MACHINE LEARNING-BASED APPROACHES FOR FORWARD AND INVERSE PROBLEMS IN ENGINEERING DESIGN

by

Shadab Anwar Shaikh

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Mechanical Engineering

Charlotte

2024

Approved by:

Dr. Harish Cherukuri

Dr. Taufiquar Khan

Dr. Ayoub Soulami

Dr. Kosta Falaggis

ABSTRACT

SHADAB ANWAR SHAIKH. Machine Learning-Based Approaches for Forward and Inverse Problems in Engineering Design. (Under the direction of DR. HARISH CHERUKURI)

The battery enclosures of current electric vehicles are made of metallic alloys, specifically aluminum or steel. Replacing these metallic alloys with a lightweight material, such as carbon fiber composite, may offer significant weight savings due to its comparable strength-to-weight ratio. Carbon fiber is corrosion-resistant and can be engineered for fire resistance and electrical insulation. It can also be fine-tuned for specific applications and performance needs, such as "crashworthiness".

Designing a carbon fiber-based battery enclosure for crash performance through trial-and-error experiments can be extremely laborious and inefficient. This inefficiency can be alleviated by using virtual manufacturing and structural analysis software. A simulation software chain allows for the virtual manufacturing and crash-testing of the battery enclosure in a single process. However, these numerical simulations are computationally expensive, time-consuming, and may require significant user interaction. Finding optimal design parameters within a reasonable time-frame can be extremely challenging.

The first part of this dissertation addresses the forward problem of accelerating the design of battery enclosures for crash performance. It involves developing a machine learning-based surrogate model of the simulation workflow that can provide quick, approximate results in a fraction of seconds. This can further support design space exploration studies.

Physical phenomena in engineering design are governed by differential equations, typically solved in a forward manner with known physical parameters, initial and/or boundary conditions, and a source term. However, there is often a need to reconstruct the source term from available measurement data, which may be corrupted with noise, along with the initial and/or boundary conditions, and physical parameters. These types of problems are known as inverse problems, more specifically, inverse source problems. Inverse source problems are

often ill-posed and are usually solved by iterative schemes and optimization techniques with regularization, which can be time-consuming. In recent years, machine learning approaches have shown promise in managing ill-posed problems and handling noisy data.

The second part of this dissertation addresses a specific type of inverse source problem, known as the dynamic load identification problem, which involves determining the time-varying forces acting on a mechanical system from the sensor measurements. The study begins with the development of a deep learning model that leverages physics information to infer the forcing functions of both linear and nonlinear oscillators from observational data. Furthermore, the study leads up to a development of a physically consistent surrogate model that is capable of providing robust predictions from the noisy observations without the need to explicitly solve the differential equation.

DEDICATION

To my Mom, who taught me the importance of education, and to my Dad, whose fortitude keeps inspiring me. They are missed but forever in my heart.

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to Dr. Harish Cherukuri for his confidence, continuous motivation, support, and mentoring during my time at UNC Charlotte. I extend my heartfelt thanks to Dr. Taufiquar Khan, Dr. Ayoub Soulami, and Dr. Kosta Falaggis for agreeing to serve on my dissertation committee.

I am grateful to the Department of Energy, PNNL, and ESI for their support of the HPC4EI project. I also appreciate the support from the Department of Mechanical Engineering and Engineering Science and the Graduate School through assistant-ships and fellowships. Special thanks are due to Dr. Ayoub Soulami for his invaluable support during my internship and for providing data for this work.

I am thankful for the technical support from ESI and the essential resources and collaboration from my colleagues at PNNL, which have been critical to my dissertation.

My gratitude also extends to my brother, sisters, and friends for motivating and supporting me directly and indirectly throughout this journey. Lastly, I am profoundly thankful to my wife for her understanding and support during this time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ARD: Automatic Relevance Determination

CI: Confidence Interval

CLE: Crush Load Efficiency

DOE: Design of Experiments

EV: Electric Vehicle

FEA: Finite Element Analysis

GPR: Gaussian Process Regression

HPC High-Performance Computing

ISP: Inverse Source Problem

LASSO Least Absolute Shrinkage and Selection Operator

LH: Latin Hypercube

MCS: Monte Carlo Simulations

ML: Machine Learning

NN: Neural Network

PINNs: Physics Informed Neural Networks

SEA: Specific Energy Absorption

CHAPTER 1: INTRODUCTION

The motivation for this dissertation work stems from two commonly encountered problems in engineering design: the forward problem and an inverse problems. Each of these are discussed in detail below.

1.1 Forward problems in engineering design

Forward problems in engineering play a crucial role in predicting the outcomes or responses of systems given specific inputs or conditions. Common applications include Computational Fluid Dynamics (CFD) for flow simulation, stress analysis in structures, and electromagnetic field simulations. For instance, in CFD, the known inputs might be the fluid properties and boundary conditions, with the goal being to predict the flow pattern, pressure distribution, and temperature variation throughout the fluid domain. Similarly, in structural analysis, the inputs include material properties, geometry, and external forces, while the outputs are the stress distribution or deformation of the structure. Electromagnetic simulations predict the distribution of electromagnetic fields around objects, given the source current and material properties.

Numerical methods, such as the Finite Element Method (FEM) and Finite Volume Methods (FVM), are commonly employed to simulate scenarios that are impractical or impossible to test physically by solving these problems. This approach can offer valuable insights into the system's behavior under various conditions, which can be helpful in the design and optimization of the system.

However, these numerical simulations can often be computationally expensive, time-consuming, and may require significant user interaction. This makes the exploration of large design spaces within a reasonable time frame intractable, impacting the overall design cycle time.

To overcome this difficulty, there is a pressing need for lightweight reduced-order models that can provide quick approximate results. Such models would serve as an efficient alternative to time-consuming simulations, allowing for rapid evaluations of design parameters. By reducing computational costs, minimizing user interaction, and accelerating the exploration process, reduced-order models can streamline decision-making and enable faster design iterations.

1.2 Inverse problems in engineering design

In many practical problems such as material characterization, shape optimization, and inverse heat transfer analysis, there is often a need to determine the cause or understand the system from the known output, as shown in Figure 1.1. For example, in material characterization, we often need to infer the material properties by studying its response to a known input. Similarly, finding the ideal shape of a component in shape optimization requires understanding the relationship between desired performance and the component's shape. Furthermore, some heat transfer problems involve identifying the heat sources or boundary conditions that would produce a desired temperature distribution. These types of problems are known as inverse problems, which are often ill-posed [11]. A well-posed problem should have a unique solution that continuously depends on the available data [2]. Unfortunately, most inverse problems often lack known analytical solutions. As a result, they are typically solved using iterative schemes and optimization techniques, which can be time-consuming.

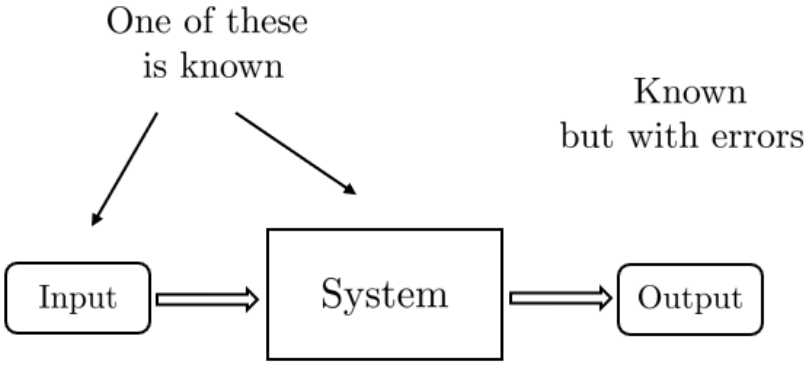


Figure 1.1: Schematic of an inverse problem [2]

Recently, data-driven approaches have gained popularity as alternatives for solving inverse problems [12]. These methods utilize machine learning and statistical modeling to learn the cause-effect relationships directly from data. By training on large datasets, data-driven techniques can effectively infer the desired information, alleviating the need for explicit analytical solutions or time-consuming iterations. Consequently, they have emerged as efficient and scalable solutions for a wide range of inverse problems.

1.3 Why use machine learning (ML)?

Machine learning, particularly deep learning, has emerged as a powerful tool in various fields due to its unique advantages. It possesses the ability to analyze vast amounts of data, uncovering patterns, trends, and complex correlations that may not be apparent through traditional approaches. This ability to handle large-scale data sets makes machine learning especially valuable in today's era of big data. Furthermore, machine learning excels at learning intricate and non-linear relationships, which may be challenging to capture using conventional methods. Deep learning, a sub-field of machine learning, is particularly adept at recognizing complex patterns and extracting meaningful insights from data.

In addition, machine learning algorithms can also handle noisy data to some extent, which is often encountered in real-world applications. Moreover, machine learning has the potential to tackle ill-posed problems [13]. Traditional approaches may struggle with these types of problem, but machine learning algorithms can provide viable solutions by leveraging their ability to learn from data and make predictions based on learned patterns. Another advantage of machine learning is its scalability. It can be easily scaled up to handle large datasets, enabling the training of models on extensive amounts of data, further enhancing their accuracy and predictive capabilities.

Despite its numerous advantages, machine learning does have some limitations that should be considered. Deep learning models, in particular, can be challenging to interpret and understand due to their complex architectures and internal workings. This lack of interpretability can make it difficult to gain insights into the underlying processes and may require additional

efforts for transparency and explainability. Additionally, there is a risk that machine learning models may generalize poorly, meaning that they may not perform well on unseen data or in situations that differ significantly from the training data. This limitation emphasizes the importance of careful validation and testing to ensure the robustness and reliability of machine learning models. Another consideration is that machine learning predictions can sometimes be physically inconsistent [13], which means that they may not align with known laws and principles. Although machine learning algorithms can uncover patterns and correlations in the data, ensuring the physical consistency of the predictions requires additional domain knowledge and constraints.

Taking into account these advantages and limitations, this research aims to harness the potential of machine learning to expedite the exploration of the design space and tackle inverse problems in engineering design.

1.4 Organization of remaining sections

The remaining sections of this dissertation are organized as follows:

Chapter 2 delves into the utilization of machine learning (ML) for solving the forward problem of accelerating the design of electric vehicle (EV) battery enclosures. This section provides a detailed examination of the current challenges associated with EV battery enclosures, the potential benefits of transitioning to carbon fiber composites, and the application of thermoforming and crash simulations. It further discusses the limitations of existing simulation workflows and introduces surrogate modeling as innovative solutions. The chapter elaborates on the research methodology, encompassing data generation for surrogate modeling, the development process of these models, and the empirical results demonstrating the efficacy of the ML models. It concludes with an analysis of uncertainty and a comprehensive discussion on the research implications.

Chapter 3 focuses on employing ML, particularly Physics-Informed Neural Networks (PINNs), for solving inverse source problems within engineering design. It outlines the challenges posed by inverse source problems, introduces the concept and application of PINNs, and presents

Duffing's equation as a case study. The literature review, methodology for estimating forcing functions from response data using PINNs, and the presentation of results are covered in detail. This chapter concludes with a discussion that reflects on the study's contribution to solving inverse problems with ML and identifies avenues for future research.

The dissertation culminates in a conclusion that presents the key findings, highlighting the significant role of ML in enhancing the efficiency and effectiveness of engineering design processes and solving inverse problems.

CHAPTER 2: ACCELERATING DESIGN OF ELECTRIC VEHICLE (EV) BATTERY ENCLOSURES USING ML

2.1 Background

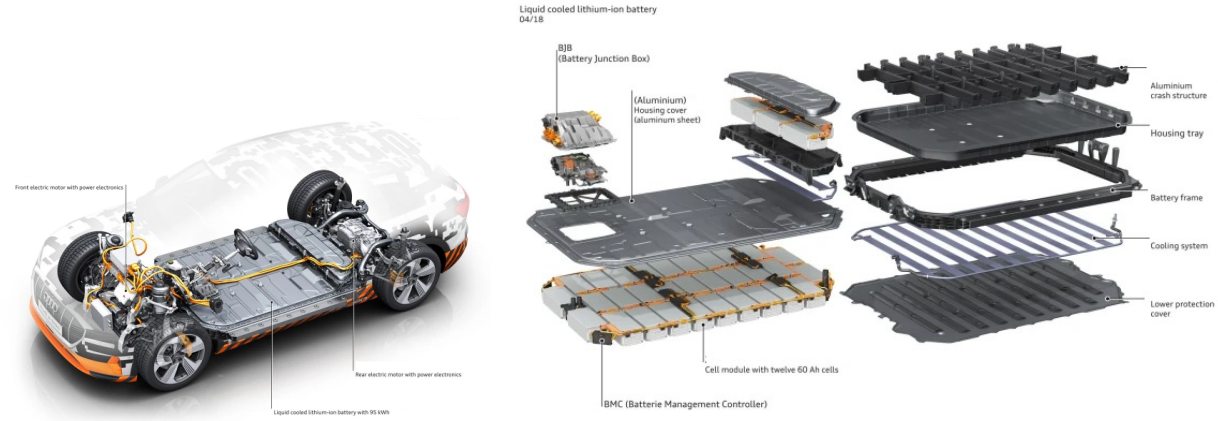
This section aims to provide background and insight into the first problem statement that this research work tackles.

2.1.1 Problems with existing EV battery enclosures

The increasing demand for electric vehicles (EVs)[14] presents a promising solution to address environmental concerns and reduce dependence on fossil fuels. However, like any technology, there are advantages and disadvantages associated with the adoption of EVs. First, they produce zero emissions, contributing to improved air quality and reduced greenhouse gas emissions, thus mitigating the adverse effects of climate change. Second, electric vehicles have lower maintenance costs compared to traditional internal combustion engine vehicles, as they have fewer mechanical components that require regular servicing. In addition, electric vehicles operate silently, reducing noise pollution in urban environments.

However, there are also certain challenges and disadvantages that need addressing. One major concern is the high cost associated with electric vehicles, particularly in terms of the initial purchase price. Another limitation is the limited driving range† of current EVs. While significant advancements have been made in battery technology, the driving range of electric vehicles is still comparatively shorter than that of conventional vehicles, which may lead to "range anxiety" for drivers. Furthermore, the weight of EVs is a significant factor that affects their driving range [15]. The battery module, in particular, contributes significantly to the overall weight of EVs.

Currently, most electric vehicles use aluminum alloy-based battery enclosures, such as the



(a) (b)

Figure 2.1: (a) Chassis (b) Battery module of Audi e-tron [3]

Audi e-tron, shown on Figure 2.1, and Nissan LEAF[3]. Aluminum is relatively lightweight‡; however, there are other materials available that offer comparable strength-to-weight ratios. The exploration of alternative lightweight materials is of great interest, as they have the potential to address multiple challenges simultaneously. These materials not only reduce the weight of the battery module but also possess desirable properties, such as electrical insulation and corrosion resistance.

2.1.2 Replacing aluminum based enclosure with carbon fiber composites

Carbon fiber composites offer exceptional strength-to-weight ratios, meaning they provide high structural integrity while being considerably lighter than aluminum. This weight reduction can have numerous benefits, such as improved fuel efficiency in transportation or increased payload capacity in aerospace applications [16]. Carbon fiber composites have unique properties that make them highly desirable for specific applications. Their composition and manufacturing processes can be fine-tuned to meet specific requirements, allowing for tailored solutions [17]. This adaptability enables engineers to optimize the performance

† These are being improved continually
‡ Compared to steel, which was a commonly used battery enclosure material in the past

and characteristics of the material, resulting in enhanced functionality and efficiency in various industries. In addition to weight savings and customization, carbon fiber composites

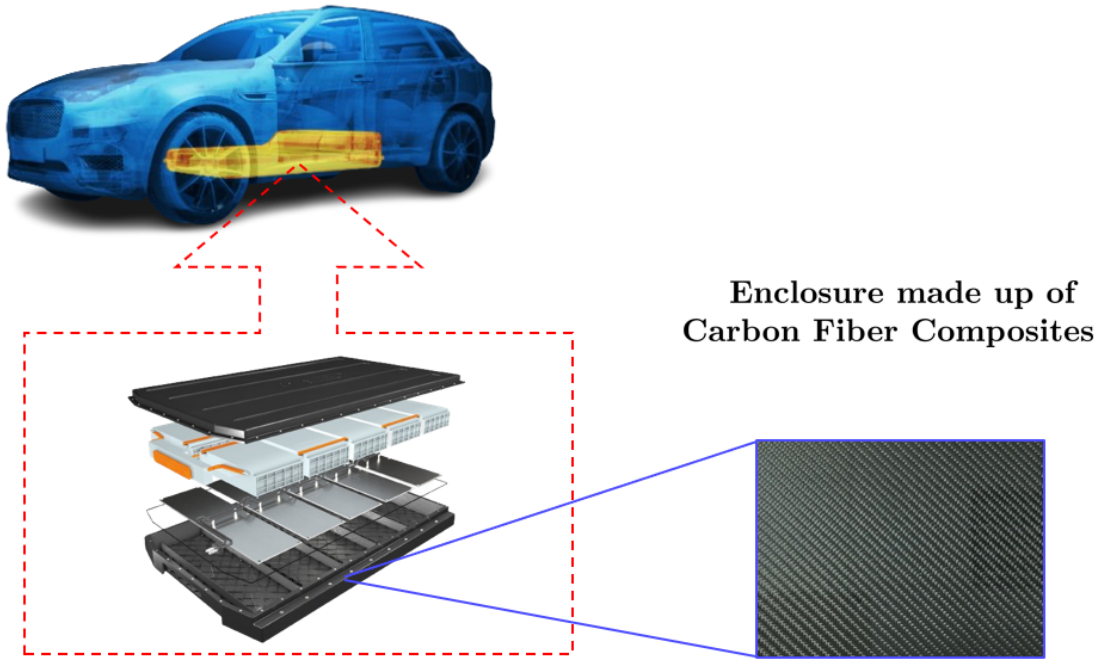


Figure 2.2: Schematic of metal battery enclosure replaced by carbon fiber composites [4] [5]

exhibit superior resistance to corrosion. Unlike aluminum, which may be susceptible to corrosion in certain environments, carbon fiber composites offer excellent corrosion resistance, ensuring the longevity and durability of the enclosure. Furthermore, carbon fiber composites can be engineered to provide additional advantages, such as fire resistance[18] and electrical insulation. These properties are of particular importance in applications where safety and protection against electrical hazards are crucial. By incorporating these engineered properties into the enclosure, carbon fiber composites offer enhanced safety and reliability compared to aluminum, see Figure 2.2.

