observation: Beam 2D

Training Summary: The MLR regressor training learning curves display per Fig.6.35 a monotonous trend up to epoch 500, demonstrating consistent improvement in performance. Beyond the epoch turning point, however, the curve begins to flatten, indicating that the framework has converged and is experiencing a slower rate of improvement. This suggests that it has reached its maximum potential for training, and the best models are found within this range. *Training Summary:* The FN regressor training learning

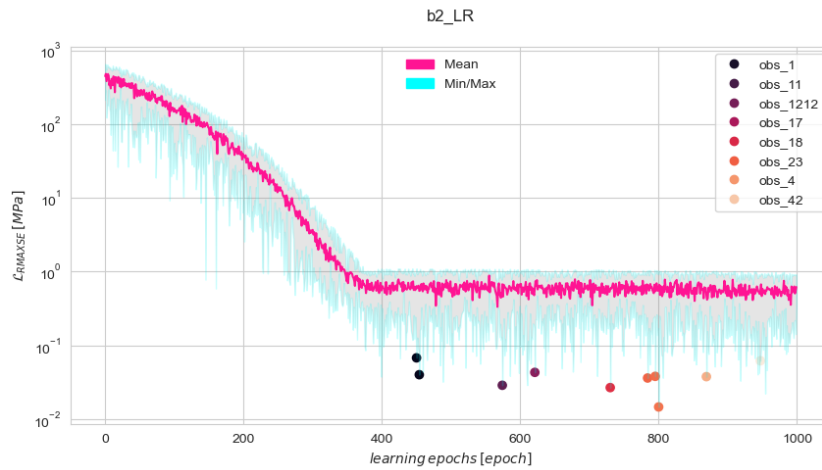


Figure 6.35: Experiments training history of MLR regressor on Beam2D dataset

curves exhibit almost no turning point and quickly converge to their limits, indicating rapid and efficient training as shown in Fig.6.36. This suggests that the framework reaches its maximum potential for training early on, with little further improvement beyond the initial stages.

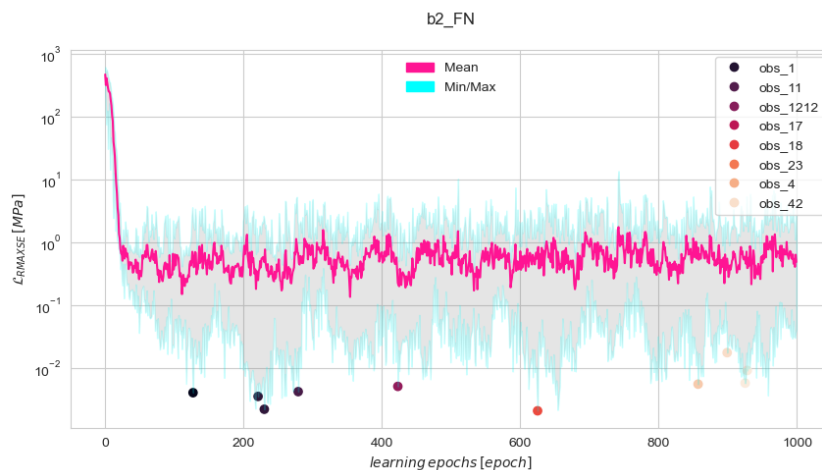


Figure 6.36: Experiments training history of FN regressor on Beam2D dataset

Training Summary: The training learning curves for the GCN regressor display by Fig.6.37 a similar pattern to the previous one, with a slight funnel-shaped turning point occurring after epoch 150. Despite this slight deviation, the curves still converge relatively quickly, suggesting efficient training with early attainment of maximum potential

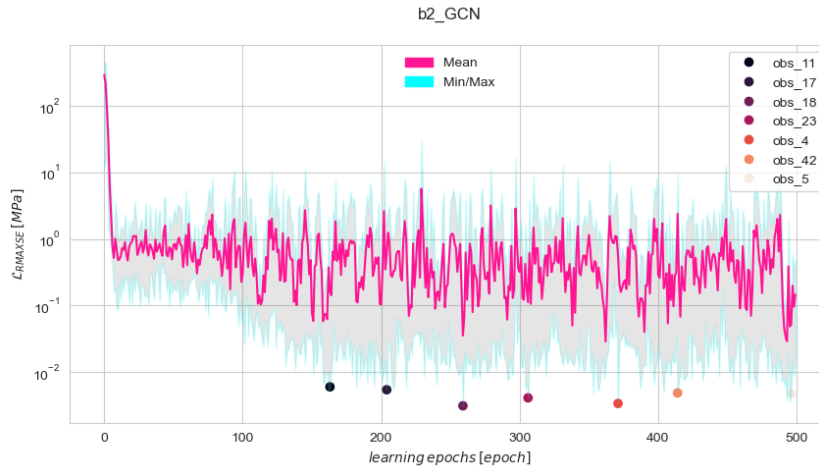


Figure 6.37: Experiments training history of GCN regressor on Beam2D dataset

Training Summary: The SAGE regressor follows a pattern similar to the previous ones, with the first best models typically discovered later, often after epoch 200. However, unlike previous regressors, a significant proportion of the best models are found towards the end of the training process depicted by Fig.6.38. This suggests that the framework requires more iterations to identify optimal models, with a notable concentration of high-performing models towards the latter stages of training.

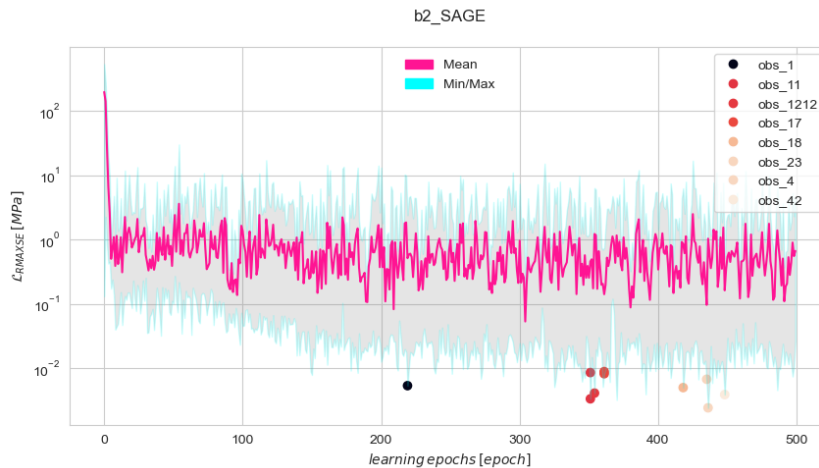


Figure 6.38: Experiments training history of SAGE regressor on Beam2D dataset

6.4.2 Training Observation: Beam 3D

Training Summary: The training learning curves per Fig.6.39 exhibit two turning points for the MLR framework trained on the Beam 3D dataset. The second turning point, around epoch 800, marks a significant change in trend, after which all the best models are typically found.