While carbon fiber composites offer several advantages, they also come with certain challenges that hinder their full realization. One significant challenge associated with carbon fiber composites is their high cost and complex manufacturing process [19] [20]. The pro-

duction of carbon fiber composites requires specialized equipment and expertise, making it a costly exercise. Additionally, the manufacturing process is sensitive to various process and design parameters. Inaccurate selection of these parameters can result in inferior quality parts, compromising the overall performance and safety of the final product. However, the good news is that there are virtual manufacturing simulation softwares that offer potential solutions to these challenges. These software tools allow virtual testing and optimization of manufacturing processes and design parameters prior to physical production.

By leveraging the capabilities of these simulation tools, we design the battery enclosure for improved crashworthiness by virtually manufacturing it by thermoforming simulations and conducting crash simulations to assess its performance under side pole impact conditions. For thermoforming PAM-FORM [21] was employed and for crash simulation VPS [22] was used. Both of these commercial codes are developed, maintained, and marketed by ESI Inc.

The upcoming section discusses a theoretical background on carbon fiber composites, details about thermoforming process, and the material models employed by PAMFORM to simulate composite behavior. Subsequent to this the next section delves into the background of the side pole impact test, accompanied by specifics of the constitutive models utilized by VPS to model the crash behavior of carbon fiber composites.

2.1.3 Thermoforming simulations of composite ply

A carbon fiber composite ply is typically made up of two constituent materials: a reinforcing material, such as fiber, and a matrix material, such as thermoset resin or thermoplastic polymers. These constituent materials are joined by means of interfacial bonding. Typically, the fibers are brittle and exhibit high tensile strength, whereas the matrix contributes to both transverse and compressive strengths of the ply. Additionally, the matrix serves as a medium for transferring load among the fibers and protects fibers from environmental degradation. Furthermore, it absorbs energy by deforming under the application of stress.

A composite laminate is usually built by stacking either of the following two types of composite ply: unidirectional (UD) ply or plain woven ply. A UD ply consists of reinforc-

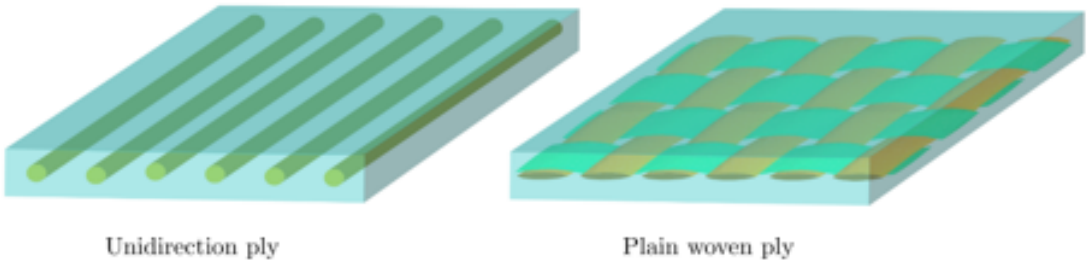


Figure 2.3: Schematic of types of composite ply [6]

ing fibers that are oriented in a single direction within the plane and run parallel to each other, providing a continuous and aligned structure. In plain woven ply, the reinforcing fibers are interlaced together in an alternating over-and-under pattern, forming a grid-like structure. The strength and stiffness of the carbon fiber composite layer can be tailored for specific structural requirements by varying the stacking sequence, thickness, and orientation of individual plies, thus providing flexibility to the designer.

The thermoforming process, as illustrated in figure 2.4, initiates by heating a composite laminate to a specific temperature until it becomes pliable. This pliable sheet is then transferred to a specially designed mold, where pressure is applied to shape it according to the contours of the mold. Subsequent to the molding phase, the shaped composite material undergoes a cooling process, leading to solidification and preservation of the molded shape. Excess material is trimmed, and it further undergoes finishing processes to meet specific requirements.

Simulating the thermoforming process using finite element methods can be challenging owing to the complex interaction between the fiber structure and the matrix during the forming process. To capture these interactions, everything should be explicitly modeled at the meso-scale, i.e., the scale of the fiber, which requires a very fine mesh. However, this fine mesh makes the analysis prohibitively complex, even for the simplest geometry.

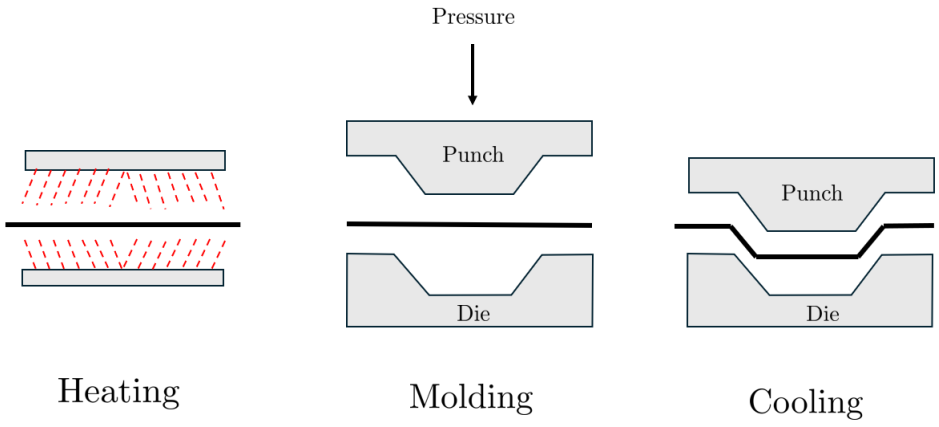


Figure 2.4: Schematic of thermoforming process

To circumvent this difficulty, modern software packages approximate the behavior by using effective properties, i.e. homogenization. As a result, larger mesh sizes can be employed, thereby reducing the computational costs of the simulations.

Traditionally, in the orthotropic material model, the transverse and longitudinal stresses are coupled, meaning that the stress in one direction influences the stress in the perpendicular direction. This coupling between the stresses is given by Poisson's ratio. However, on the other hand, composites are modeled as an othrotropic material, but without considering this interaction between longitudinal and transverse stress [23]. As a result, the stress in a given direction is only dependent on the strain and modulus in that particular direction, as given by equation 2.1.3 [6]. This approximation reduces the complexity without compromising the accuracy of the simulations. Further, this relation between stress, strain and modulus can be given following equation,

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau \\ \sigma_{b1} \\ \sigma_{b2} \end{Bmatrix} = \begin{bmatrix} E_1 & 0 & 0 & 0 & 0 \\ 0 & E_2 & 0 & 0 & 0 \\ 0 & 0 & G & 0 & 0 \\ 0 & 0 & 0 & B_1 & 0 \\ 0 & 0 & 0 & 0 & B_2 \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma \\ z\chi_1 \\ z\chi_2 \end{Bmatrix} \tag{2.1}$$

where, E_1, E_2, B_1, B_2 are the Young's modulus and bending stiffness in direction 1 and 2, as shown in figure 2.5. Likewise, σ_1, σ_2, are then normal stresses, and ε_1, ε_2 are corresponding normal strains in direction 1 and 2. Similarly, σ_{b1}, σ_{b2}, are then bending stresses, and χ_1, χ_1 are corresponding bending curvatures in direction 1 and 2. Additionally, z, corresponds to the coordinates of the layer which is used to transform the bending curvatures in case the fibers in the layers are not aligned to global coordinate axis (**X** - **Y**). Furthermore, τ, γ are shear stress and shear angle respectively.

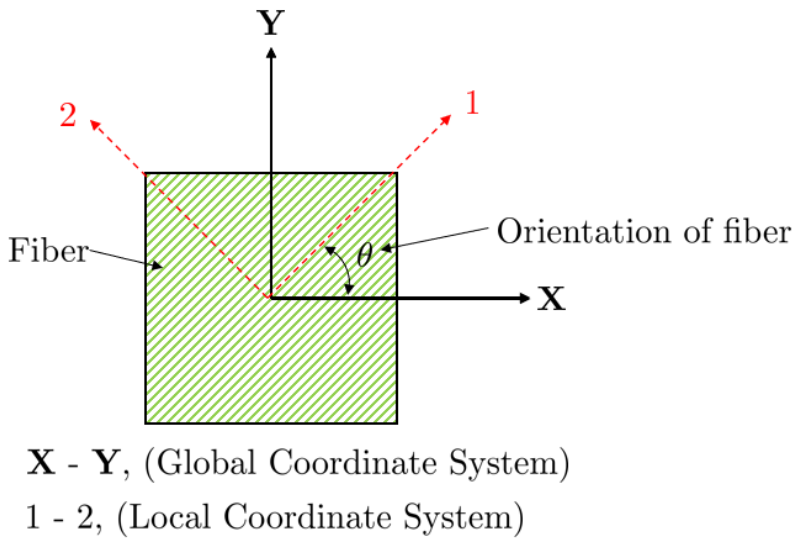


Figure 2.5: Schematic of fiber orientation

Although, decoupled modelling of composite alleviates complexity, additionally, viscoelasicity should be incorporated to model the behaviour of matrix. Hence, the moduli are function of strain, strain rate and temperature and are typically obtained by interpolating stress-strain curves at different strain rate and temperature. In PAMFORM this is calculated employing lookup tables on experimental data [6].

The bending and membrane behaviour of ply is modeled using the Mindlin plate theory [24]. In addition to the conventional displacement degrees of freedom, each node incorporates an additional degree of freedom related to temperature.

2.1.4 Crash simulations of carbon fiber composite

Crash testing plays a crucial role in evaluating the safety performance of automobiles, ensuring their ability to protect occupants in the event of collisions. Several types of crash tests are conducted to evaluate different scenarios and potential hazards, providing valuable insights into a vehicle's crashworthiness and contributing to the enhancement of overall safety standards. Among the commonly conducted crash tests are frontal impact tests, side-impact tests, and rollover tests. Frontal impact tests simulate head-on collisions, side-impact tests assess a vehicle's ability to protect occupants in lateral crash, and rollover tests evaluate stability and occupant protection during rollover events.

The side pole impact test is designed to replicate crash with narrow objects like poles or trees since the real-world side-impact scenarios often involve collisions with such objects. By assessing a vehicle's structural integrity and its capacity to mitigate injury risks during these specific types of accident, the side pole impact test provides essential data to improve overall safety.

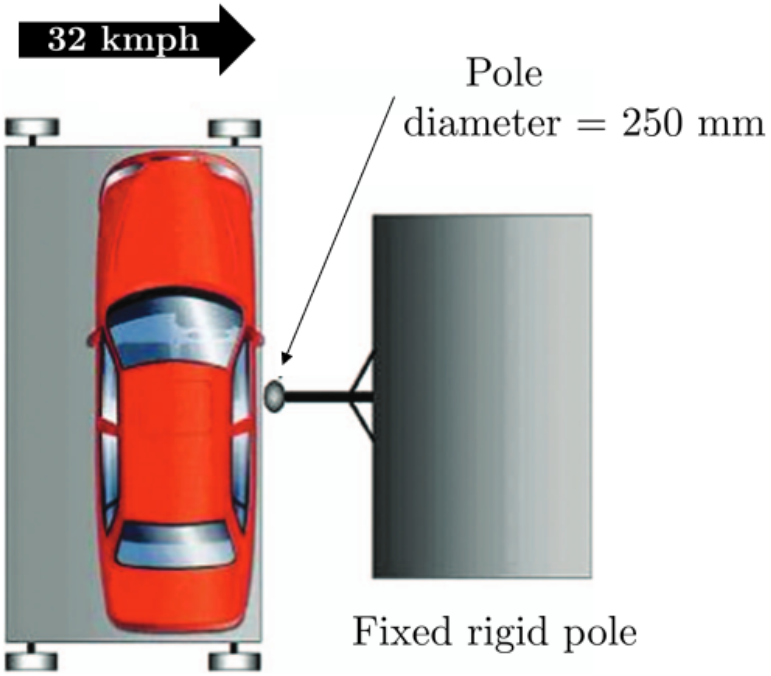


Figure 2.6: Schematic of a side pole impact test [7]

During the test, a vehicle with a dummy is subjected to a controlled collision where it is propelled to a velocity of about 32 kmph (20 mph) such that the side of the vehicle collides with a fixed rigid pole having diameter of about 250 mm [25] [26], shown in Figure 2.6. Data from sensors placed within the vehicle and on crash test dummies are analyzed to evaluate the vehicle's structural integrity, see Figure 2.7, and level of protection provided by the vehicle in such crash scenarios.

For EVs, the side pole impact test holds particular significance owing to their unique design characteristics and potential challenges associated with high-voltage battery systems. EVs typically feature a distinctive structure with batteries positioned along the vehicle's floor, contributing to a lower center of gravity. Although this design enhances stability and handling, it introduces specific considerations for side-impact scenarios.

In this study, ESI's Virtual Performance Solution (VPS) [22] software was utilized to

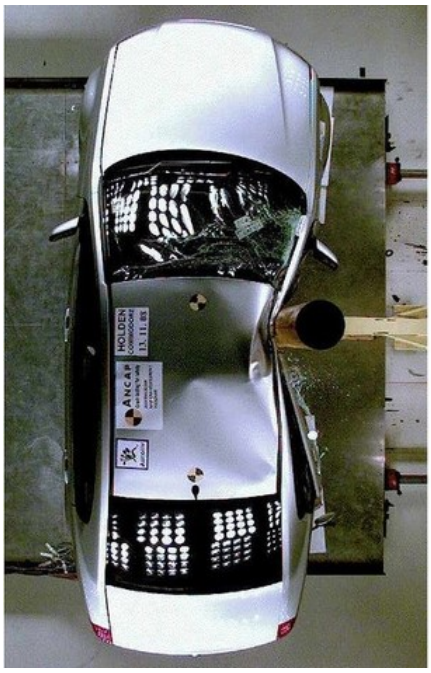


Figure 2.7: Deformation of vehicle after side pole impact test [8]

simulate a side pole impact scenario. VPS was used owing to its advanced material modeling capabilities, notably for composites, which significantly improve the accuracy of simulations. Moreover, its ability to integrate multi-physics analysis within a single platform streamlines the development process, accelerating the design cycle. This comprehensive approach not only enhances the reliability of crash simulations but also facilitates a more efficient workflow.

VPS software offers a 4 noded 2D shell element (MAT 131) suitable for simulating multi-layer carbon fiber plies. MAT 131 is multi-material, multi-layer shell element consisting of one integration point for every layer [24]. Further, the software provides a global ply card for specifying distinct material properties associated to stiffness and strength. The approach for assessing mechanical degradation utilizes a failure model established by Ladeveze and LeDantec [27], encompassing three separate damage evaluation metrics:d_1, d_2, and d_{12}. These functions assess damage levels on a scale from 0 to 1, where 0 indicates no damage and 1 signifies complete failure, as the composite material experiences damage either by fiber rupture or matrix micro-cracking [24] [9]. This increase in the damage function values leads to further deterioration of the mechanical properties of the composite [24], as indicated by

$$[S] = \begin{bmatrix} \frac{1}{E_{11}(1-d_1)} & \frac{\nu_{12}}{E_{11}} & 0 \\ \frac{\nu_{12}}{E_{22}} & \frac{1}{E_{22}(1-d_2)} & 0 \\ 0 & 0 & \frac{1}{G_{12}(1-d_{12})} \end{bmatrix} \tag{2.2}$$

where $[S]$ denotes the elastic compliance matrix, and E_{11}, E_{22}, G_{12} and ν_{12} are the four critical material parameters. To accurately capture the composite material's response, a logarithmic damage function is utilized. Further, the evolution of plastic strain adheres to the generalized hardening function equations [28]. The total strain at any given point is given as the sum of the elastic and plastic strain components.

2.1.5 Issues with current simulation work-flow for enclosure design

The current simulation workflow, shown in figure 2.8, starts with a thermoforming simulation to virtually manufacture the battery enclosures. Subsequently, crash simulation to simulate a side-pole impact test on the formed part, finally extracting crucial crash parameters.

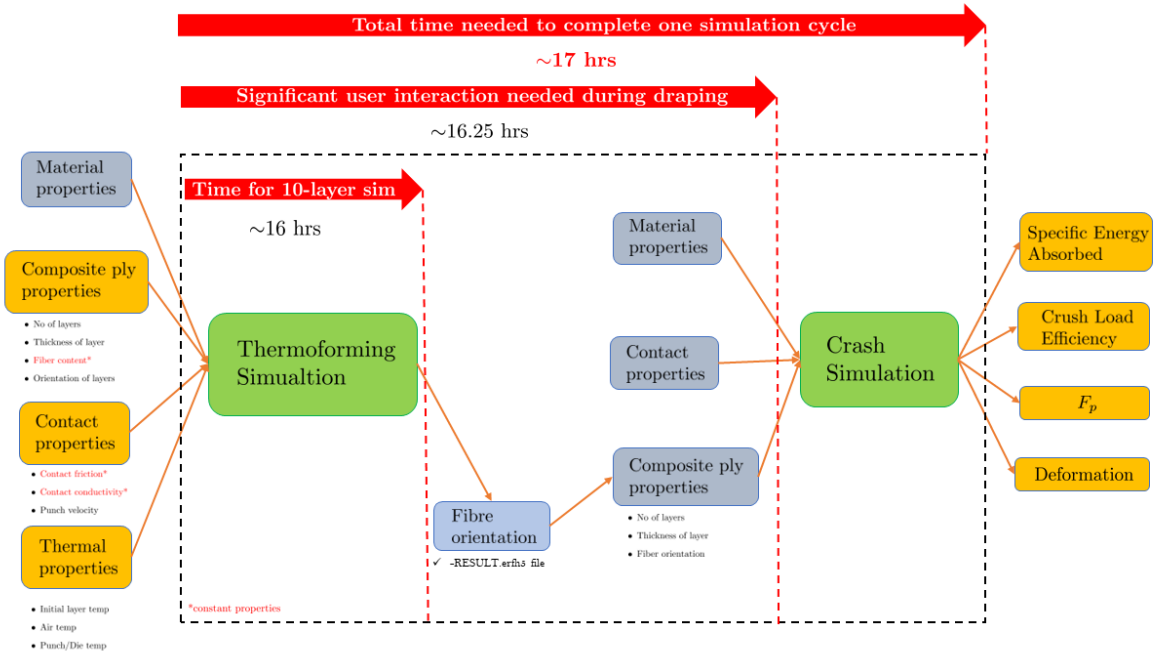


Figure 2.8: Current finite element simulation work-flow

Nonetheless, the issue with the current workflow is the time taken to complete each step. Firstly, the thermoforming simulation of 10 layer composite laminate for example, takes an average time of approximately 16 hours on a 16 cores HPC unit. Subsequently, the crash simulation, takes around average 45 minutes on a 8 core 11^{th} Gen Intel (R) i7@3GHz PC. In addition, to link these two simulations together the results from thermoforming should be manually provided to the crash simulation. This step requires significant user interaction. It was found that the total time taken to complete one full simulation run was approximately around 17 hours with significant user interaction needed in the whole process. This issue with the simulation workflow's time requirements makes the decision process extremely challenging and emphasizes the need for a surrogate model to cut the overall simulation time, illustrated in figure 2.9.