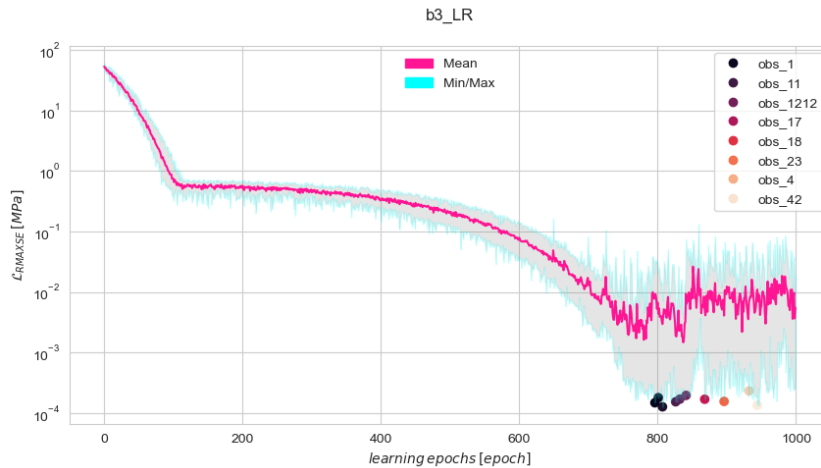


Figure 6.39: Experiments training history of MLR regressor on Beam3D dataset

Training Summary: For the FN framework, the training learning curves 6.40 display almost no turning point, with the first models typically emerging between epochs 150 to 500. However, a noticeable outlier is observed after epoch 800. Additionally, a slight divergence is visible in the learning curves, suggesting some variability in model performance during training.

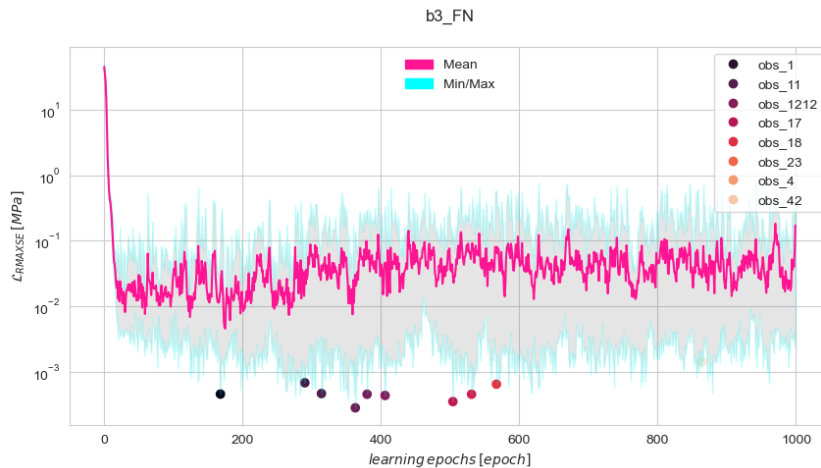


Figure 6.40: Experiments training history of FN regressor on Beam3D dataset

Training Summary: For the GCN framework, the training learning curves demonstrate 6.41 almost no turning point, with the first models typically emerging around epoch 150. The learning curves exhibit stability throughout the training process, indicating consistent and reliable model performance.

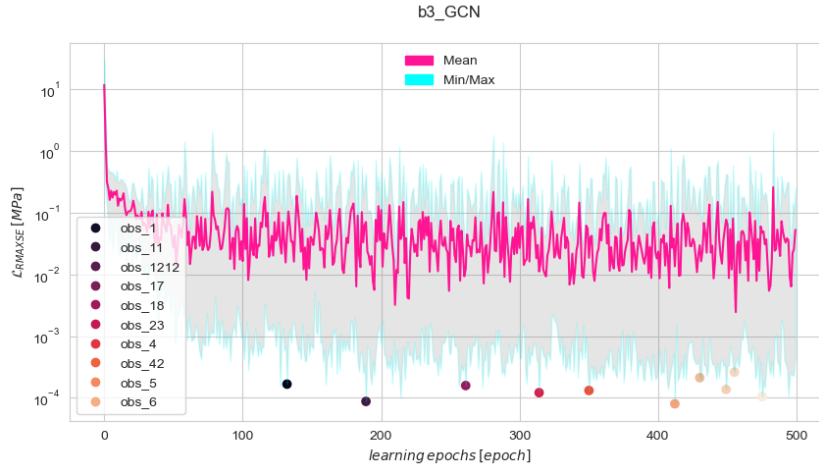


Figure 6.41: Experiments training history of GCN regressor on Beam3D dataset

Training Summary: Similar to the previous regressor, the training learning curves for the GCN regressor using the SAGE framework also demonstrate almost no turning point, as Fig.6.42 shows. The first models typically emerge around epoch 150, and the learning curves remain stable throughout the training process, indicating consistent and reliable model performance.

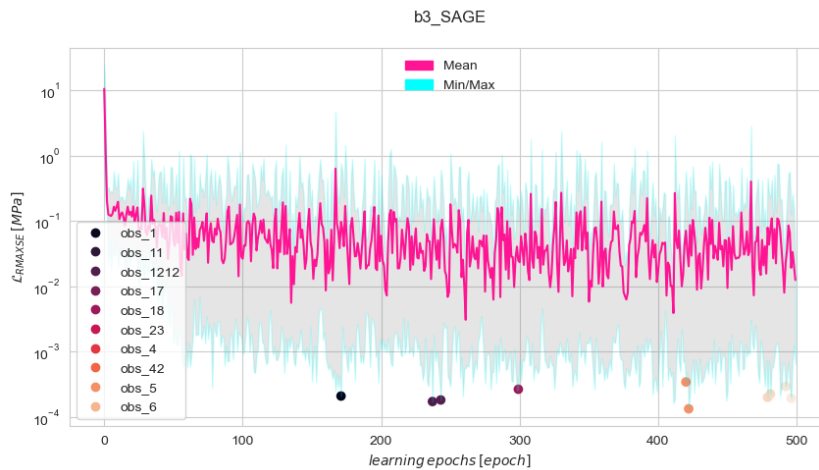


Figure 6.42: Experiments training history of SAGE regressor on Beam3D dataset

6.4.3 Training Observation: Fibonacci's spring

Training Summary: The training learning curves for the MLR framework trained on the Fibonacci spring dataset, the training learning curves exhibit no turning points, as seen in Fig.6.43. However, the presence of outliers emerging after epoch 200 suggests that the regressor may struggle to effectively capture the characteristics of this dataset.

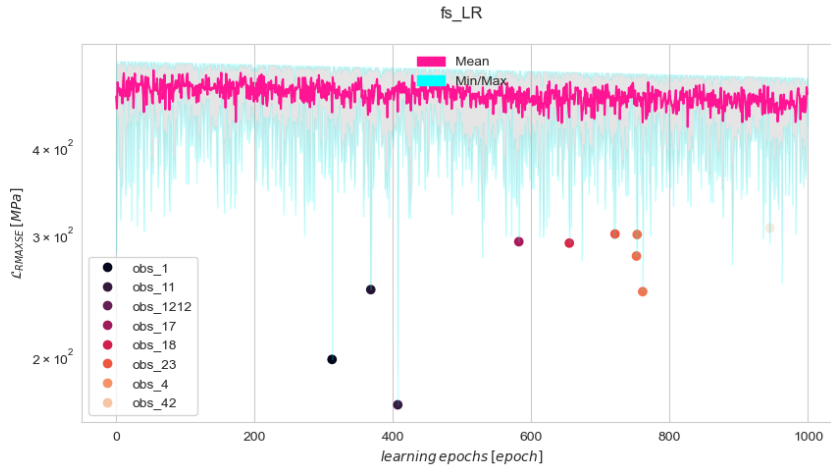


Figure 6.43: Experiments training history of MLR regressor on Fibonacci' spring dataset

Training Summary: For the FN framework, the training learning curves at Fig.6.44 display a turning point around epoch 700 and subsequently, the models converge after epoch 800, suggesting that optimal model performance is achieved at this stage of training.

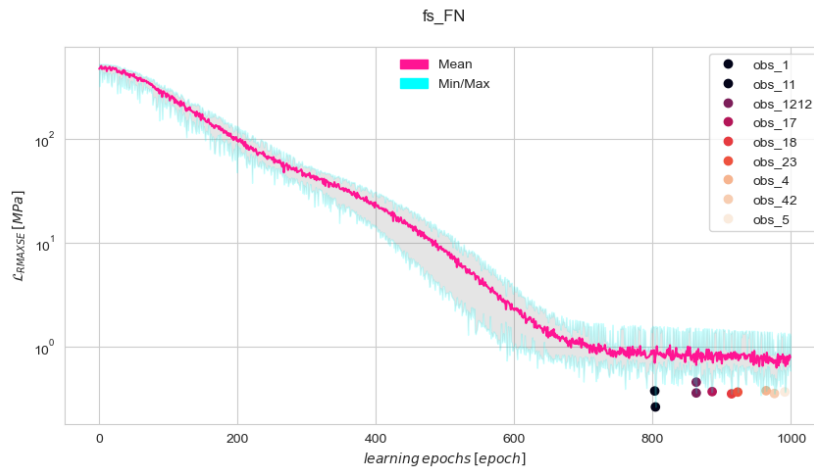


Figure 6.44: Experiments training history of FN regressor on Fibonacci' spring dataset

Training Summary: For the GCN framework, the training learning curves exhibit a turning point around epoch 50, indicating a change in the trend of model performance as depicted by Fig.6.45. The first model typically emerges around epoch 150, followed by discovering additional models over time.

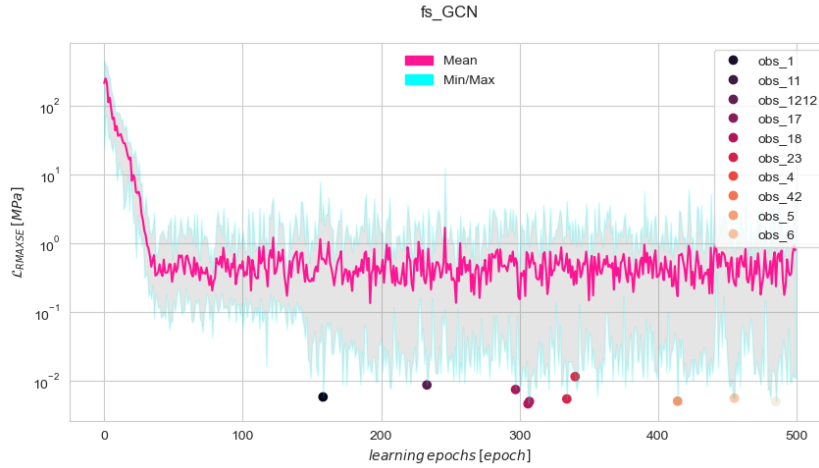


Figure 6.45: Experiments training history of GCN regressor on Fibonacci' spring dataset

Training Summary: The training learning curves for the regressor, utilising the SAGE framework, show by Fig.6.46 a pattern similar to the previous regressor. The best models are typically found after epoch 200, indicating a stable training process with consistent performance.

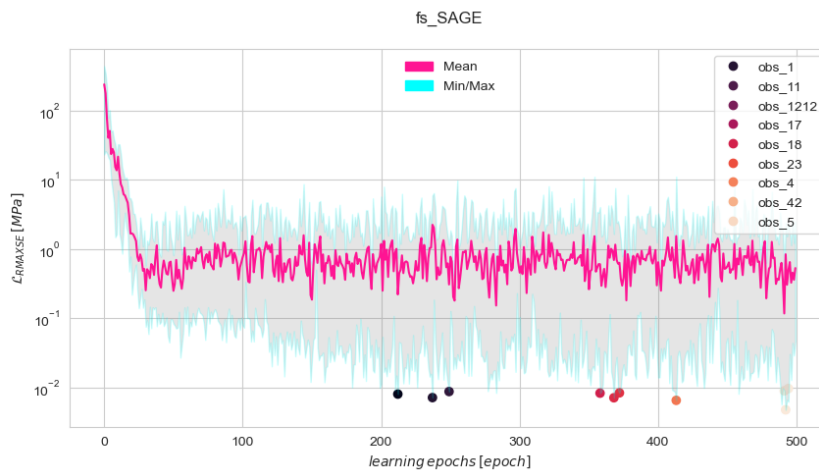


Figure 6.46: Experiments training history of SAGE regressor on Fibonacci' spring dataset

6.4.4 Training Observation: Plane

Training Summary: The training learning curves by Fig.6.47 are visualised for the MLR framework trained on the Plane dataset, and the learning curves exhibit a pattern similar to the MLR regressor trained on the Fibonacci dataset. There are no discernible turning points, and the models may be found as outliers, suggesting potential difficulty in capturing the characteristics of the Plane dataset with the MLR framework. *Training Summary:* For the

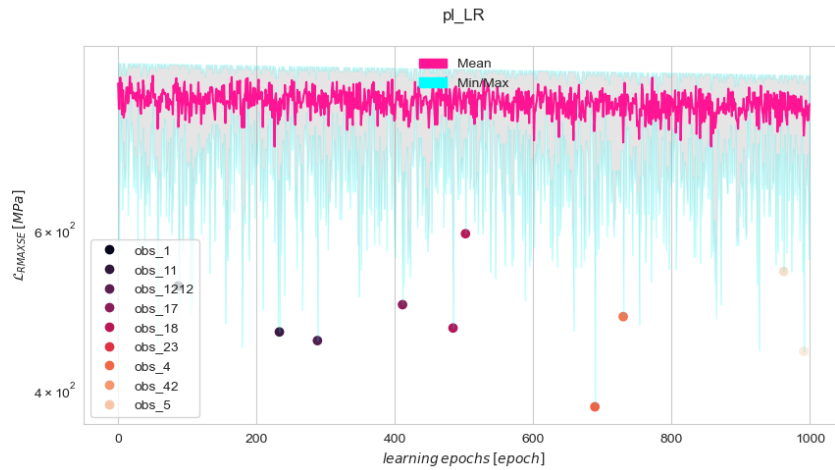


Figure 6.47: Experiments training history of MLR regressor on Plane dataset

FN framework, the training learning curves exhibit a shape resembling an inverse sigmoid function, typically converging after epoch 800. This pattern suggests a gradual improvement in model performance over time, with optimal models usually achieved in the later stages of training as depicted in Fig.6.48.