Figure 2.9: Simulation workflow replaced by black-box reduced order model

The subsequent section discusses theory and mathematical underpinnings of surrogate modeling using machine learning techniques.

2.1.6 Surrogate modeling using ML

From a mathematical standpoint, a computational model can be conceptualized as a "black box" function, denoted as $\mathcal{M}$, which maps the input parameters $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ of the system to the outputs $\mathbf{Y} \in \mathbb{R}^{n \times d_2}$. This mapping is represented by the equation:

$$\mathbf{Y} = \mathcal{M}(\mathbf{X}) \tag{2.3}$$

where, $\mathcal{M} : \mathbf{X} \mapsto \mathbf{Y}$, with d_1 and d_2 being the dimensions of the input and output spaces, respectively.

In this study, $\mathcal{M}$, in an abstract sense, represents a finite element simulation for battery design. The input to the models are material parameters such as the number of composite layers (n_{ls}), thickness of each composite layer (t_l), fiber orientation ($\mathbf{\Phi}_{fib}$), and process parameters for the thermoforming process, including the velocity of the punch (v_p), initial temperatures of the composite layer (T_i), punch/die (T_{pd}), and air (T_{air}).

Typically, the finite element model predicts the output such as crash parameters such as specific energy absorption (SEA), crush load efficiency (CLE), peak force (F_p), and intrusion of the side pole inside the battery enclosure after impact ($\Delta \mathrm{Y}_{\mathrm{node}}$).

Conventionally in surrogate modeling, the objective is to develop an approximate function, i.e., a simulator $\hat{\mathcal{M}}$, that mimics the behavior of the actual function $\mathcal{M}$. This simulator is constructed by executing the actual model $\mathcal{M}$ on a collection of input parameters $\mathbf{X} = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\right]^{\mathrm{T}} \in \mathbb{R}^{N \times 10}$, where N is the number of runs of the computational model (i.e., the number of finite element simulations), and 10 is the dimension of $\mathbf{x}$ where, $\mathbf{x} = [n_{ls}, t_l, v_p, T_i, T_{pd}, T_{air}, \mathbf{\Phi}_{fib}]$, with $\mathbf{\Phi}_{fib}$ being a vector of size 4. These input parameters are typically sampled on using sampling schemes such as Latin Hypercube sampling, quasi-random sampling owing to their space filling properties. Finally, the predicted output of the model is collected in the matrix $\mathbf{Y}$, where $\mathbf{Y} = \left[\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(N)}\right]^{\mathrm{T}} \in \mathbb{R}^{N \times 4}$, with 4 being the dimension of $\mathbf{y}$, defined as $\mathbf{y} = [\mathrm{SEA}, \mathrm{CLE}, F_p, \Delta \mathrm{Y}_{\mathrm{node}}]$ where SEA, CLE stands

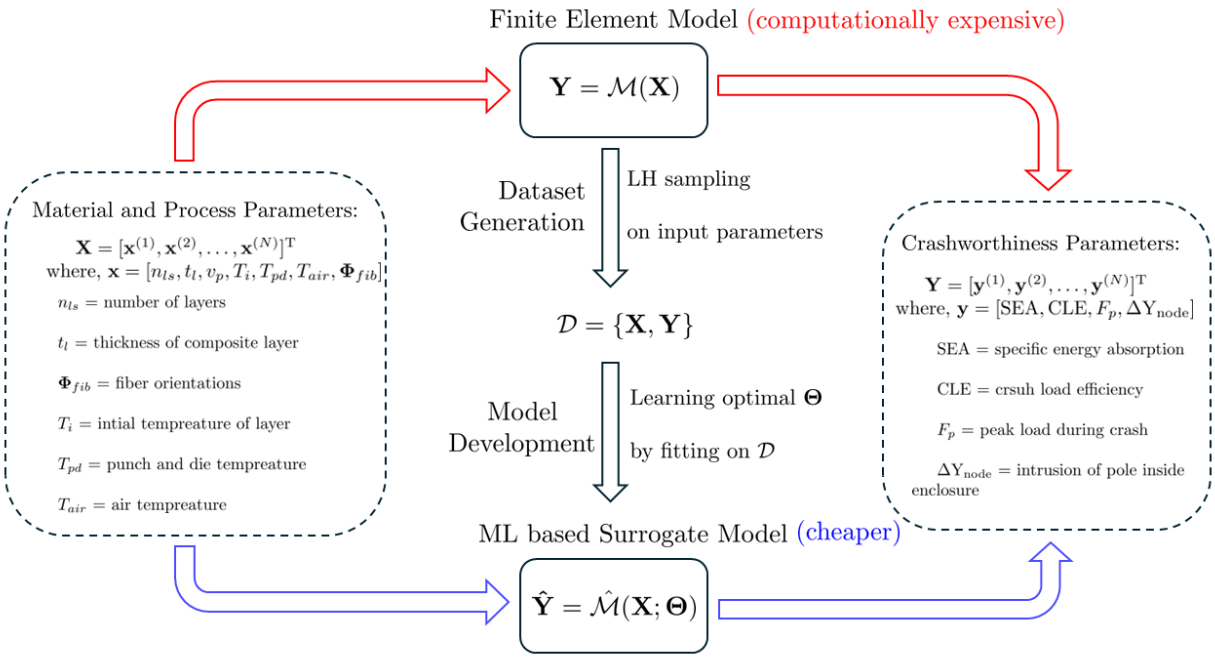


Figure 2.10: Schematic of a ML based surrogate model

for specific energy absorbed and crush load efficiency respectively. Further, F_p is peak load experienced by enclosure during impact and ΔY_{node} is the intrusion of side pole into the enclosure after impact. The crash parameter are discussed in detail in 2.3.1.4. Finally, the inputs and outputs are assembled together to form a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$.

This process can be succinctly expressed by the equation:

$$\hat{\mathbf{Y}} = \hat{\mathcal{M}}(\mathbf{X}; \mathbf{\Theta}) \tag{2.4}$$

where, $\mathbf{\Theta}$ represents the collection of model parameters. The primary objective is to find optimal $\mathbf{\Theta}$ by utilizing dataset $\mathcal{D}$ and optimization strategies such that $\hat{\mathbf{Y}} \approx \mathbf{Y}$. However, the complexity of this task is often heightened by the dimensionality of inputs (also known as input features) and outputs (also known as response). Additionally, the model should not only closely fit the given data but should also capture the underlying patterns and relationships inherent in the data, enabling the model to generalize well to unseen data.

Additionally, the model $\hat{\mathcal{M}}$ can represent any ML algorithm including but not limited to

linear regression, neural networks, support vector machines (SVM), and Gaussian processes regression (GPR). In the context of this study, our approach begins with the application of linear regression, progressing subsequently to Gaussian processes due to their inherent capacity to effectively capture non-linear patterns within the dataset. A comprehensive discussion of each model is provided in the subsequent subsections. Further, for a deeper dive on surrogate modeling concepts, and theoretical background reader's are encouraged to consult the "Engineering design via surrogate modelling: a practical guide" by Alexander Forrester, Andras Sobester and Andy Keane [29]

2.1.6.1 Linear regression

Linear regression is a foundational statistical method aimed at modeling the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. Building upon the mapping function defined in equation 2.4, linear regression can be perceived as a specific instance of this mapping. In this instance, the model takes the form of a linear combination involving the input features $\mathbf{X}$, with model parameters denoted as $\mathbf{\Theta} = \{\mathbf{W}, \mathbf{b}\}$ representing the weights assigned to these features, including the intercept. Mathematically, this relationship is expressed as:

$$\mathbf{Y} = \mathbf{X}\mathbf{\Theta} \tag{2.5}$$

The primary objective of linear regression is to determine the optimal parameter vector $\mathbf{\Theta}^*$ that minimizes the difference between the predicted outputs $\hat{\mathbf{Y}}_{\text{pred}}$ and the true outputs $\mathbf{Y}$. This discrepancy is commonly quantified using a cost function such as Mean Squared Error (MSE), given by:

$$\text{MSE} = \frac{1}{n}||\mathbf{Y} - \hat{\mathbf{Y}}_{\text{pred}}||_2^2 \tag{2.6}$$

As the number of input features increases, the complexity of the model becomes important. A higher number of features may lead to overfitting, where the model excels on the training

data but performs poorly on unseen data. To address this, regularization techniques like LASSO (L1 regularization) and Ridge (L2 regularization) are frequently employed.

For Lasso regression, the cost function, denoted as $\text{MSE}_{\text{LASSO}}$, incorporates a penalty term based on the absolute values of the weights:

$$\text{MSE}_{\text{LASSO}} = \frac{1}{n}||\mathbf{Y} - \hat{\mathbf{Y}}_{\text{pred}}||_2^2 + \lambda||\mathbf{W}||_1 \tag{2.7}$$

Similarly, for Ridge regression, the cost function $\text{MSE}_{\text{RIDGE}}$ includes a penalty term based on the squared values of the weights:

$$\text{MSE}_{\text{Ridge}} = \frac{1}{n}||\mathbf{Y} - \hat{\mathbf{Y}}_{\text{pred}}||_2^2 + \lambda||\mathbf{W}||_2^2 \tag{2.8}$$

Here, λ in both cases serves as a regularization parameter, constraining the magnitude of the cost function of model parameters. Ultimately, the goal in linear regression, as with any predictive modeling technique, is to learn a set of parameters $\boldsymbol{\Theta}^*$ that enables the model to make accurate predictions on new, unseen data, thereby demonstrating the model's ability to generalize. This process necessitates careful consideration of the model's complexity, the quality and quantity of the data, and the chosen methodology for evaluating model performance.

2.1.6.2 Gaussian Process Regression (GPR)

GPR stands out as a robust surrogate modeling technique that has gained significant popularity recently. Notably, GPR excels due to its adaptability to model complex relationships leveraging its non-parametric nature that can fit a wide range of functions on given input points. Further, in quantifying uncertainty in predictions, addressing both intrinsic noise in the problem and errors in the parameter estimation process. Furthermore, GPR provides a natural framework for introducing kernels into regression models, enabling the exploitation of structural patterns in the data through careful kernel selection [30] [31].

This section aims to provide a concise introduction to GPR, supported by its mathematical

underpinnings. For a more in-depth understanding of GPR concepts, readers are encouraged to delve into the comprehensive treatment provided in "Gaussian Processes for Regression" by Christopher Williams and Carl Rasmussen [32].

From a mathematical standpoint, Gaussian process is a random process $f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^d$, with d being dimension of the input space, such that any finite subset of these inputs points given by $\mathbf{X} = \left[\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right]^{\mathrm{T}} \in \mathbb{R}^{n \times d}$ and their corresponding output $f(\mathbf{X}) = \left[f\left(\mathbf{x}^{(1)}\right), \ldots, f\left(\mathbf{x}^{(n)}\right)\right]^{\mathrm{T}} \in \mathbb{R}^n$ is also a joint Gaussian distribution.

Gaussian process regression starts with a prior of the unknown function given by $f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$ which is fully characterized by a mean function $m(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$, that defines the prior mean of f for certain input points $\mathbf{x}$, given by;

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.9}$$

and covariance function $k\left(\mathbf{x}, \mathbf{x}'\right) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ which is also known as kernel of GPR, that basically captures the linear dependence of function values at two input points $\mathbf{x}$ and $\mathbf{x}'$ and given by

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f\left(\mathbf{x}'\right) - m\left(\mathbf{x}'\right)\right)\right]. \tag{2.10}$$

To make posterior calculations manageable the prior mean $m(\mathbf{x})$ is often set to zero at all points i.e. $m(\mathbf{x}) = 0$. In this case the covariance function defines the shape of the function sampled from the prior.

Most commonly used kernel (i.e. covariance function), is the radial basis function kernel (a.k.a squared exponential kernel) given by:

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{x}'\right\|^2}{2l^2}\right) \tag{2.11}$$

where σ_f and l are the are two hyper-parameters of kernel (or more specifically the GPR model), known as the signal variance and the length-scale respectively. Additionally, this is

special case of general class of kernels called Matérn kernels given by:

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\left\|\mathbf{x} - \mathbf{x}'\right\|}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\left\|\mathbf{x} - \mathbf{x}'\right\|}{l} \right) \tag{2.12}$$

where ν controls the smoothness, Γ is the gamma function, and K_ν is a modified Bessel function. As $\nu \to \infty$ this become equivalent to the radial basis kernel.

The kernel in Eq. 2.11 uses same length-scale across all input dimensions d. Another approach involves using a different length-scale l_i for each input dimension x_i, known as automatic relevance determination (ARD), given by:

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{i=1}^{d} \frac{\left(x_i - x_i'\right)^2}{l_i^2} \right) \tag{2.13}$$

where the kernel parameters are the length scales $l_1, \ldots, l_d$ and the signal amplitude, σ_f. The value of the length-scale determines the relevance of each input feature to the GPR model. This kernel is also known as anisotropic variant of the squared exponential kernel.

In GPR the task is to infer relationship between the inputs $\mathbf{X}$ and target $\mathbf{y}$ i.e. $\mathbf{y} = \left[y^{(1)}, \ldots, y^{(n)}\right]^{\mathrm{T}} \in \mathbb{R}^n$ where n is number of noisy observations, assembled as a dataset given by $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. Following the definition of Gaussian process, the value of functions at the training inputs $\mathbf{X}$ and new unseen input points $\mathbf{X}^*$ i.e. $\mathbf{f}^*$ are jointly Gaussian, written as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}^*} \\ \mathbf{K}_{\mathbf{X}^*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^*,\mathbf{X}^*} \end{bmatrix} \right). \tag{2.14}$$

where $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ is the covariance matrix between the training points, $\mathbf{K}_{\mathbf{X},\mathbf{X}^*}$ and $\mathbf{K}_{\mathbf{X}^*,\mathbf{X}}$ are the covariance matrices between the training points and new points (also known as cross-covariance matrix), $\mathbf{K}_{\mathbf{X}^*,\mathbf{X}^*}$ is the covariance matrix between the new points and σ_n^2 represents variance of an additive independent and identically distributed (i.i.d.) Gaussian noise with zero mean.

The predictions can be made by drawing the samples from the posterior distribution,

which is also multivariate Gaussian by definition [32], with posterior mean given by

$$\text{mean}(\mathbf{f}^*) = \mathbf{K}_{\mathbf{X},\mathbf{X}^*}^{\mathrm{T}} \left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}, \tag{2.15}$$

and covariance

$$\text{cov}\left(\mathbf{f}^*\right) = \mathbf{K}_{\mathbf{X}^*,\mathbf{X}^*} - \mathbf{K}_{\mathbf{X},\mathbf{X}^*}^{\mathrm{T}} \left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{K}_{\mathbf{X},\mathbf{X}^*}. \tag{2.16}$$

An important point to note here is calculating mean and covariance using equations 2.15 and 2.16 involve inverting the matrix which may be often large and ill-conditioned. Further, to alleviate these issues oftentimes a small regularization term is on the diagonal elements of the matrix. Further, σ_n^2 in the equations also serves as a regularization term.

In GPR model, training implies estimating unknown hyper-parameters (i.e. optimal model parameters) $\Theta = [l, \sigma_f, \sigma_n]^{\mathrm{T}}$, where l, σ_f, and σ_n are length scale, signal amplitude and noise variance respectively, typically involve maximizing log marginal likelihood given by [33] :

$$\log p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\Theta}\right) = -\frac{1}{2}(\mathbf{y}^{\mathrm{T}} \left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}) - \frac{1}{2} \log \left|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_n^2 \mathbf{I}\right| - \frac{N}{2} \log(2\pi). \tag{2.17}$$

In this equation the first term quantifies how well the model fits the data, second term quantifies model complexity and it added to prefer simpler over complex during training.

2.1.7 Monte Carlo(MC) error propagation

Monte Carlo error propagation is a powerful technique used to estimate the uncertainty of a system's output based on the uncertainties in its input parameters. This method is particularly useful in complex systems where analytical solutions for error propagation are impractical or impossible. The Monte Carlo approach involves generating a large number of possible scenarios for the input parameters, each within their respective probability distributions, and then running simulations to observe how these variations affect the output. With large enough number of input samples the MC method can provide a complete description of the statistical behavior of the system [34].

The basic steps in MC error propagation include defining the probability distributions for all uncertain inputs, generating a large number of random samples from these distributions, further running the model on these input samples and evaluating the system's output for each set of samples, and finally analyzing the distribution of the outputs to assess the uncertainty.