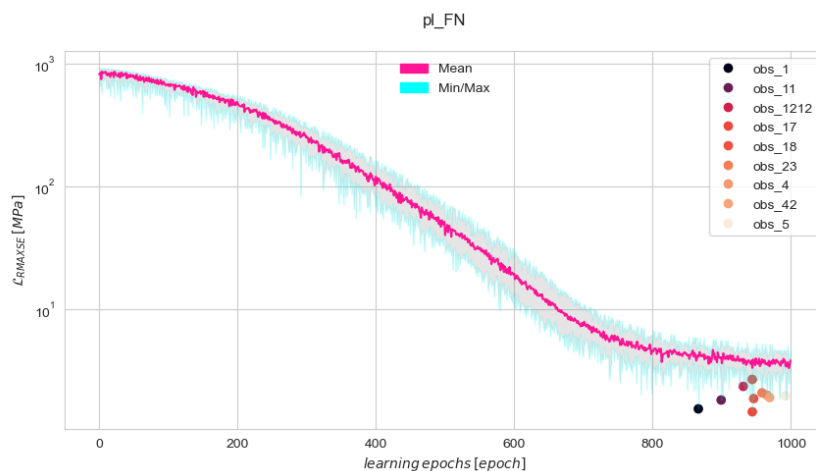


Figure 6.48: Experiments training history of FN regressor on Plane dataset

Training Summary: For the GCN framework, the training learning curves exhibit a pattern similar to the GCN regressor trained on the Fibonacci dataset. The best models are typically found after epoch 200, indicating stable learning with consistent performance as illustrated in Fig.6.49.

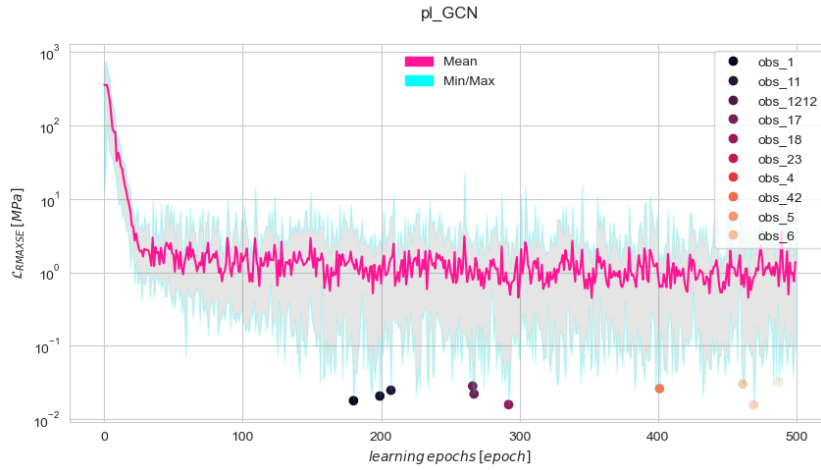


Figure 6.49: Experiments training history of GCN regressor on Plane dataset

Training Summary: For the SAGE framework, the training learning curves show a pattern similar to the previous regressor, with a more steep convergence turn point around epoch 50, as depicted in Fig.6.50. The first model typically emerges around epoch 150, suggesting efficient training with rapid improvement in performance after the initial stages.

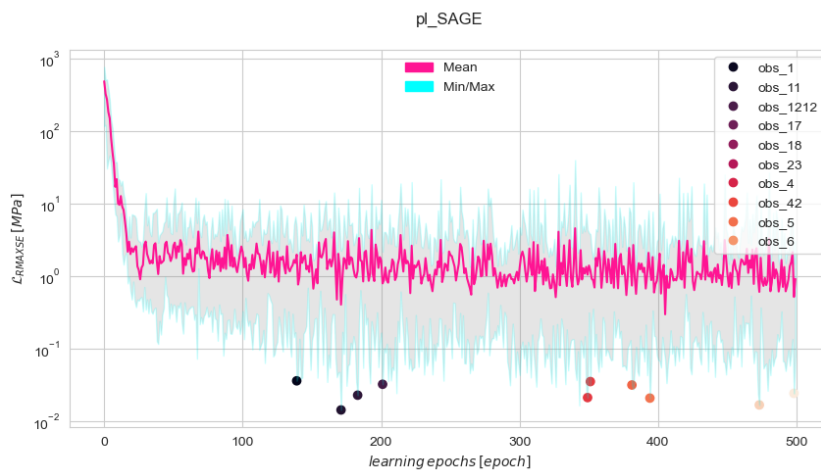


Figure 6.50: Experiments training history of SAGE regressor on Plane dataset

6.5 Graph Reduction Experiment

The comparison is visualised by boxplots per Fig.6.51 for the prediction accuracy $\hat{y}(\sigma)$ achieved by various methods on the smaller datasets (Beam 2D, Beam 3D) of the GF dataset. The boxplots illustrate the distribution of prediction errors for each technique, highlighting their performance variability and effectiveness in capturing underlying patterns in the data.

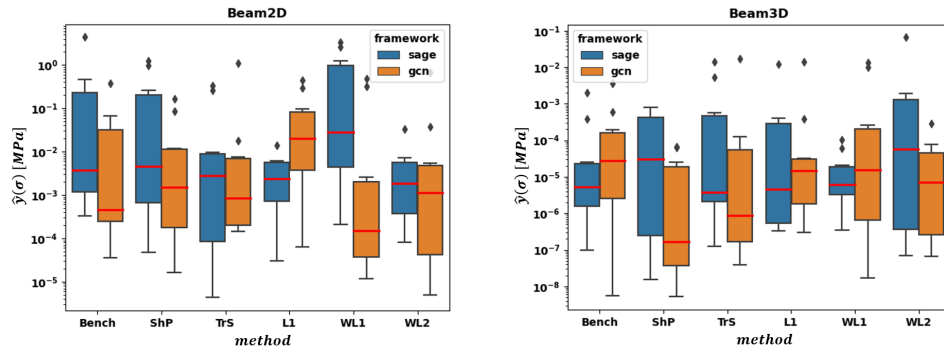


Figure 6.51: Boxplots for prediction accuracy $\hat{y}(\sigma)$ of the smaller nodes count datasets: Beam 2D, Beam 3D

Further comparison by boxplots visualised per Fig.6.52 for prediction accuracy $\hat{y}(\sigma)$ achieved by various methods on the data sets of the higher nodes count (Fibonacci Spring, Plane) of the GF dataset. The box plots illustrate the distribution of prediction errors for each method, highlighting their performance variability and effectiveness in capturing the underlying patterns in the data.

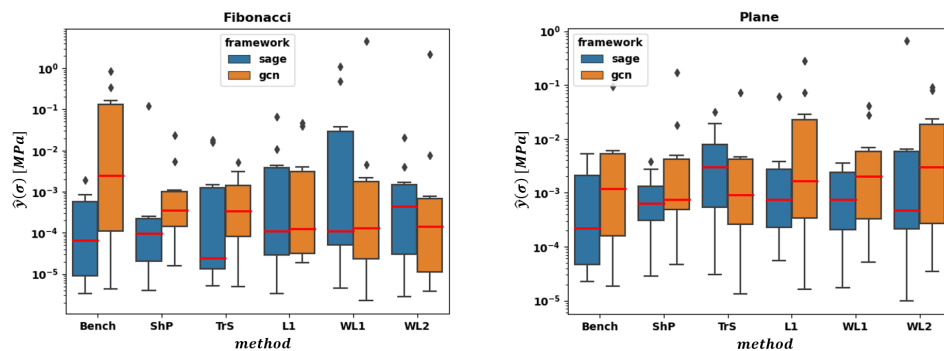


Figure 6.52: Boxplots for prediction accuracy $\hat{y}(\sigma)$ of the higher nodes count datasets: Fibonacci Spring, Plane

Experiments for frameworks GCN and SAGE were conducted for graph reduction, with detailed results available in Appendices Tab.D.2 and Tab.D.3. The comprehensive information provided in these tables serves as a valuable resource for further analysis and understanding of the efficacy of graph reduction techniques in optimizing model complexity and performance.

6.6 Results Discussion

The results balancing section evaluates and ranks the model-centric approach employed in constructing and establishing the DT, with a focus on the essential building blocks within the PM. In this section, the primary assessment tools utilized are visually based, enabling a quick assessment of the distribution and statistical characteristics of crucial metrics driving the maturity of the DT development.

The experiments involved different epoch counts for each model framework: MLR and FN were trained for 1000 epochs, while GCN and GraphSAGE were trained for 500 epochs. The discrepancy in epoch count prompts the question of why GCN and GraphSAGE did not have equal or increased epoch counts. The primary consideration in determining epoch count is computational demand, especially for neural network models dealing with graph structures, which can be computationally intensive due to the complexity of operations and dataset size. The epoch count for GCN and GraphSAGE was set to 500 to balance computational efficiency and model performance. Despite being shorter than MLR and FFNN training durations, this duration still allows graph-based models to capture essential patterns and relationships within the data. Notably, epoch count selection is not solely based on computational demand; factors such as model convergence behaviour and accuracy within the given epoch count also influence the training duration. Therefore, the epoch counts were carefully selected to strike a reasonable balance between computational efficiency and model performance for the specific graph-based frameworks.

The Box plots serve as a valuable visualization tool, enabling the differentiation of model performance across multiple experiments. The distribution of the RMAXSE metric, visualized in a logarithmic scale about the GF benchmarking GNNs, provides insight into the overall spread of model predictions on the validation set. Notably, more straightforward model frameworks such as MLR and FFNN tend to outperform GNN frameworks for small PM (Beam2D and Beam3D). For example, the complex structure of the Plane dataset is better predicted using the FFNN architecture compared to GCN, with a difference of one order. Additionally, the summary table offers further

insight into the performance of individual techniques. For instance, on the dataset \mathcal{D}_{b2} , LR achieves an average RMAXSE of 8 ± 8 , while FN achieves 0.0005 ± 0.0003 and GCN achieves 4 ± 5 . These results underscore the importance of carefully selecting the model framework and highlight the strengths and weaknesses of each approach. Furthermore, considering the training time required for each model (Tab.6.2), it's evident that simpler frameworks like LR and FN have significantly lower training times compared to more complex frameworks like GCN and SAGE. This highlights the trade-off between model complexity and training time, where simpler models may offer faster training but potentially sacrifice performance compared to more complex counterparts.

The GT evaluation was conducted on the *Beam2D* and *Beam3D* datasets, where the FN framework emerged as the most suitable choice due to its slight complexity and balanced performance in providing reasonable predictions. Conversely, the GCN framework was found to be the least suitable choice due to its high computational demand, which would pose a significant obstacle in real-time adaptation. These findings underscore the importance of selecting an appropriate regression framework tailored to the dataset characteristics, with the FN framework demonstrating superior performance while maintaining computational efficiency. The GT evaluation and MD utilities were employed to assess the performance of various regression frameworks on both the *Fibonacci's spring and Plane* datasets. The SAGE framework was observed to outperform others in accurately predicting across the entire operational range, with only slight deviations noted for small values. Conversely, the MLR framework exhibited complete incapability in making accurate predictions for both datasets. The MD analysis further supported these findings, revealing that the SAGE framework maintained a consistent and reliable prediction trend, while the MLR framework displayed significant inconsistencies. These results emphasize the importance of selecting an appropriate regression framework tailored to the dataset characteristics, with the SAGE framework demonstrating superior performance and reliability.

The training learning curves revealed distinct patterns in model convergence and performance across different frameworks and datasets. For instance, stable learning curves with convergent models typically found after epoch 800 were observed for regressors trained on the FN framework. Similarly, regressors utilising the GCN framework displayed consistent performance with optimal models identified after epoch 200. In contrast, rapid performance improvement, particularly after epoch 50, was noted for those trained on the SAGE framework. These observations suggest that the choice of framework can significantly impact the training process and ultimate model performance. Furthermore, insights into model training and convergence dynamics are provided by the varying shapes of the learning curves, including sigmoidal, funnel-shaped, and inverse sigmoidal curves. Overall, the importance of careful framework selection is underscored by these findings, and potential

areas for further investigation, such as understanding the factors driving the observed differences in convergence patterns and exploring strategies to optimize model training for specific frameworks and datasets, are highlighted.

The experiments on graph reduction techniques for modelling mechanical systems provide insights into their potential benefits and challenges. The proposed methods effectively reduce the complexity of finite element models while preserving essential structural features. Findings are illustrated in the box plots of Fig.6.51 and Fig.6.52, along with numerical results in Tab.D.1 and Tab.D.2. The shortest path and TSP solutions offer straightforward and computationally efficient approaches to reducing graph complexity, although their effectiveness may vary depending on the mechanical system's specific characteristics. Laplacian reduction emerged as a more sophisticated technique, requiring careful tuning, particularly regarding edge reduction and characteristic values in the FEM. It leverages the spectral properties of the Laplacian matrix to achieve significant complexity reduction while preserving structural integrity. However, achieving optimal results with this method demands a deeper understanding of the mechanical system's structural characteristics and appropriate parameter selection.

The conducted experiments, aimed at exploring the possibilities of the designed methodology outlined in Section 3.13, have yielded valuable insights and clarity while paving the way for further research and exploration in the field. The initial methodology appears suitable for developing digital twins for mechanical structural systems, marking a promising step forward in this study area.



Chapter 7

Conclusion

The art of mathematically creating models for mechanical structures has been continuously evolving throughout history, with its origins tracing back to the era of Pythagoras or even earlier. Over time, advancements in mathematics, engineering, and technology have propelled the development of sophisticated modeling techniques. These techniques aim to accurately represent the behavior and properties of mechanical structures, enabling engineers and scientists to analyze, design, and optimize various systems. From the early conceptualizations of geometric shapes and basic principles of mechanics to the modern-day application of advanced computational methods, the field of modeling mechanical structures has undergone a remarkable transformation. This ongoing evolution reflects our relentless pursuit of understanding and harnessing the principles governing the physical world, driving innovation and progress in engineering and science.

The thesis was motivated by the desire to make a comprehensive contribution to the field by presenting a well-described and well-supported idea. By delving into the intricacies of the hybrid modelling of digital twins for mechanical structures using the finite element method and graph neural networks, the thesis sought to offer valuable insights, methodologies, and findings that could advance the field. Through careful examination, analysis, and experimentation, the thesis aimed to provide a solid foundation for future research and practical applications in the domain of digital twin modelling. By addressing the research question with a comprehensive approach, the thesis aimed to make a meaningful contribution to the existing body of knowledge. The first part of the thesis captures the dynamically changing status quo of digital twins from a mechanical structural perspective. Acknowledging that the DT topic is a uniformly broad universal tool and that applications can most likely

and experimentation, contributing meaningfully to accomplishing *the 1. Goal is set* at Chapter 2.2.

While PMs can be computationally expensive, their meticulous design yields invaluable insights into complex mechanical systems. Despite the computational demands, the knowledge gained from these models offers significant advantages, particularly in accurately mimicking real-world behaviours. When crafted with care and attention to detail, PMs serve as indispensable tools for understanding and optimizing mechanical systems. The established methodology reached *the Goal2.* by introducing a base to effectively reuse high-value knowledge from an FE model in order to deliver a trained regressor.