As MC method involves running large number samples, utilizing computational model (i.e. finite element simulation) $\mathcal{M}$ discussed in section 2.1.6 is infeasible. However, to alleviate this issue the surrogate model $\hat{\mathcal{M}}$ can be employed [35].

Mathematically, MC method can be explained as:

$$\mathbf{y}_i = \hat{\mathcal{M}}(\mathbf{x}_i), \quad i = 1, 2, \ldots, N \tag{2.18}$$

where, $\mathbf{x}_i$ are random samples drawn from the probability distributions of the input parameters, and N is the total number of simulations. The output of each simulation, $\mathbf{y}_i$, constitutes one realization of the model's output given the stochastic nature of the inputs

The uncertainty in the model output can then be quantified using statistical measures computed from the ensemble of $\mathbf{y}_i$. The mean μ and standard deviation σ of the model output are calculated as follows:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i, \tag{2.19}$$

$$\sigma = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{y}_i - \mu)^2. \tag{2.20}$$

These measures provide an estimate of the expected output and its variability due to the input uncertainties. The MC method does not require any assumptions about the linearity or non-linearity of the model $\hat{\mathcal{M}}$, making it a versatile tool for error propagation in a wide range of applications.

Furthermore, MC error propagation can be extended to evaluate the sensitivity of the model output to each input parameter. By analyzing the variation in outputs $\mathbf{y}_i$ with respect to changes in each input parameter $\mathbf{x}_i$, one can identify which inputs have the most significant

impact on the output uncertainty. This sensitivity analysis is crucial for prioritizing efforts in reducing input uncertainties or for model simplification.

2.2 Literature Review

Over the last several decades, there has been a global push towards adopting EVs as a viable alternative for vehicles powered by petroleum due to their potential benefits. However, there are two main challenges that hinder the full exploitation of EVs. Firstly, EVs tend to be heavier than their fossil fuel counterparts, largely because of the significant weight added by battery packs and their enclosures, which require a reinforced structure and suspension system [36]. The second major concern is the risk of fire hazards if the battery pack gets damaged in an accident, leading to potential thermal runaway, fires, and explosions [37].

A proposed solution to mitigate the weight issue is the utilization of carbon fiber reinforced polymer (CFRP) composites for the construction of the battery enclosure. CFRP composites offer superior strength and stiffness, which is why they are increasingly replacing traditional aluminum alloys in aircraft components, now constituting a major portion of the materials used in modern aircraft [38]. This trend is similarly observed in the automotive industry, with a growing adoption of composites [39], and efforts are being made to explore battery enclosure designs that incorporate these materials [40].

Choosing CFRP for battery enclosures seems beneficial solely from a weight reduction perspective. Nevertheless, considering fire safety is essential before such materials can be broadly accepted for this application. Although CFRP materials are inherently resistant to fire and corrosion [18], the primary fire risk originates within the battery pack itself, particularly if it is damaged [37]. Therefore, the effectiveness of the enclosure in protecting the battery pack from damage, especially considering its typical mounting position at the bottom of the vehicle, which is susceptible to impacts from accidents, must be thoroughly evaluated [41][37][42]. This study focuses particularly on side impacts during crashes, as damage to the battery pack has been well documented [43], but the protective capacity of an enclosure during such events has received less attention. There has been few studies

investigating the battery enclosure design [44][45].

The idea of employing CFRPs to create enclosures capable of withstanding crashes presents several benefits as well as potential challenges. Developing a CFRP enclosure requires determining various design and manufacturing process parameters, collectively known as design variables [9]. These include the number of layers, fiber orientation, and stacking sequence, along with process parameters such as punch velocity and the temperature of the layers, assuming a thermoforming manufacturing process is used. With numerous parameters to adjust, there is potential to tailor the enclosure's properties for enhanced crash performance. However, the challenge lies in understanding how adjustments to each parameter influence overall crashworthiness and in pinpointing the ideal configurations. Finding optimal parameters through exhaustive experimentation and trial and error can be extremely difficult, time-consuming, and ineffective. Fortunately, several physics-based simulation tools are available to assist in this process.

ESI Inc. offers two specialized software programs, PAMFORM and VPS, which are commercially available and employ FEM for simulating the thermoforming process and performing structural analysis on carbon fiber composites, respectively. PAMFORM is particularly effective as it accounts for the dynamics of fiber-matrix interactions during the thermoforming process [46]. It also allows for high-throughput thermoforming simulations by varying the process parameters related to composite materials. Similarly, VPS possesses the ability to accurately model the complex damage behaviors observed in composites during impacts [47]. Moreover, ESI offers functionality to chain these software programs to streamline the analysis. PAMFORM can be used to virtually manufacture CFRP battery enclosures by the thermoforming process, and VPS can be used for performing side pole impact crash simulations on the formed CFRP enclosure [9]. Utilizing VPS alongside PAMFORM simplifies the process of integrating thermoforming results into crash simulations, thereby streamlining the pipeline.

Nonetheless, finding optimal parameters remains a challenge even with these simulation

tools, as they are often computationally expensive. To address this, physics-based simulations should be combined with data-driven techniques, which can evaluate every potential combination of design variables, thereby providing optimal parameters. This strategy could also be classified as a process-property modeling task, often encountered in the field of Integrated Computational Materials Engineering (ICME) [48]

Over the past few years, machine learning has made significant strides in dealing with various data types, including images, audio, and text, with notable achievements documented in the literature [49][50][51]. Moreover, deep learning algorithms have been extended to address the challenges associated with tabular data [52][53][54][55]. However, deep learning models face unique difficulties when applied to tabular data, which typically features heterogeneous and non-sequential attributes. This is in contrast to image data, which contains spatial hierarchies and local correlations that these models can effectively leverage [56]. Consequently, tree-based models, which naturally handle feature interactions and nonlinear relationships, often perform better in tabular data scenarios [57][58].

In addition to advances in machine learning for handling conventional data types, recent years have also seen substantial progress in applying these techniques across various domains of material science [59][60][61][62]. These methods offer promising paths to accelerate the design space exploration process, enabling designers to efficiently navigate design spaces. Among these, Gaussian Processes (GPs) emerges as a powerful tool, for its ability to model complex relationships and providing predictions with associated uncertainty estimate [32] [30].

GPs have been successfully applied to solve a variety of problems in manufacturing and material science. Hoffer et al.[63] employed GPs to develop a surrogate model for FEM simulations of the forging process for superalloys. Tapia et.al. [64] used GPR to identify optimal process parameters for Laser Powder Bed Fusion (L-PBF) additive manufacturing of 316L steel. Similarly, Zhou et al.[65] developed an adaptive surrogate model based on Gaussian Processes for the optimization of process parameters in injection molding sim-

ulations. Furthermore, Radaideh and Kozlowski[66] created a GPR surrogate for nuclear reactor simulations, further performing uncertainty analysis. Saunders et al.[67] developed a functional GP surrogate for predicting the mechanical behavior of additively manufactured micro-structures. Noack et al.[68] proposed a GPR surrogate with anisotropic kernels and non i.i.d. measurement noise for automated material discovery. Further, Chen et. al.[69] utilized GPR to model the constitutive relations of materials for stochastic structural analysis.

A plethora of studies have explored the application of computational techniques and machine learning for the design and optimization of composite materials [70][71][72][73]. Zhang et al. [74] proposed a coupled FEA and ML technique, utilizing Neural Networks (NN) and Random Forest (RF), to predict mechanical properties of composite laminates. In [75], the authors explored the fracture mechanics of laminated composites through experimental methods, FEA, and ML algorithms, focusing on Mode I, Mode I/II, and Mode II loading situations. This study compares the effectiveness of FEA and ML against experimental results in analyzing the fracture behavior of composites. The authors in [76] investigated the efficacy of three ensemble machine learning models for predicting the strain enhancement ratio of fiber-reinforced polymer composites, utilizing a comprehensive dataset of 729 experiments compiled from existing literature.

Despite the extensive application of FEA and ML in the design and manufacturing of composites, their utility in expediting the design of composite battery enclosures for crashworthiness has not been thoroughly explored. In response to this gap, the initial part of this study focused on generating a dataset through time and resource intensive high-throughput simulations, based on the workflow established by Kulkarni and Hale et al. [9]. Following this, tree-based ensemble methods were developed by fitting them on the generated dataset [1].

This work focuses on development of probabilistic surrogate model based on GPs that can provide not only predictions, but also the associated uncertainty estimates. The surrogate

model can be employed to expedite design space exploration studies, offering a promising avenue for rapidly optimizing the design of composite structures to meet specific performance criteria.

2.3 Methodology

This section delves into the details of the data generation phase and, subsequently, the model development phase. Each of these are discussed in more detail in the following subsections

2.3.1 Data generation for surrogate model

In the data generation phase, several finite element simulations were performed utilizing various composite material properties and processing conditions. The range of these parameters was fixed keeping in mind the best practices and realistic thermoforming process conditions. These simulations were performed on high-performance computing (HPC) nodes owing to their computationally intensive nature. Finally, the results were extracted and post-processed to form a dataset which was later used to train the machine learning model. Subsequent subsections discuss each of these steps in detail

2.3.1.1 Creation of design of experiment (DOE) matrix

The design variables, encapsulating both, the composite properties and processing conditions, are detailed in Table 2.1. These variables were selected by considering both commonly used material properties and realistic processing conditions in actual thermoforming processes.

Following these ranges, approximately 400 data points were sampled using Latin Hypercube sampling (LHS) to ensure uniform coverage of the design space, owing to its space-filling properties [77, 78]. To perform LHS, the pyDOE [79] Python library was utilized. The number of input variables and the desired number of samples were specified, resulting in the LHS matrix. Finally, the DOE matrix was constructed by scaling the LHS matrix to fit the specified ranges.

Table 2.1: Range used for creating DOE matrix

Design Variables	**Range**
Number of layers, n_{ls}	4 - 16, only even
Thickness of each layer, t_l	0.1 - 0.6 mm
Fiber orientations, $\mathbf{\Phi}_{fib}$	(0, 45, -45, 90), (30, -30, 60, -60)
Punch velocity, v_p	4 - 6.5 m/s
Layer initial temperature, T_i	200 - 400 °C
Punch / Die temperature, T_{pd}	20 - 220 °C
Air temperature, T_{air}	10 - 30 °C

2.3.1.2 Finite element simulations setup

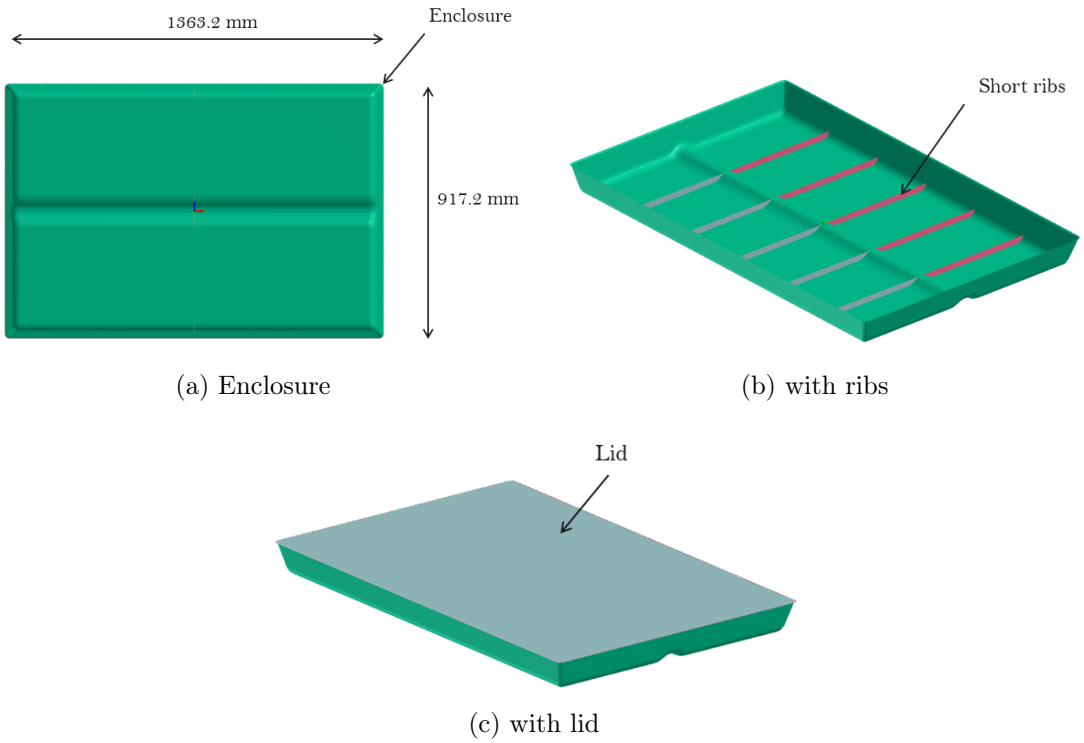
In this study, a single simulation cycle consisted of a chain of two finite element simulations, where first the enclosure was virtual manufactured and later the as formed enclosure was virtually crash tested. For manufacturing, thermoforming simulation was employed using PAM-FORM, similarly, for simulating the side pole impact test VPS was utilized. The theoretical background of each of these simulations is discussed in section 2.1.3. The following subsections provide details about the simulation setup

Geometry of battery enclosure used in the simulations

The geometry of the battery enclosure, as shown in the figure 2.11, was utilized in the subsequent simulation setups. The dimensions of the enclosure were set based on dimensions of an actual battery module [80, 81, 9]. However, to make further analysis more manageable, several simplifications were incorporated.

Firstly, no internal or external mounting was considered. Nevertheless, a nonstructural mass of approximately 100 kg was added to account for the battery module and emulate the realistic weight of a battery pack. Additionally, for added support, the enclosure was internally reinforced with ribs, as depicted in the figure as shown in the figure 2.11 (b). These ribs also serve as demarcations between different battery module components in actual battery packs.

Furthermore, a 15-degree relief angle was introduced at the ends of the enclosure to facil-



(a) Enclosure (b) with ribs

(c) with lid

Figure 2.11: Geometry of battery enclosure used in the analysis [9]

itate the easy removal of the punch during the thermoforming manufacturing process. The entire assembly was covered by a lid, figure 2.11 (c). Both the lids and ribs were assumed to be made of elasto-plastic material with material properties detailed in Appendix Table A.3.

Thermoforming simulation

Performing thermoforming simulation, shown in Figure 2.12, consists of two distinct stages. In the first stage, the composite sheets are heated to a temperature conducive to forming, similarly, the punch and die are also kept at specific temperatures. In the second stage, the punch is given a downward stroke with a certain velocity that should facilitate forming. During this stage, the composite sheet is pressed against the die by the punch leading to severe shearing and deformation of the sheet. This forming stroke is continued further until the composite sheet completely takes the shape of the die.

In this work, several simulations with different input parameters, listed in Table 2.1, were performed. However, for the sake of brevity, the setup of just one simulation sample with 10

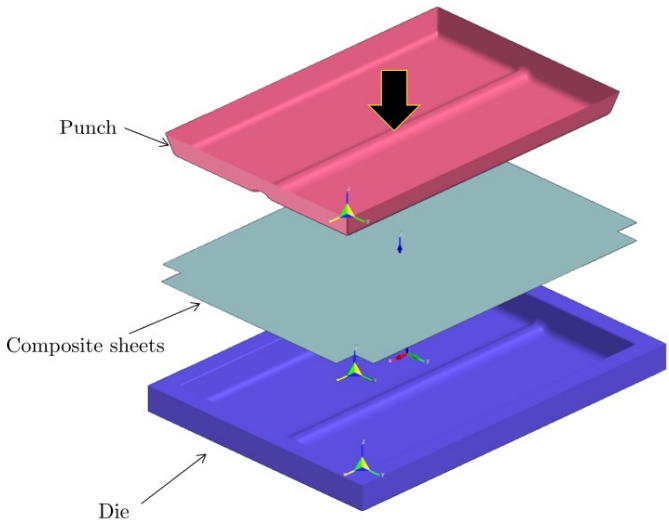


Figure 2.12: Schematic for thermoforming simulation [9]

number of composite layer and its corresponding process parameters.

Following the dimensions of the enclosure, the geometry of the die was created. Additionally, to facilitate the easy removal of punch after completion of forming stroke, a relief angle of 15 degrees was provided onto the die. A custom outline of the ply with cutout relief — to resolve the convergence issue — was created. The dimensions of the cutout were decided using the following formula :

$$c^2 = \frac{S_p - A_s}{4}, \tag{2.21}$$

where c is the length of cutout, S_p is the outer surface area of punch, A_s is the area of composite sheet. It was found that the square cutout of size 105 $\times$ 105 mm for sheet of size 1500 $\times$ 1052 mm didn't lead to any further convergence issues.

Later, geometry of the die and outline of ply were imported into software. Further, within the software, the punch geometry was created by replicating the die geometry and offsetting it at a distance equivalent to the total thickness of the composite sheet along the z direction. For this specific simulation setup, the layer thickness of 0.25 mm was used leading to an offset distance of 1 mm.

Finally, the composite layer was created from the outline of the ply and was later meshed

with a 4-noded 2D shell element with a mesh size of about 3 mm. Additionally, the material properties of the ply were applied using the inbuilt software material database [6]. Moreover, same type of shell element and mesh size was used all other thermoforming simulations.

The contact between all the composite plies and in between the die/punch and composite layer was defined using "friction penalty". Further, the coefficient of friction for this friction penalty was calculated by interpolation of viscosity curves of the resin material which is a function of shear rate and temperature. In PAMFORM, this functionality is implemented with the use of look-up tables on the experimental data [6]. The thermal conductivity and layer separation stress of value 0.45 W/m. K and 0.005 GPa respectively were used.

Both punch and die were considered rigid and were kept at a temperature 153 °C with the die locked at the origin of the global reference axis, and the punch allowed to move along the z-direction with a velocity of 5 m/s. The composite layer was kept at an initial temperature 255 °C and air temperature was maintained at 18 °C Further, the composite sheet was pressed against the die with a downward motion of punch until the sheets completely took the shape of the die and the simulation was terminated. The simulation is said to be complete when the solver terminates without any error.