The Chapter 3 provides insight into regressors as representative of DD modelling to perform regression tasks on nodes of a graph reflecting a PM, utilizing carefully chosen FEM data to construct the training dataset.

Then, the hypothesis verification by suitable experiments were done to establish a baseline of frameworks, facilitating the selection of optimally performing regressors to *fulfil Objective3.*

In particular, the last goal is addressed in Chapter, especially by Chapter 6, which provides insight on model diagnostic based on overall experiment to understand the performance model in the context of delivering compiled correct transferred behaviour of a particular PM. This highlights assumptions of false system predictions, and therefore, the set monitoring set of metrics is utilised as essential to DTFMG mimicking a mechanical system. *This fulfilling Goal4* aims to use DTHMFG at a regular operation of the physical asset.

7.1 Outlook of possible applications

A DT based on aGNN can be applied in the control loops of systems to optimize their performance and enhance their functionality. Here are a few examples of how a digital twin based on a GNN could be used in control loops:

Model-based control: The digital twin can be used to model the behavior of the system and make predictions about how it will respond to different inputs. This information can be used in a model-based control loop, where the control system adjusts the inputs to the system based on the predicted outputs from the digital twin.

Real-time optimization: The digital twin can be used to optimize the performance of the system in real-time, by learning from the system's past behavior and adapting the control inputs to achieve the desired performance.

Fault detection and diagnosis: The digital twin can be used to detect and diagnose faults in the system by comparing the predicted behaviour of the system to the actual behaviour. If the two differ significantly, this may indicate the presence of a fault that needs to be addressed.

Predictive maintenance: The digital twin can be used to predict when maintenance is needed for the system, by learning from the system's past behavior and identifying patterns that may indicate the need for maintenance.

Digital twins based on a regressor of GNN can be a powerful tool for optimizing and enhancing the performance of control loops in systems. By leveraging the GNN-powered digital twin to accurately model the intricate behaviour of the system and make informed predictions regarding its response to various inputs, a higher level of control precision and system optimization can be achieved, thereby augmenting its overall functionality.

The primary objective of this research is to achieve the highest level of accuracy in modelling. However, it is essential to acknowledge that striving for perfect accuracy may prove to be impractical or even unnecessary in some cases. Instead, the focus is placed on finding methods for synthesizing and interpolating between regression models to operate within the operational space. By doing so, the models can adapt and predict unseen data more effectively. This approach allows for a balance between accuracy and practicality, as the goal is to create models that provide valuable insights and predictions while accounting for the inherent complexities and uncertainties present in real-world systems. The evidence gathered throughout this study supports the endeavour to develop highly accurate models while recognizing the limitations and challenges inherent in achieving absolute perfection.



Appendices



Appendix A

Index

 **A**

Adjacency list, 47

Adjacency matrix, 47

 **D**

Data-Driven modelling, 19

Degree Matrix, 48

Digital Twin, 7

 **F**

Feed-Forward Neural Networks, 23

Finite Element Method, 16

 **G**

Graph Convolutional layer, 54

Graph Neural Network, 23, 52

Graph Reduction, 49

Ground Truth evaluation, 63

H

Hybrid modelling, 13

L

Laplacian matrix, 48

M

Model Diagnostic, 100

P

Physical-based modelling, 15

R

Regressor validation, 64

Root Max Square Error, 62

S

Sample and Aggregated Embeddings Layer, 54

Shortest Path, 50

Spectral Reduction, 51

T

Traveling Salesman Problem, 50

W

Weighted Spectral Reduction, 52

Appendix B

List of Author Publications

Thesis Related

The following author publications are in the scope of the thesis and results have been included or cited in the text.

- A1 M. Ciklamini, “Gf dataset: Mesh-based graphs dataset for a digital twin of a mechanical systems,” in *2023 24th International Conference on Process Control (PC)*, IEEE, June 2023.
- A2 M. Ciklamini and M. Cejnek, “Enhancing digital twin accuracy through optimizing graph reduction of finite element models,” in *Graph Theory-based Approaches for Optimizing Neural Network Architectures*, Springer Nature Optimization and Engineering, 2024. Submitted in 12 Mar 24, in review.
- A3 M. Ciklamini and M. Cejnek, “Greedy sigma: Reinforcement learning inclusion to alter design sequence of finite element modelling,” in *Multi-scale and Multidisciplinary Modeling, Experiments and Design*, Springer Nature, 2024. Submitted in 11 Jan 24, in review.
- A4 M. Ciklamini, “Graph neural network preprocessing for purpose of digital twin of mechanical system,” in *New Methods and Practices in the Instrumentation and Automatic Control and Informatics 2020*, Czech Technical University at Prague, Sept. 2020.
- A5 M. Ciklamini, “Scalers effect on performance of standard machine learning models,” in *New Methods and Practices in the Instrumentation*

Automatic Control and Informatics 2019, pp. 32–34, Czech Technical University at Prague, 2019.

■ Other Publications

The following list of selected author publications have not been included or cited in the text.

- B1 M. CIKLAMINI and J. KOKES, “Distillation of fuzzy model by feed forward neural network for cryptocurrency trend estimation,” 2020.
- B2 P. BERGMANN, M. CIKLAMINI, J. S. HÖLZL, and J. J. REISENBERGER, “Ep4136369 (a1) - component assembly, and wind turbine in which the component assembly is installed.” Application number:EP2021072452920210415.
- B3 J. S. HÖLZL, M. CIKLAMINI, and K. HERBST, “Us11486446b2 plain bearing arrangement.” Publication:US11486446B2 · 2022-11-01.
- B4 J. S. HÖLZL, M. CIKLAMINI, K. HERBST, and A. WALDL, “Ep3942189 (a1) - plain bearing arrangement.” Application number:EP20200718127 20200304.
- B5 J. S. HÖLZL, M. CIKLAMINI, and K. HERBST, “Wo2021207776 (a1) - component assembly, and wind turbine in which the component assembly is installed.” Publication:US11486446B2 · 2022-11-01.

Appendix C

Bibliography

- [1] W. K. Liu, S. Li, and H. S. Park, “Eighty years of the finite element method: Birth, evolution, and future,” *Archives of Computational Methods in Engineering*, vol. 29, p. 4431–4453, June 2022.
- [2] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. S. Hale, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells, “Dolfinx: The next generation fenics problem solving environment,” 2023.
- [3] A. Javidinejad, *Essentials of Mechanical Stress Analysis*. Mechanical and aerospace engineering, CRC Press, 2023.
- [4] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, p. 422–440, 2021.
- [5] L. E. Leal, M. Westerlund, and A. Chapman, “Autonomous industrial management via reinforcement learning: Self-learning agents for decision-making - a review,” *ArXiv*, vol. abs/1910.08942, 2019.
- [6] J. Jia and A. R. Benson, “Residual correlation in graph neural network regression,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery*, ACM, aug 2020.
- [7] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, “Relational inductive biases, deep learning, and graph networks,” 2018.