In the thermoforming process, composite sheets undergo significant deformation, resulting in the distortion of fiber orientations at specific locations. These alterations in fiber orientations plays an important role in determining the overall performance of the formed part. Further recognizing their significance, it is crucial to accurately incorporate these changes into the subsequent simulation phase, thereby ensuring the completeness of the simulation chain. This critical transition is known as draping, functioning as a link between the deformed state of the composite sheet after thermoforming and the next simulation phase i.e. side pole impact crash simulation.

Side pole impact crash simulation

A schematic of side pole impact crash simulation is shown in Figure 2.13. Initially, the geometry was imported and meshed using VPS's shell MAT131, a 4-noded 2D shell element

with one integration point, employing a mesh size of 5 mm. Subsequently, the result file from the thermoforming simulation, denoted by the extension ".erfh5", was imported into the software. The enclosure geometry was draped with this result file to update the material model, incorporating composite properties of the formed part.

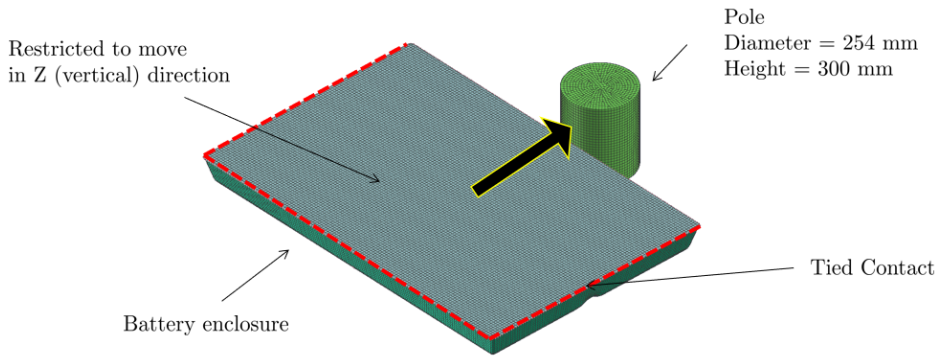


Figure 2.13: Schematic for crash simulation [9]

Within the VPS, a stationary pole following the dimensions employed in actual side impact crash testing was created. In addition, ribs and a lid were created within the software and connected to the enclosure geometry using a tied contact. This ensures that the entire enclosure assembly functions as a unified entity throughout the impact. In terms of material properties, the pole was treated as rigid, while the ribs and lid were characterized as elasto-plastic. The material properties for the rib, lid, and enclosure used in the simulations are listed in the table in the Appendix.

To emulate realistic side pole impact conditions, the enclosure assembly was propelled toward the stationary rigid pole with a velocity of 32 km/h. The contact interaction between the enclosure and the pole was defined using the "Symmetric Node-to-Segment with Edge Treatment" card definition in VPS. The associated parameters, including a coefficient of friction of 0.2, stiffness proportional damping of 0.1, and a contact thickness equal to half of the laminate thickness, were employed. The simulation was performed for 10 milliseconds, with output data saved at intervals of 0.01 milliseconds.

2.3.1.3 Automation of simulation chain

The automation of the simulation chain, depicted in Figure 2.14, consisted of four distinct stages. In the initial phase, Python scripts were created to automate the thermoforming simulation setup, as detailed in this Section 2.3.1.2. These scripts generated simulation setup files for various layer configurations (e.g., 4 layers, 6 layers, and so on). The input parameters included die geometry, specific material, and processing parameters obtained by Latin hypercube sampling, as discussed in this section 2.3.1.1. The scripts produced files with ".pc" and ".ori" extensions containing simulation-specific details understood by the solver.

Advancing to the second stage, the generated .pc files were uploaded to the High-Performance Computing (HPC) cluster. Subsequently, four simulations were performed concurrently, each utilizing 16 cores to improve throughput. For job scheduling on the HPC cluster, SLURM scripting with job array functionality, coupled with bash scripting, was employed. With this configuration on the HPC, a 10-layer thermoforming simulation, for example, required about 16 hours to complete.

In the third stage, leveraging the results of successful thermoforming simulations, Python scripting was employed to automate the generation of crash simulation setup files. This involved using the thermoforming simulation results as input to perform draping on the enclosure geometry of the master crash simulation setup file. Additionally, other parameters of the master crash simulation file dependent on draping were modified to create new crash simulation setup files.

Finally, the crash simulations were performed by providing the setup files to the VPS solver. Each simulation was executed on 8 cores with the same velocity and simulation time. The results from the crash simulation were later post-processed to obtain final crash parameter values. In total, 400 simulations were conducted, of which about 65% were successful. This is attributed to the uncertainty in achieving the formability of a composite sheet with a thermoforming process given a specific set of input parameters.

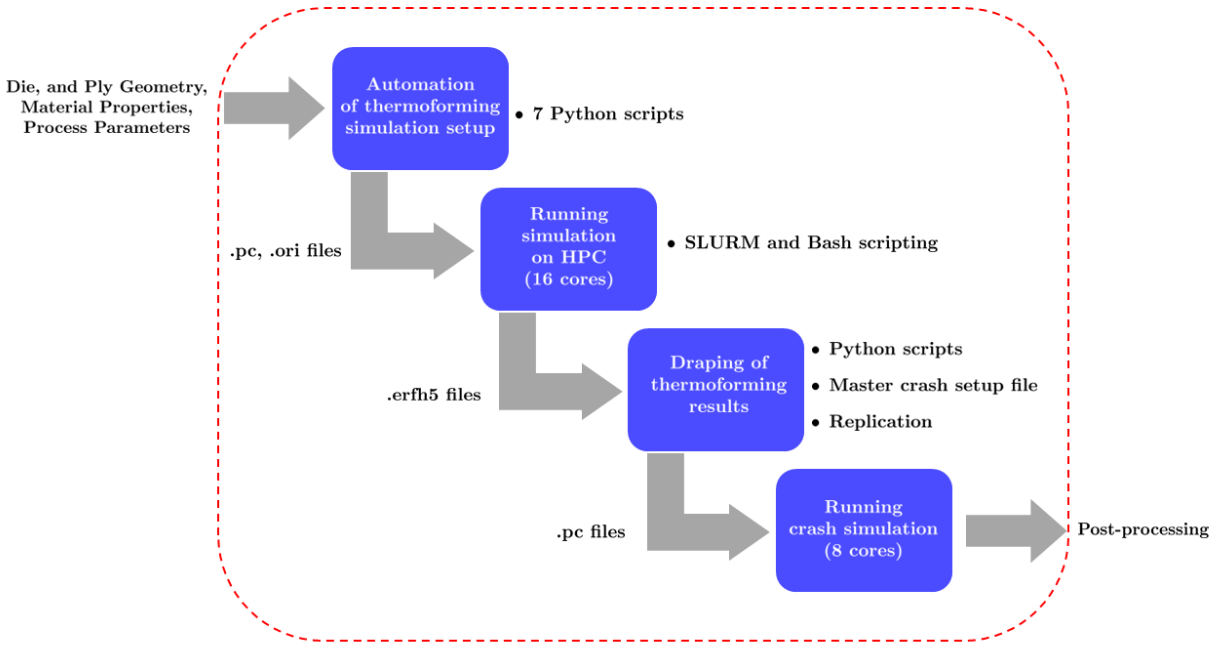


Figure 2.14: Schematic of automation process

2.3.1.4 Post-processing of simulation results

Following the simulation result files from the previous step, Python scripts were developed to extract both, the contact force plot between the enclosure and the pole, and the total energy absorbed by the enclosure assembly during impact, as CSV files. To calculate the intrusion of the stationary pole inside the enclosure assembly, two edge sensor nodes were strategically positioned. One was placed at the point of contact between the enclosure and the pole, while the other was placed at the opposite end of the enclosure along the same line. The translation data from these sensor nodes during the duration of impact were recorded and extracted as CSV files. A sample of the extracted results pertaining to the 10 layer sample simulation discussed in the previous sections are shown in Figure 2.17 in Section 2.4.2.

Using the extracted csv files, crash parameters such as peak load (F_p), Crush Load Efficiency (CLE), Specific Energy Absorption (SEA) and deformation (ΔY_{node}) were calculated. Each of these parameters are discussed in detail as follows:

- **Peak Load (F_p):** This parameter represents the maximum load experienced by the enclosure during side pole impact [82] [83]. It is obtained directly by selecting the maximum value of the contact force plot. It is a crucial measure to understand the highest force that the enclosure needs to withstand to prevent damage to the battery module inside.

- **Crush Load Efficiency (**CLE**):** Calculated as the ratio of the average load during a side pole impact (F_{avg}) to the peak load (F_p) i.e.

$$\text{CLE} = \frac{F_{avg}}{F_p} \tag{2.22}$$

  where,

$$F_{\text{avg}} = \frac{\int_0^T F dt}{T}. \tag{2.23}$$

  This metric provides insights into the stability of impact [84] [85][86]. Its value ranges from 0 to 1 with 1 being theoretical maximum. A higher CLE value implies a stable impact.

- **Specific Energy Absorption (**SEA**):** This is the most important parameter used in the automotive industry to evaluate the crashworthiness [84][85][87]. It provides a measure of how effectively the enclosure absorbs energy per unit its mass. Further, it given by dividing the energy absorbed (EA) during an impact by the total mass of the enclosure assembly (m_{total}). It is given by :

$$\text{SEA} = \frac{\text{EA}}{m_{\text{total}}} \tag{2.24}$$

  where, $m_{\text{total}} = m_{\text{ribs}} + m_{\text{lid}} + m_{\text{encl.}} + m_{\text{ns-mass}}$.

- **Deformation ($\Delta \text{Y}_{\text{node}}$):** This additional parameter was added to quantify penetration of pole inside the enclosure after impact. Understanding this is critical for assessing

the structural integrity and safety of the battery pack. Ideally a composite enclosure should hinder the intrusion to protect the battery module inside. In this study, it was calculated as the displacement of a specific node on the enclosure after impact.

2.3.2 Surrogate model development

2.3.2.1 Data pre-processing

Following post-processing step, discussed in the previous section 2.3.1.4, the input variables (i.e. n_{ls}, t_l, $\mathbf{\Phi}_{fib}$, v_p, T_i, T_{pd}, and T_{air} and crash parameters (i.e. F_p, CLE, SEA, $\Delta \mathrm{Y}_{\mathrm{node}}$) were assembled as input features and target labels into a dataset. Further, during the analysis, it was observed that approximately 8% of these data points in the dataset exhibited high values of SEA and were not used in the analysis. Subsequently, the final dataset consisted of 245 data points. To facilitate model training and evaluation, the dataset was divided into a training and testing set following a 80 - 20 split i.e. 196 data points for training and 49 for evaluation. Both, the input features and the target labels were scaled using the Z-scoring technique to improve training process of machine learning model.

2.3.2.2 Model selection

In the initial phase of the study, linear regression using the Scikit-learn [88] library was applied to the training dataset to assess its ability to predict output labels. This step aimed to evaluate its potential for predictions and, if unsuccessful, to derive valuable insights into the learning task. The application revealed that the linear regression model was able achieve appreciable accuracy in predicting two crash parameters. However, its performance in predicting other output labels was poor.

To investigate if regularization may improve model's performance. Both L_1 and L_2 regularization were implemented, and it was observed that regularization had no impact in improving the predictions. Despite this effort, the linear regression model remained unable to capture the non-linear relationships between input variables and output labels.

Recognizing the limitations of linear regression in capturing non-linear patterns, GPR were

selected for their inherent ability to capture non-linear relationships and providing point-wise uncertainty estimates in predictions.

GPR was implemented using GPy library [89] owing to its simplicity, extensive support for various kernels and stability. Initially, commonly used RBF kernel was employed which improved the predictions further compared to LR. Following various other kernel combinations were test until predictions were obtained. The results these experiments are discussed in detail in section 2.4.3.

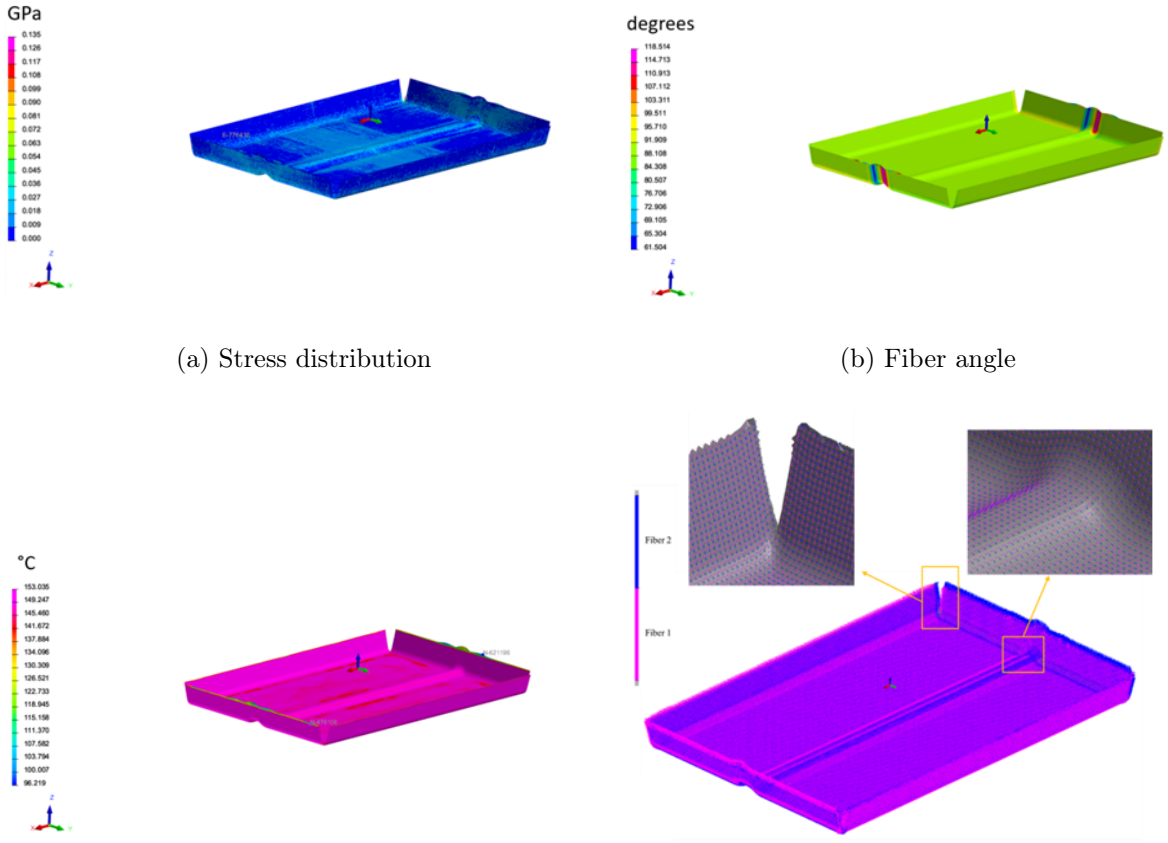
2.4 Results

This section presents the findings of this study, beginning by highlighting the results of thermoforming, followed by crash simulation. It then explores the predictions made by the machine learning model, including its validation and performance with new datasets. The final part of this section delves into the results obtained from Monte Carlo uncertainty quantification, providing a comprehensive overview of the various aspects of the research.

2.4.1 Thermoforming simulations

The simulations were performed using the simulation setup outlined in the preceding sections, and the results are shown in Figure 2.15 (a)-(d). As shown in (a), the stress levels remained below the fibers' failure limit, resulting in no fiber breakage. Moreover, the carbon fiber laminate conformed to the die's shape without any wrinkles, distortions or voids.

The surface finish was found to be good, and the die's curved surfaces were accurately capture during thermoforming.Throughout the thermoforming process, the organosheet underwent significant deformation, causing a deviation in the fiber angle from its original value. The altered fiber orientations are depicted in figure 2.15 (b) and (d). Notably, in most regions, the fiber orientation remains constant, except in areas with pronounced curvature, as seen in Figure 2.15 (d).



(a) Stress distribution

(b) Fiber angle

(c) Final temperature distribution

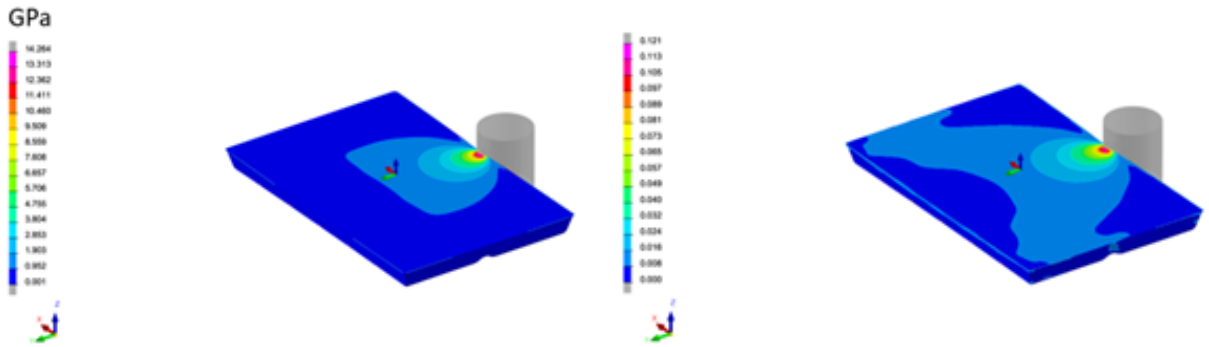
(d) Final fiber orientations

Figure 2.15: Results from thermoforming simulation of 10 layer simulation sample [1]

These modified fiber orientations play a crucial role in defining the overall mechanical properties of the formed part. Consequently, these new orientations were incorporated into the battery enclosure part in the subsequent crash simulation, as detailed in the following section.