- [23] Y. Salehi and D. Giannacopoulos, “Physggn: A physics-driven graph neural network based model for predicting soft tissue deformation in image-guided neurosurgery,” *arXiv preprint arXiv:2109.04352*, 2021.
- [24] V. Borozan, S. Fujita, A. Gerek, C. Magnant, Y. Manoussakis, L. Montero, and Z. Tuza, “Proper connection of graphs,” *Discrete Mathematics*, vol. 312, no. 17, pp. 2550–2560, 2012. Proceedings of the 8th French Combinatorial Conference.
- [25] D. Carvajal-Patiño and R. Ramos-Pollán, “Synthetic data generation with deep generative models to enhance predictive tasks in trading strategies,” *Research in International Business and Finance*, vol. 62, p. 101747, 2022.
- [26] F. Cai, A. I. Ozdagli, and X. Koutsoukos, “Variational autoencoder for classification and regression for out-of-distribution detection in learning-enabled cyber-physical systems,” *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2131056, 2022.
- [27] H. White, “A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity,” *Econometrica*, vol. 48, p. 817, May 1980.
- [28] T. S. Breusch and A. R. Pagan, “A simple test for heteroscedasticity and random coefficient variation,” *Econometrica*, vol. 47, p. 1287, Sept. 1979.
- [29] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010.
- [30] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, “Zinc molecule dataset from constrained graph variational autoencoders for molecule design,” *The Thirty-second Conference on Neural Information Processing Systems*, 2018.
- [31] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 (canadian institute for advanced research),”
- [32] T. Zhou and K.-H. Chiam, “Synthetic data generation method for data-free knowledge distillation in regression neural networks,” *Expert Systems with Applications*, vol. 227, p. 120327, 2023.
- [33] Y. Hong, S. Park, H. Kim, and H. Kim, “Synthetic data generation using building information models,” *Automation in Construction*, vol. 130, p. 103871, 2021.
- [34] C. M. de Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, and J. Hodgins, “Next-generation deep learning based on simulators and synthetic data,” *Trends in Cognitive Sciences*, vol. 26, no. 2, pp. 174–187, 2022.

- [35] M. He, Q. Zhao, and H. Zhang, “Multi-sample dual-decoder graph autoencoder,” *Methods*, vol. 211, pp. 31–41, 2023.
- [36] L. Godfrey, “Tests for regression models with heteroskedasticity of unknown form,” *Computational Statistics & Data Analysis*, vol. 50, no. 10, pp. 2715–2733, 2006.
- [37] V. T. White, “Admissibility in discrete Pde,” *Archives of the Angolan Mathematical Society*, vol. 99, pp. 520–522, Dec. 1993.
- [38] D. Hartmann and H. van der Auweraer, “Digital twins,” 2020.
- [39] D. M. Grieves and J. Vickers, “Origins of the digital twin concept,” 2016.
- [40] A. Rasheed, O. San, and T. Kvamsdal, “Digital twin: Values, challenges and enablers from a modeling perspective,” 2020.
- [41] Baillargeon, Rebelo, Fox, Taylor, and Kuhl, “The living heart project: A robust and integrative simulator for human heart function,” 2014.
- [42] Y. Dabiri, K. L. Sack, N. Rebelo, P. Wang, Y. Wang, J. S. Choy, G. S. Kassab, and J. M. Guccione, “Method for Calibration of Left Ventricle Material Properties Using Three-Dimensional Echocardiography Endocardial Strains,” *Journal of Biomechanical Engineering*, vol. 141, 08 2019. 091007.
- [43] R. royce, “Pioneering the IntelligentEngine,” vol. 141, 08 2019.
- [44] J. Artero-Guerrero, J. Pernas-Sánchez, J. Martín-Montal, D. Varas, and J. López-Puente, “The influence of laminate stacking sequence on ballistic limit using a combined experimental/fem/artificial neural networks (ann) methodology,” *Composite Structures*, vol. 183, pp. 299–308, 2018. In honor of Prof. Y. Narita.
- [45] S. Chakraborty, S. Adhikari, and R. Ganguli, “The role of surrogate models in the development of digital twins of dynamic systems,” *Applied Mathematical Modelling*, vol. 90, pp. 662–681, 2021.
- [46] L. A. E. Leal, M. Westerlund, and A. Chapman, “Autonomous industrial management via reinforcement learning: Self-learning agents for decision-making – a review,” 2019.
- [47] R. Biswas and R. C. Strawn, “Tetrahedral and hexahedral mesh adaptation for cfd problems,” *Applied Numerical Mathematics*, vol. 26, no. 1, pp. 135–151, 1998.
- [48] L. Hu, Z. Liu, W. Hu, Y. Wang, J. Tan, and F. Wu, “Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network,” *Journal of Manufacturing Systems*, vol. 55, pp. 1–14, 2020.

- [49] Y. Chen, L. Meng, and J. Zhang, “Graph neural lasso for dynamic network regression,” 2019.
- [50] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” 2015.
- [51] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” 2015.
- [52] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” 2018.
- [53] G. Capuano, “Smart finite elements: An application of machine learning to reduced-order modeling of multi-scale problems,” 2019.
- [54] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” 2017.
- [55] F. Alet, A. K. Jeewajee, M. Bauza, A. Rodriguez, T. Lozano-Perez, and L. P. Kaelbling, “Graph element networks: adaptive, structured computation and memory,” 2019.
- [56] G. Da San Martino, N. Navarin, and A. Sperduti, “Graph kernels exploiting weisfeiler-lehman graph isomorphism test extensions,” in *Neural Information Processing* (C. K. Loo, K. S. Yap, K. W. Wong, A. Teoh, and K. Huang, eds.), (Cham), pp. 93–100, Springer International Publishing, 2014.
- [57] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, “Benchmarking graph neural networks,” 2020.
- [58] N. Choma, F. Monti, L. Gerhardt, T. Palczewski, Z. Ronaghi, Prabhat, W. Bhimji, M. M. Bronstein, S. R. Klein, and J. Bruna, “Graph neural networks for icecube signal classification,” 2018.
- [59] R. Selvan, T. Kipf, M. Welling, A. G.-U. Juarez, J. H. Pedersen, J. Petersen, and M. de Bruijne, “Graph refinement based airway extraction using mean-field networks and graph neural networks,” *Medical Image Analysis*, vol. 64, p. 101751, 2020.
- [60] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.
- [61] X. Suzuki and A. Wu, “Maximality methods in rational graph theory,” *Archives of the Indian Mathematical Society*, vol. 20, pp. 204–223, Dec. 2005.
- [62] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2023.