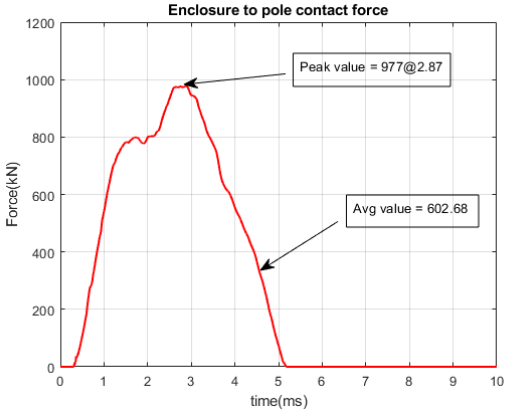
2.4.2 Crash simulations

Crash simulations were conducted to simulate the side pole impact test of the formed enclosure using explicit dynamic analysis. Owing to large deformations and nonlinear response, the explicit time marching scheme was implemented. Notably, it was observed that the battery enclosure assembly bounces back after the impact with a rigid pole without significant permanent deformation.

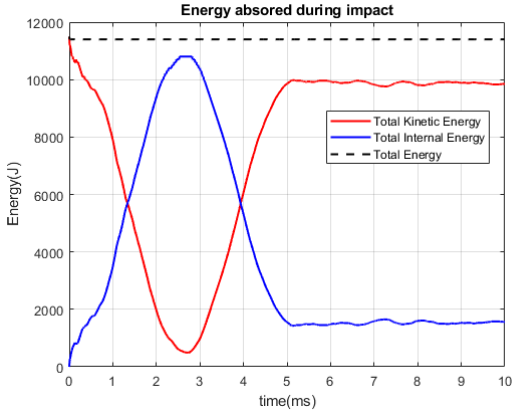


(a) Stress distribution (b) Strain distribution

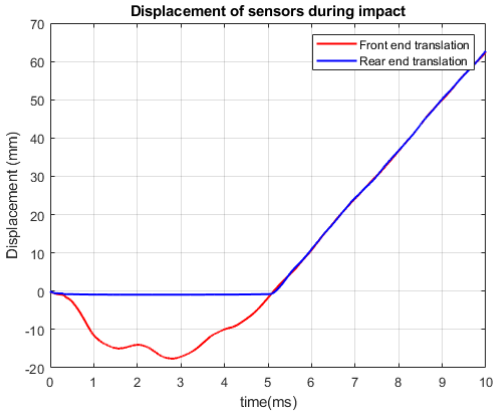
Figure 2.16: Von-mises stress and strain distribution in enclosure assembly during impact [1]



(a) Contact force plot



(b) Energy absorbed by enclosure



(c) Translation of edge sensor nodes

Figure 2.17: Post-processing result from a 10 layer simulation sample

Figure 2.16 illustrates the (a) Von Mises stress and (b) strain distribution at the impact's peak. Notably, a stress concentration was detected at the impact point. Further, Figure 2.17 (a), the maximum and average contact force values during impact are depicted, serving as key parameters for subsequent crash parameters calculations. 2.17 (b) shows the energy interaction throughout the impact, demonstrating an increase in internal energy and a decrease in kinetic energy.

However, the total energy was found to remain constant during the impact. Furthermore, Figure 2.17 (c) shows the translation of edge-node sensors during impact. By subtracting the translations of edge node sensors the intrusion of pole inside the battery enclosure was calculated.

2.4.3 Predictions from ML model

The objective of this study was to develop a surrogate model by fitting a regression model on generated data, as discussed in Section 2.3.1. To evaluate the performance of the regression model metric such as R-squared (R^2), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) were utilized.

The analysis started by fitting a linear regression model, which served as a baseline model. Linear regression exhibited satisfactory performance in predicting F_p and $\Delta \mathrm{Y}_{\mathrm{node}}$. However, it fell short in accurately predicting two other labels. Further, LR was regularized with L_1 (LASSO) and L_2 (Ridge) norms of the weight in an attempted to alleviate the effect of any irrelevant features on the predictions. The best value of regularization parameter λ was found by preforming grid search on [0.0001, 0.001, 0.01, 0.1, 1, 10] with five fold cross validation. It was found LASSO with $\lambda = 0.01$ provided a marginal improvement, as detailed in Table 2.2, but the predictions for SEA and CLE were still subpar, as seen in figure 2.18. This indicated that while linear regression is effective for certain types of data, it not suitable capturing more complex relationships or non-linear patterns.

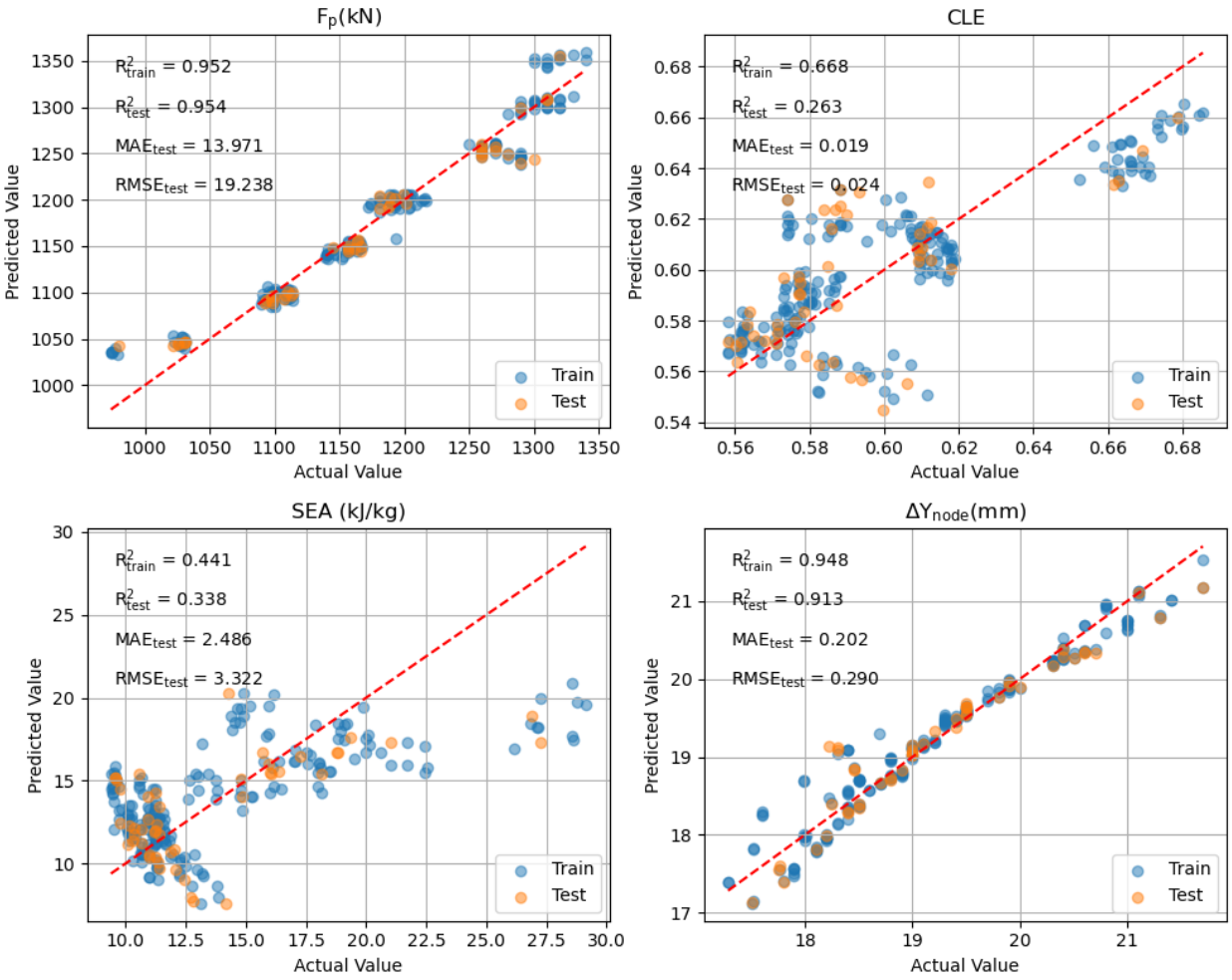


Figure 2.18: Predictions from linear regression

Table 2.2: Comparative results for performance of LASSO and GPR with different kernels

Op / Est	LR (LASSO) ($\lambda = 0.01$)				GPR (RBF)				GPR (M32)			
	R^2_{train}	R^2_{test}	MAE	RMSE	R^2_{train}	R^2_{test}	MAE	RMSE	R^2_{train}	R^2_{test}	MAE	RMSE
F_p	0.952	0.954	13.971	19.238	0.981	0.959	13.71	18.146	0.988	0.962	13.267	17.502
CLE	0.668	0.263	0.019	0.024	0.867	0.481	0.016	0.02	0.908	0.474	0.016	0.02
SEA	0.441	0.338	2.486	3.322	0.758	0.512	2.076	2.854	0.847	0.553	2.021	2.731
ΔY_{node}	0.948	0.913	0.202	0.29	0.984	0.959	0.137	0.199	0.990	0.957	0.137	0.206

Subsequently, Gaussian process regression was employed to address the limitations of linear regression. The GPR models were constructed using different kernels, starting with a RBF and progressing to other kernels such as Exponential, Matern$\frac{5}{2}$, Matern$\frac{3}{2}$.

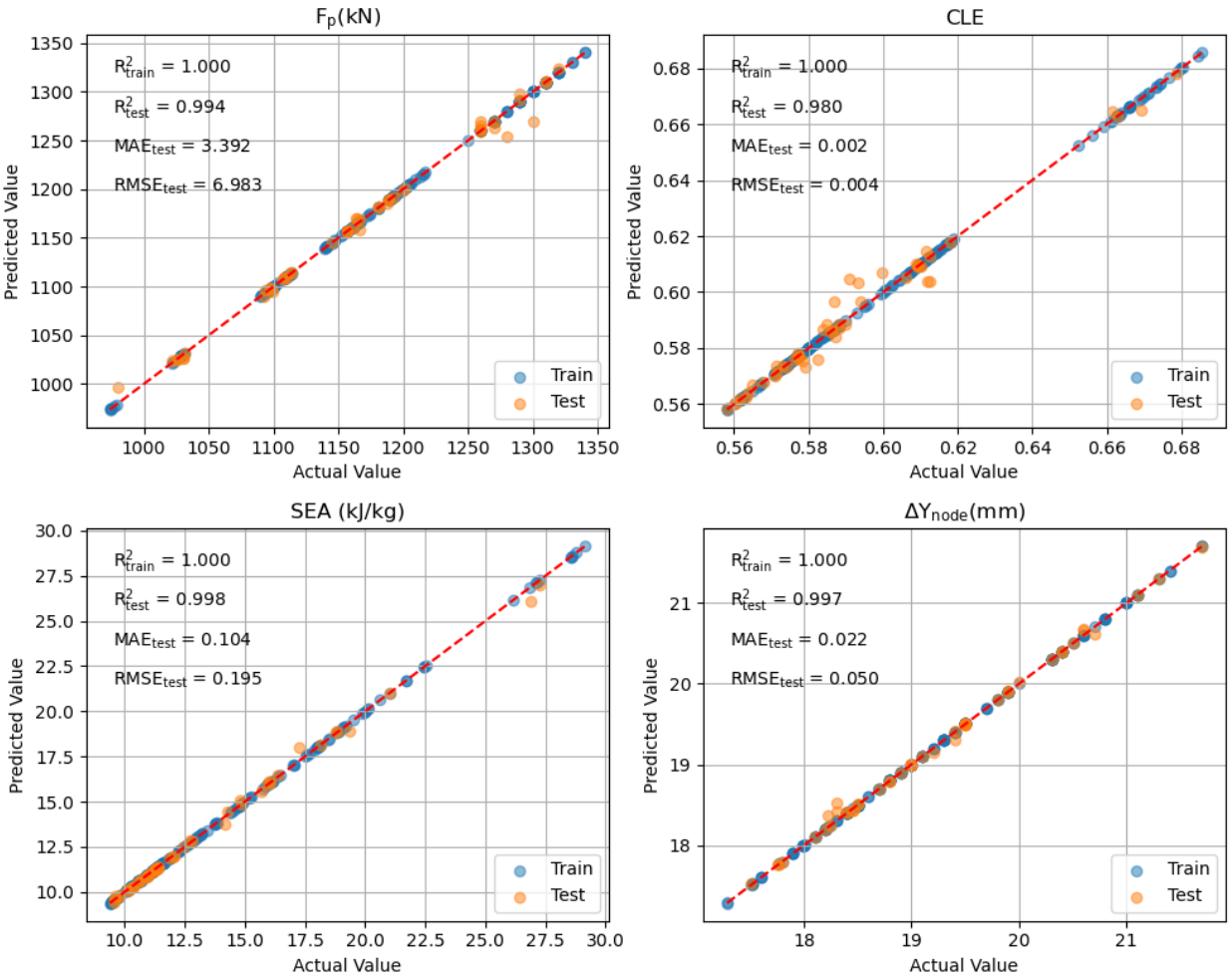


Figure 2.19: Predictions from Gaussian Process Regression

The prediction results for RBF, Matern$\frac{3}{2}$ kernel is shown in Table 2.2. As observed, the RBF kernel was able to provided a significant improvement in prediction of SEA and CLE compared to LASSO regression. Further, Matern$\frac{3}{2}$ provided a marginal improvement in predictions.

Table 2.3: Comparative results for performance of GPR with different ARD kernels

Op / Est	GPR (RBF + ARD)				GPR (Exp + ARD)				GPR (MAT32 + ARD)			
	R^2_{train}	R^2_{test}	MAE	RMSE	R^2_{train}	R^2_{test}	MAE	RMSE	R^2_{train}	R^2_{test}	MAE	RMSE
F_p	1	0.981	5.100	12.414	1	0.989	3.518	9.189	1	0.994	3.392	6.983
CLE	1	0.982	0.002	0.004	1	0.981	0.002	0.004	1	0.980	0.002	0.004
SEA	1	0.984	0.199	0.515	1	0.998	0.102	0.199	1	0.998	0.104	0.195
ΔY_{node}	1	0.994	0.040	0.075	1	0.996	0.023	0.063	1	0.997	0.022	0.05

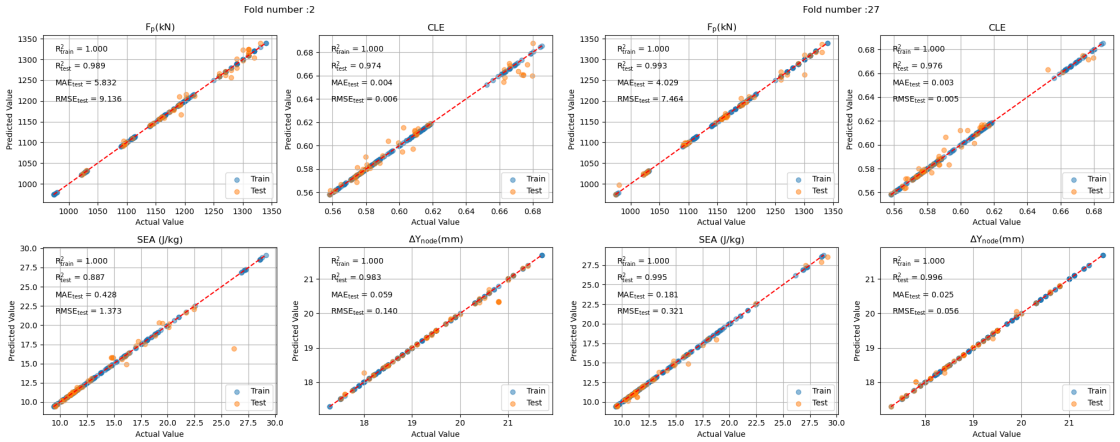
Finally, to improve the predictions further, automatic relevance determination (ARD) [90] was incorporated in each tested kernels. The ARD allows models to adaptively adjust the relevance of input features to improve predictions, this discussed in detail in section 2.1.6.2.

As detailed in Table 2.3, each kernel used previously was tested with ARD. It was found that the Matern$\frac{3}{2}$ with ARD provided excellent results in achieving the highest R^2 scores across all output labels, as seen in figure 2.19. Further it can be noted that Matern$\frac{3}{2}$ kernel with ARD was able to effectively capture the underlying patterns in the dataset and was able to produce more accurate predictions compared to other models

2.4.3.1 Repeated K- fold cross validation

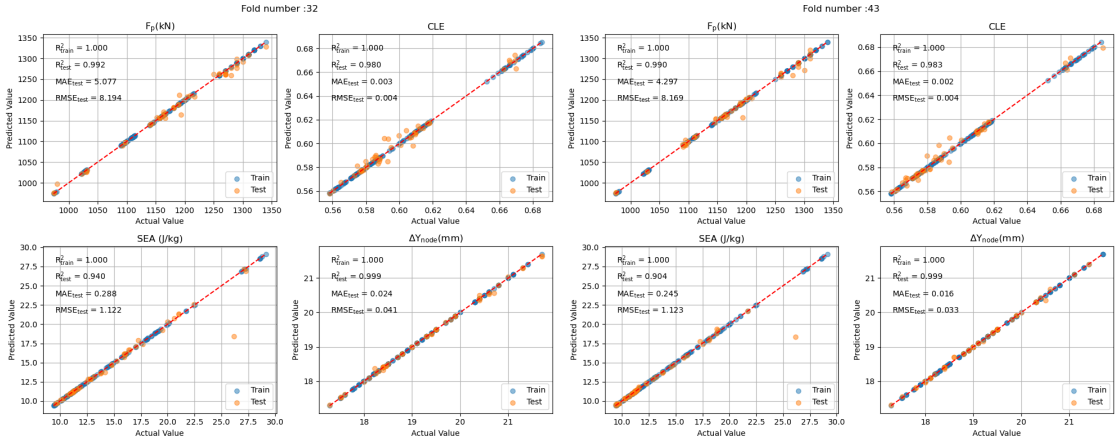
In this dissertation, repeated K-fold cross-validation was employed as a method to assess the consistency of performance and generalizability of the GPR model. This technique involves dividing the dataset into k equally sized subsets, and iteratively using $k-1$ of these for training the model, while the remaining subset is reserved for testing.