- [63] L. Wu, P. Cui, J. Pei, and L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore, 2022.
- [64] C. Chen, K. Li, X. Zou, and Y. Li, “Dygm: Algorithm and architecture support of dynamic pruning for graph neural networks,” in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1201–1206, 2021.
- [65] L. Wang, W. Huang, M. Zhang, S. Pan, X. Chang, and S. W. Su, “Pruning graph neural networks by evaluating edge properties,” *Knowledge-Based Systems*, vol. 256, p. 109847, 2022.
- [66] A. Loukas, “Graph reduction with spectral and cut guarantees,” *Journal of Machine Learning Research*, vol. 20, no. 116, pp. 1–42, 2019.
- [67] A. Hald, “On the history of maximum likelihood in relation to inverse probability and least squares,” *Statistical Science*, vol. 14, no. 2, pp. 214 – 222, 1999.
- [68] P. I. Louangrath, “Heteroscedastic analysis in nonlinear modeling,” *SSRN Electronic Journal*, 2013.
- [69] M. Ciklamini, “Gf dataset: Mesh-based graphs dataset for a digital twin of a mechanical systems,” in *2023 24th International Conference on Process Control (PC)*, IEEE, June 2023.
- [70] M. Ciklamini and M. Cejnek, “Enhancing digital twin accuracy through optimizing graph reduction of finite element models,” in *Graph Theory-based Approaches for Optimizing Neural Network Architectures*, Springer Nature Optimization and Engineering, 2024. Submitted in 12 Mar 24, in review.
- [71] M. Ciklamini and M. Cejnek, “Greedy sigma: Reinforcement learning inclusion to alter design sequence of finite element modelling,” in *Multiscale and Multidisciplinary Modeling, Experiments and Design*, Springer Nature, 2024. Submitted in 11 Jan 24, in review.
- [72] M. Ciklamini, “Graph neural network preprocessing for purpose of digital twin of mechanical system,” in *New Methods and Practices in the Instrumentation and Automatic Control and Informatics 2020*, Czech Technical University at Prague, Sept. 2020.
- [73] M. Ciklamini, “Scalers effect on performance of standard machine learning models,” in *New Methods and Practices in the Instrumentation Automatic Control and Informatics 2019*, pp. 32–34, Czech Technical University at Prague, 2019.

Appendix D

Graph Reduction: Algorithm and Tables

Algorithm 4 Calculate \mathcal{G}^{Lred}

Require: \mathcal{G}

Ensure: $L(\mathcal{G}) = D(A) - A(\mathcal{G})$

Require: $\lambda, \mathbf{v} = \text{Hermitian } L(\mathcal{G})$

▷ $L\mathbf{v} = \lambda\mathbf{v}$

Require: η, γ

▷ reduction factor, threshold

1: $k = \|\mathbf{v} * \eta\|$

2: $\mathcal{K} = \text{argsort}(\lambda_0, \dots, \lambda_k)$

▷ Select vector of top eigenvalues

3: $\mu = \lambda[:k]$

▷ selected eigenvalues

4: $\mathbf{v}' = \mathbf{v}[:, :k]$

▷ selected eigenvectors

Ensure: $\mathcal{G}' = \mathcal{G}(\mathcal{N})$

▷ Inherite original nodes

5: **for** $\nu_{i,j} \in \mathcal{G}(\Sigma)$ **do**

6: $p_i \leftarrow \lambda_{\mathbf{s}, \mathbf{i}}, \mathbf{v}_{\mathbf{s}, \mathbf{i}}$

7: **if** $v_i(\sigma) > \gamma$ **then** ▷ Decide on edge cut from σ mechanical stress importance

8: $V' \leftarrow v_i$

9: **end if**

10: **end for**

11: $\mathcal{G}' \leftarrow \text{addEdges}(V')$

▷ Initiate reduced graph

12: Calculate $\mathcal{A}'(\mathcal{G}')$

13: Calculate $D(\mathcal{G}')$

▷ Degree matrix of \mathcal{G}'

14: Extract : $d'_{i,j} \leftarrow D(\mathcal{G}') \cdot I$ ▷ array of nodes degree by identity matrix

15: Find nodes: $e_{d=0} \leftarrow d'_{i,j} < 1$

16: Find nodes: $e_{d=1} \leftarrow d'_{i,j} < 2$

17: Concatenate: $e' \leftarrow e_{d=0} \cup e_{d=1}$

18: Pairing nodes: $e_a \leftarrow \text{roll}(e', 1)$

19: $\mathcal{G}' \leftarrow \text{addEdges}(e_a)$

20: **return** \mathcal{G}'

Dataset	Bench.		ShP		TrS		L1		WL1		WL2		
	[u]	[v]	[v]	[-/-]	[v]	[-/-]	[v]	[-/-]	[v]	[-/-]	[v]	[-/-]	
Beam2D	33	104	1	64	0.62	64	0.62	78	0.75	68	0.65	86	0.82
Beam3D	24	88	1	46	0.52	50	0.57	58	0.66	78	0.89	62	0.70
Fibonacci	1426	7518	1	2850	0.38	3166	0.42	4034	0.53	5170	0.69	3154	0.42
Plane	4758	48210	1	9514	0.19	10780	0.22	21724	0.45	28654	0.59	9772	0.20

Table D.1: Table of results - proposed techniques compared to benchmark.

Dataset	Method Reduction	Benchmark [MPa]	L1 [MPa]	ShP [MPa]	TrS [MPa]	WL1 [MPa]	WL2 [MPa]
Beam2D	mean	5.003E-2	9.259E-2	2.728E-2	1.124E-1	8.109E-2	7.125E-2
	std	$\pm 1.185E-1$	$\pm 1.529E-1$	$\pm 5.422E-2$	$\pm 3.439E-1$	$\pm 1.747E-1$	$\pm 2.087E-1$
Beam3D	mean	4.425E-4	1.476E-3	1.536E-5	1.743E-3	2.413E-3	4.559E-5
	std	$\pm 1.097E-3$	$\pm 4.501E-3$	$\pm 2.678E-5$	$\pm 5.440E-3$	$\pm 5.064E-3$	$\pm 9.087E-5$
Fibonacci	mean	1.380E-1	9.171E-3	3.183E-3	1.157E-3	4.488E-1	2.197E-1
	std	$\pm 2.689E-1$	$\pm 1.819E-2$	$\pm 7.328E-3$	$\pm 1.682E-3$	$\pm 1.417E0$	$\pm 6.915E-1$
Plane	mean	1.123E-2	3.900E-2	2.001E-2	8.758E-3	8.461E-3	2.033E-2
	std	$\pm 2.884E-2$	$\pm 8.793E-2$	$\pm 5.396E-2$	$\pm 2.233E-2$	$\pm 1.433E-2$	$\pm 3.462E-2$

Table D.2: Summary table for GCNframework listing the proposed reduction strategies and their respective performance on presented datasets

Dataset	Method Reduction	Bench [MPa]	L1 [MPa]	ShP [MPa]	TrS [MPa]	WL1 [MPa]	WL2 [MPa]
Beam2D	mean	5.150E-1	3.756E-3	2.473E-1	6.141E-2	7.279E-1	5.536E-3
	std	$\pm 1.364E0$	$\pm 4.329E-3$	$\pm 4.588E-1$	$\pm 1.246E-1$	$\pm 1.228E0$	$\pm 1.021E-2$
Beam3D	mean	2.420E-4	1.317E-3	2.218E-4	2.018E-3	2.173E-5	6.960E-3
	std	$\pm 6.241E-4$	$\pm 3.884E-3$	$\pm 3.044E-4$	$\pm 4.519E-3$	$\pm 3.423E-5$	$\pm 2.062E-2$
Fibonacci	mean	3.917E-4	8.340E-3	1.236E-2	3.640E-3	1.610E-1	2.849E-3
	std	$\pm 6.185E-4$	$\pm 2.054E-2$	$\pm 3.878E-2$	$\pm 7.171E-3$	$\pm 3.554E-1$	$\pm 6.450E-3$
Plane	mean	1.174E-3	7.202E-3	1.139E-3	7.208E-3	1.355E-3	6.729E-2
	std	$\pm 1.790E-3$	$\pm 1.879E-2$	$\pm 1.231E-3$	$\pm 1.024E-2$	$\pm 1.364E-3$	$\pm 2.059E-1$

Table D.3: Summary table for SAGE framework listing the proposed reduction strategies and their respective performance on presented datasets