The repetition of this process multiple times, with shuffled data splits, ensures the reliability of performance metrics, reducing the impact of potential randomness in the initial partitioning. This technique provides a more comprehensive view of model performance, ensuring the predictions are robust and reflective of the model's ability to generalize to diverse data instances.



(a) Sample 2

(b) Sample 27



(c) Sample 32

(d) Sample 45

Figure 2.20: Randomly selected samples from repeated K-fold cross-validation step

Figure 2.20 shows predictions on the random samples selected from the cross-validation process. Further, the Figure 2.21 depicts average value of the performance metrics — R^2, MAE, and RMSE — over 10 repeats. For each of the output — F_p, CLE, SEA, and ΔY_{node} — the variation in performance metrics across the 10 repeats is evident. This variability is a normal aspect of repeated k-fold cross-validation, reflecting how different splits of the data can influence the model's performance

Despite the fluctuations, the graphs generally indicate a strong predictive capability, as shown by the high R^2 values, and low MAE and RMSE values, which is indicative of a

well-performing model. These results underscore the robustness of the GPR model, as it consistently yields reliable predictions across different subsets of data.

2.4.4 Prediction on new dataset

Following the evaluation of model with repeated k-fold cross validation in the preceding section. The performance of the GPR model was further evaluated on a new dataset. The new dataset consisted of 10 data points that were generated by performing new simulations. In each of theses simulations $n_{ls} = 4$, whereas other properties such as v_p, T_i, T_{pd}, T_{air} were randomly selected (from table 2.1) while the values of t_l, and $\mathbf{\Phi}_{fib}$ were set different from the range was used [91].

The thermoforming simulation was performed on a HPC unit with 16 cores, further, the crash simulation was performed on 8 core 11^{th} Gen Intel (R) i7@3GHz PC. The total time taken for one complete simulation was about 29000 seconds (i.e. 8 hrs) approximately.

Table 2.4: $n_{ls} = 4$; v_p, T_i, T_{pd}, T_{air} randomly selected (from table 2.1); t_l and $\mathbf{\Phi}_{fib}$, different from testing and training

Outputs	0.7 mm, 6.5 m/s, 318 °C, 131 °C , 24 °C, (45, -45, 45, -45)			0.9 mm, 5.7 m/s, 304 °C, 91 °C , 22 °C, (45, -45, 45, -45)		
	Sim.	GPR	% error	Sim.	GPR	% error
F_p	1050.0	1040.7	**0.88**	1030.0	1092.4	**6.06**
CLE	0.527	0.577	**9.43**	0.542	0.581	**7.41**
SEA	13.61	12.18	**10.51**	13.78	14.25	**3.44**
ΔY_{node}	17.14	17.52	**2.24**	16.85	17.84	**5.90**

For sake of brevity, the results of only four simulations are detailed in Table 2.4 and 2.5. The results for the rest of the simulations are listed in Appendix. It is important to note that a significantly different values of t_l and $\mathbf{\Phi}_{fib}$ was used in these simulations compared to one used data generation step. For comparison the predictions from the GPR model are listed beside the simulation results.

Further, it should be noted that GPR model was able to provide the approximate predictions instantaneously compared to the actual simulations. Additionally, the % errors for the

four simulations were well within reasonable range with as lows as 0.88 % for F_p and as high as 10.51% in case of SEA. This demonstrated GPR models predictive accuracy with model's predictions being consistently close to the simulation results. This further underscores its effectiveness in capturing the complex relationships within the dataset.

Table 2.5: $n_{ls} = 4$; v_p, T_i, T_{pd}, T_{air} randomly selected (from table 2.1); t_l and $\mathbf{\Phi}_{fib}$, different from testing and training

Outputs	0.7 mm, 6.5 m/s, 318 °C, 131 °C , 24 °C, (0, 45, -45, 60)			0.9 mm, 5.7 m/s, 304 °C, 91 °C , 22 °C, (0, 45, -45, 60)		
	Sim.	GPR	% error	Sim.	GPR	% error
F_p	1000.0	1018.143	**1.81**	1000.0	1052.0	**5.2**
CLE	0.549	0.579	**5.54**	0.554	0.578	**4.45**
SEA	13.70	12.60	**8**	13.65	13.45	**1.48**
ΔY_{node}	16.63	17.30	**3.99**	16.54	17.51	**5.85**

Subsequently, from further analysis of results from the whole dataset it was found that mean absolute % error (MAPE) for F_p, CLE, SEA and ΔY_{node} was 4.09, 5.76, 8.08, and 5.48 respectively. Furthermore, % error for all the response variables for all simulations were less than equal to 14.64 %. The variation in error rates across different outputs and configurations highlights the model's sensitivity to input parameters.

2.4.5 Comparison with previously published results

In this section the performance of the GPR model is compared against the outcomes reported in previous study utilizing tree-based ensemble methods [1]. The primary objective was to assess the improvements in predictive accuracy and error reduction achieved through the application of GPR model.

The tree-based ensemble methods are advanced machine learning algorithms that combine predictions from multiple tree models to improve the overall predictive performance compared to a single tree. The most common ensemble methods include bagging, boosting, and their notable implementations are: Random Forest (RF), Gradient Boosting (GB), and Extreme Gradient Boosting (XGBoost).

Bagging, or Bootstrap Aggregating involves generating multiple decision trees by using random subsets of the training data, drawn with replacement. Once the trees are built, bagging makes predictions by aggregating the results of all the trees through majority voting for classification problems or averaging for regression. This process helps reduce variance, leading to a more robust and accurate model.

Boosting involves building a series of models in a sequential manner, where each model attempts to correct the errors of its predecessor. The key idea is to train weak learners sequentially, with each new model focusing more on the data points that were previously misclassified or difficult to predict. Unlike bagging, boosting adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa.

Table 2.6: Comparative results for the performance of GPR, RF, GB, and XGBoost from the previous study [1] on the holdout set

Outputs	This study (GPR)		RF		GB		XGBoost	
	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE
F_p	0.994	3.392	-	-	-	-	-	-
CLE	**0.980**	**0.002**	0.944	0.004	0.969	0.003	0.968	0.004
SEA	0.998	0.104	-	-	-	-	-	-
ΔY_{node}	**0.997**	**0.022**	0.967	0.109	0.970	0.086	0.969	0.094

RF is an extension of the bagging technique. It creates a forest of decision trees, each trained on a random subset of the training data and using a random subset of features at each split during the tree building process. This method of "feature bagging" helps in decorrelating the trees, making the ensemble model more robust than individual decision trees. The final prediction is made by aggregating the predictions of all the trees. Further, GB and XGBoost are both based on the boosting technique. GB uses a gradient descent algorithm to minimize the differentiable loss functions when adding a new tree. On the other hand,

XGBoost is based on the GB framework. However, it is highly efficient, flexible and can handle sparse data. It includes a variety of regularization techniques that reduce overfitting and improve overall performance. Unlike the ensemble methods that rely on aggregating predictions from multiple models, GPR uses a non-parametric Bayesian approach to model the underlying distribution of the data.

Table 2.6 presents a summarized comparison of the performance metrics between the two studies. Notably, GPR showcases significantly improved accuracy and a reduction in error for output variables such as CLE and $\Delta \mathrm{Y}_{\mathrm{node}}$. The higher R^2 values and substantially lower MAE with GPR suggest superior ability in modeling complex patterns within the dataset, resulting in an improved predictions.

2.4.6 Uncertainty estimate from GPR posterior

The strength of GPR lies in its ability to provide not only predictions but also associated uncertainty which is indicative of model's confidence in its predictions. The mean from the GPR posterior represents predictions, similarly the variance, which is used to calculate the confidence interval, provides an uncertainty estimate.

The uncertainty estimate from the GPR using Matern$\frac{3}{2}$ with ARD kernel for F_p, CLE, SEA, and $\Delta \mathrm{Y}_{\mathrm{node}}$, is shown in the Figure 2.22. The blue error bar represents a 95% confidence interval (CI) that is the range within which the actual value is expected to lie with 95% probability, assuming a normal distribution of the predictions.

From the plots, it is evident that the CIs for almost all the output labels are narrow, implying the model's accuracy and high confidence in its predictions. However, it is important to note that this estimate primarily represents the epistemic component of total uncertainty [92] [63]. Since the inputs to the model were considered without uncertainty, and the noise variance estimate obtained from GPR being close to zero, suggesting the absence of an aleatoric component.

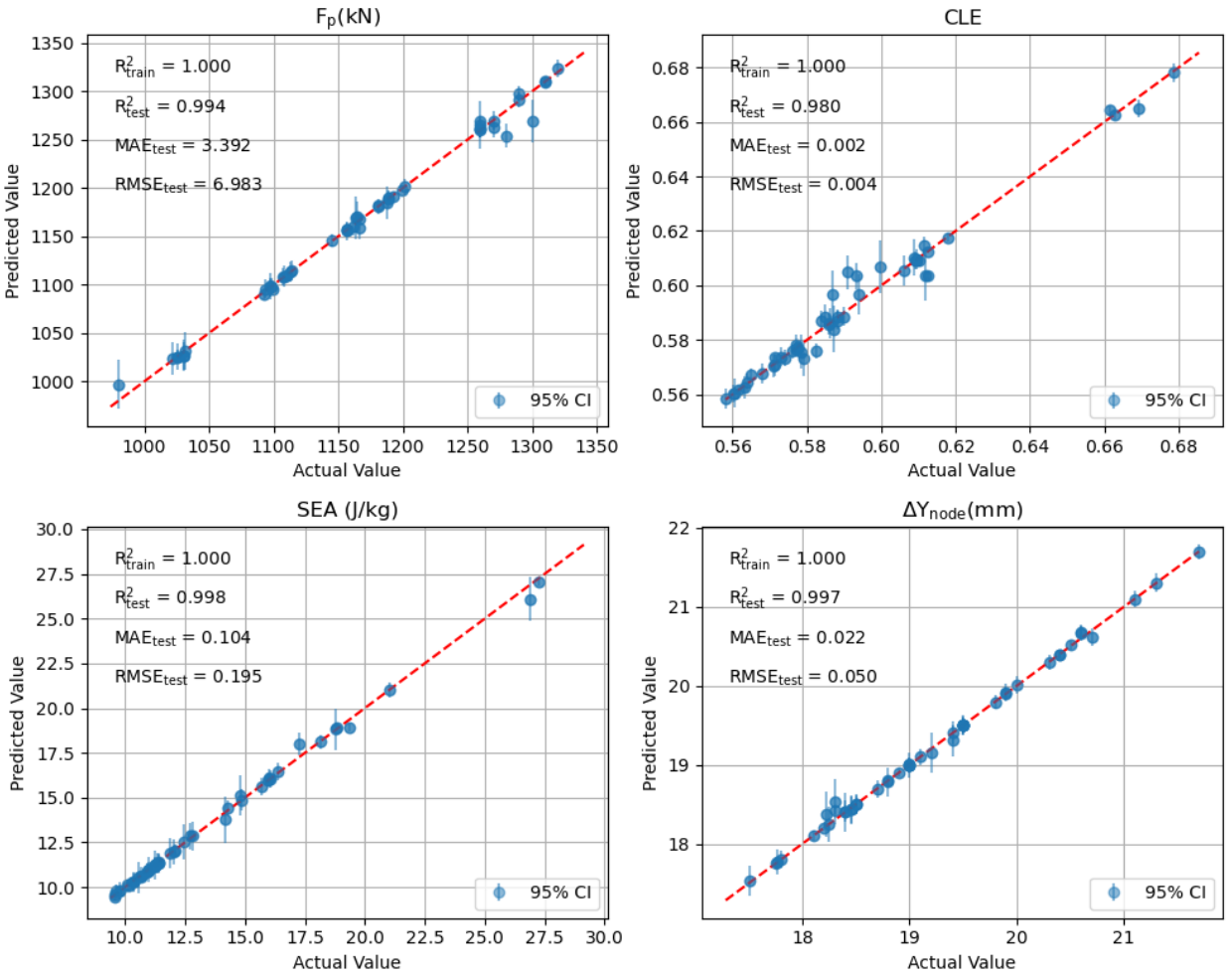


Figure 2.22: Uncertainty estimate (95% CI) from GPR (MAT32 + ARD) posterior on hold-out set represented as an error bar

Propagating the input uncertainties within the GPR framework increases modeling complexity as this approach challenges the assumption of a Gaussian distributed posterior [93] [94]. Specifically, including input uncertainty introduces additional variability that may affect the shape of the resulting posterior distribution, potentially deviating from the Gaussian form [95]. To address this challenge several techniques such as Taylor series expansion, moment matching integration are employed. A comprehensive list of techniques used to tackle this problem are enumerated here [96].

Given the inherent complexity of performing uncertainty propagation within the GPR framework, this study employed Monte Carlo uncertainty propagation detailed in the sub-

sequent section.

2.4.7 Monte Carlo uncertainty propagation

The Monte Carlo uncertainty propagation study was undertaken to assess the effect of uncertainties associated with input variables on the output. The simulation scenarios were established by dividing the range outlined in Table 2.1 into three distinct cases: two at the boundaries of the range and one at the midpoint. For each scenario, two fiber orientation configurations were analyzed: Configuration A, defined by $\mathbf{\Phi}_{fib} = [0, 45, -45, 90]$, and Configuration B, defined by $\mathbf{\Phi}_{fib} = [30, -30, 60, -60]$.

The input variables t_l, v_p, T_i, T_{pd}, T_{air} were assumed to normally distributed with probability distribution parameters listed in table 2.7. Additionally, the input variable $\mathbf{\Phi}_{fib}$, not listed in the table, was also assumed with normally distributed with $\mu = A/B$ i.e. $[0, 45, -45, 90], [30, -30, 60, -60]$ and $\sigma = 2^{\circ}$ [97]. The uncertainly on n_{ls} variable was not considered owing to its deterministic nature in a realistic manufacturing setup.

Table 2.7: Probability distribution of input variables used for Monte Carlo uncertainty quantification study

Input Var.	Assumed Prob. Dist	Monte Carlo Simulation		
		Case #1	Case #2	Case #3
n_{ls}	Const.	4	10	16
t_l	Normal	$\mu = 0.1$, $\sigma = 1\%\ \mu$	$\mu = 0.35$, $\sigma = 1\%\ \mu$	$\mu = 0.6$, $\sigma = 1\%\ \mu$
v_p	Normal [98]	$\mu = 4$, $\sigma = 1\%\ \mu$	$\mu = 5.25$, $\sigma = 1\%\ \mu$	$\mu = 6.5$, $\sigma = 1\%\ \mu$
T_i	Normal [99]	$\mu = 200$, $\sigma = 1\%\ \mu$	$\mu = 300$, $\sigma = 1\%\ \mu$	$\mu = 400$, $\sigma = 1\%\ \mu$
T_{pd}	Normal [100]	$\mu = 20$, $\sigma = 0.75\%\ \mu$	$\mu = 120$, $\sigma = 0.75\%\ \mu$	$\mu = 220$, $\sigma = 0.75\%\ \mu$
T_{air}	Normal [100]	$\mu = 10$, $\sigma = 1.5\%\ \mu$	$\mu = 20$, $\sigma = 1.5\%\ \mu$	$\mu = 30$, $\sigma = 1.5\%\ \mu$

A total of six MC simulations were performed by sampling 5×10^6 samples and running pretrained GPR model on each of these samples. The results were aggregated and statistical parameters such as mean and standard deviation were extracted by fitting the observed

distribution, a shown in Figure 2.23 to 2.28.

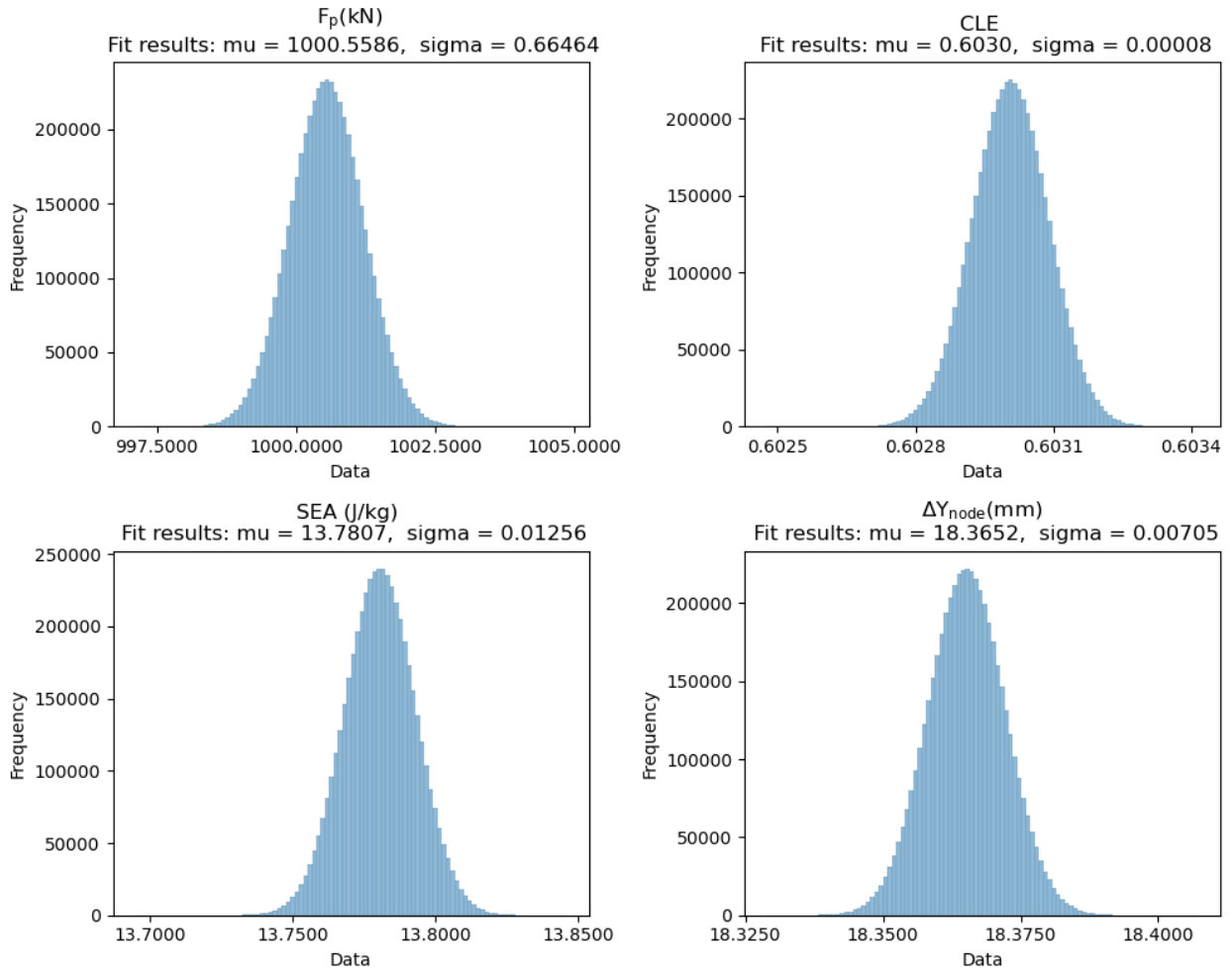


Figure 2.23: Results from MCS #1 with A configuration

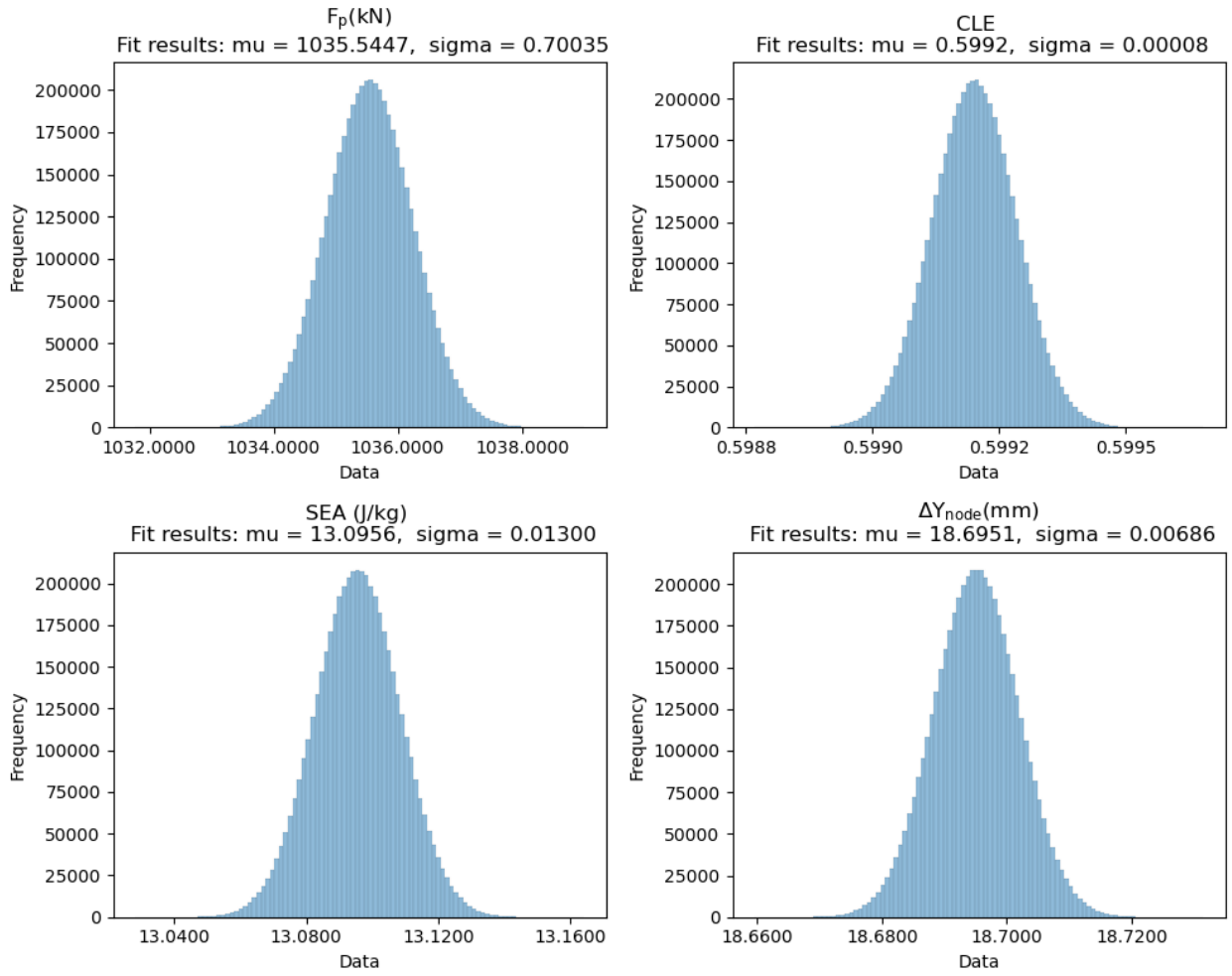


Figure 2.24: Results from MCS #1 with B configuration

The results for case #1 with configuration A, with four layers is shown in figure 2.23. As observed the histogram for F_p displayed a normal distribution with a mean μ of 1000.5586 kN and a standard deviation σ of 0.66464 kN. Further, the SEA plots showed a mean of 13.7807 J/kg with a standard deviation of 0.01256 J/kg. The CLE and ΔY_{node} had means of 0.6030 and 18.3652 respectively, with very tight standard deviations. Furthermore, in case of configuration B, shown in figure 2.24, similar behaviour was observed.

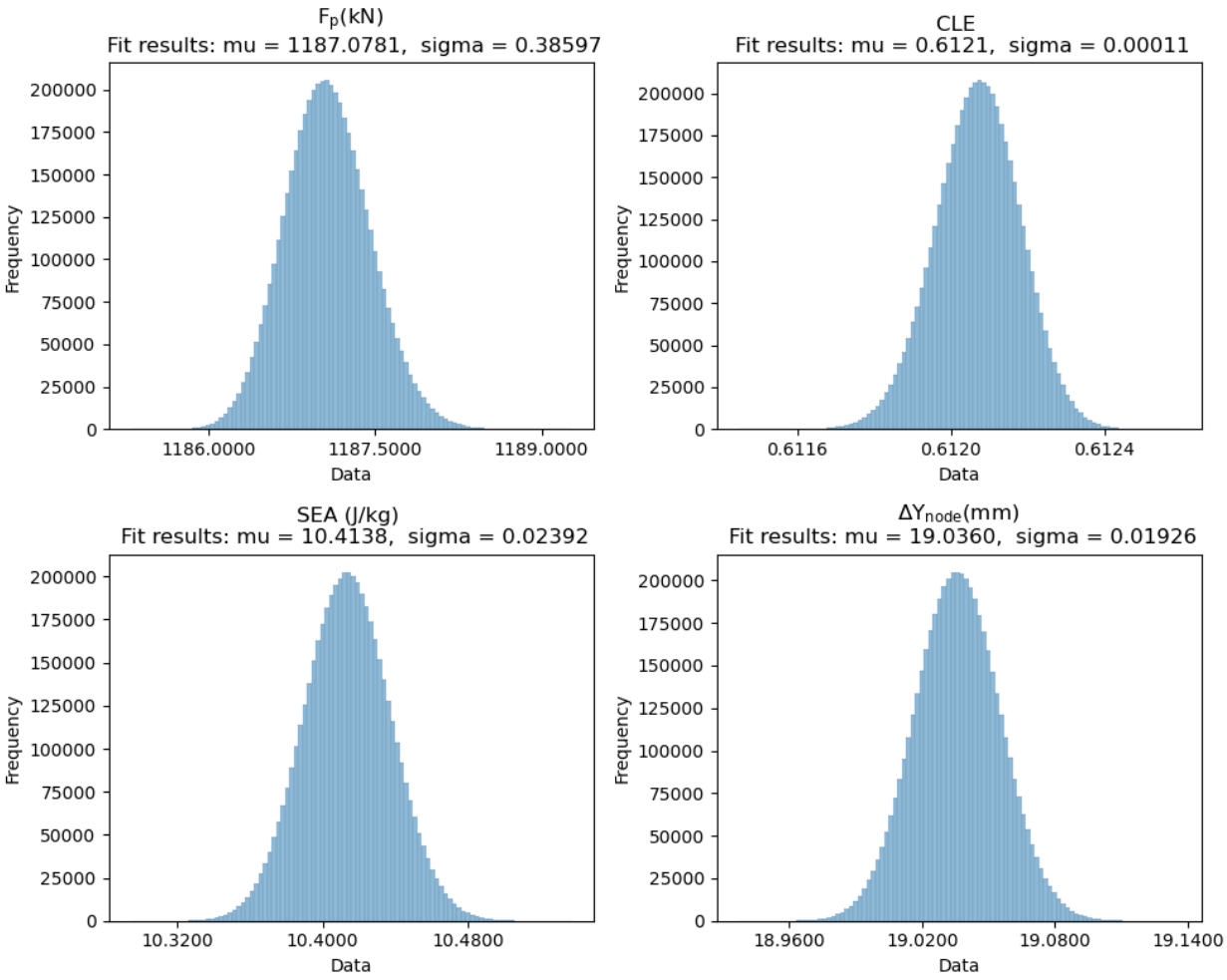


Figure 2.25: Results from MCS #2 with A configuration

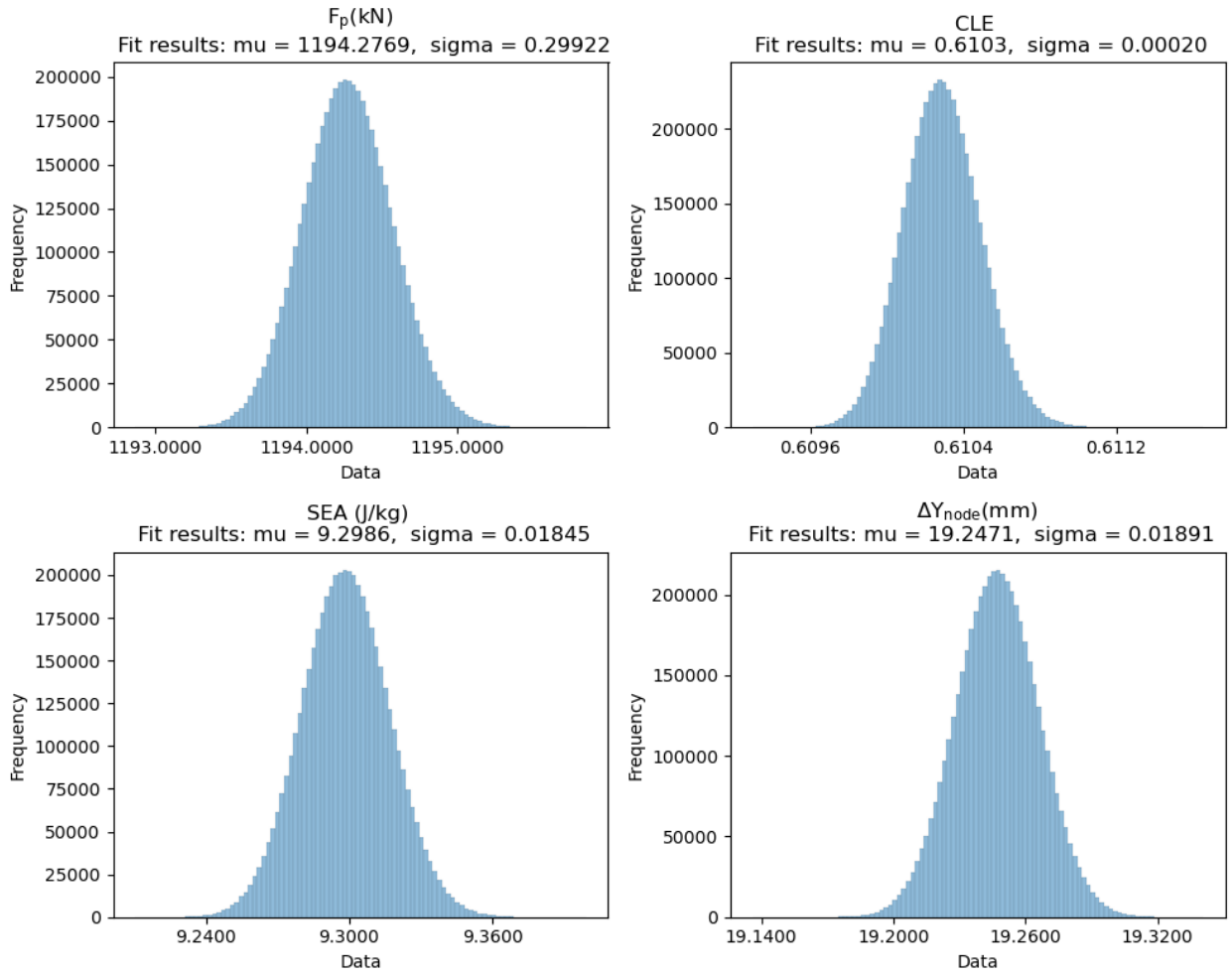


Figure 2.26: Results from MCS #2 with B configuration

The results for the case #2 with increased value of $n_{ls} = 10$ are shown in figure 2.25 and 2.26 for both configurations A and B respectively. As observed, the value of mean for F_p increased with values of standard deviations, 0.38597 kN and 0.29922 kN for A and B respectively. Similarly, an increase in the magnitude of both CLE and ΔY_{node} was observed. However, it was found that magnitude of SEA was decreased compared to the previous case.

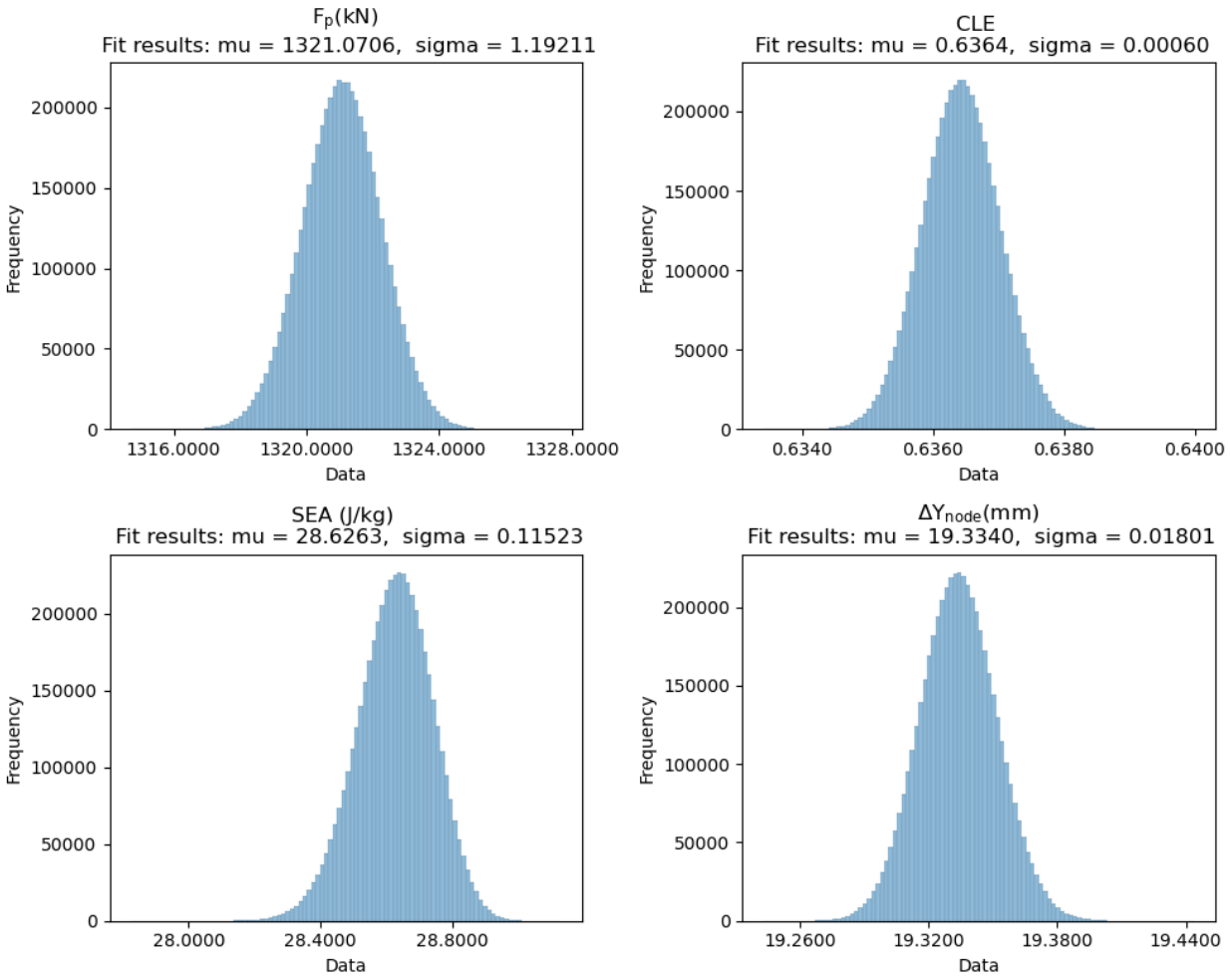


Figure 2.27: Results from MCS #3 with A configuration

In the third simulation scenario with n_{ls} set to 16, as illustrated in Figure 2.27, further evolution in the output distributions was observed. The F_p increased to 1321.0706 kN, accompanied by a widened standard deviation of 1.19211 kN, indicating increased variability. In-case of SEA the value of mean increased to 28.6263 J/kg, with a standard deviation of 0.11523 J/kg. The CLE and ΔY_{node} maintained their narrow distribution, with means at 0.6364 and 19.3340, and standard deviations at 0.00060 and 0.01801, respectively. A similar behaviour was observed in-case of configuration B, shown in figure 2.28.

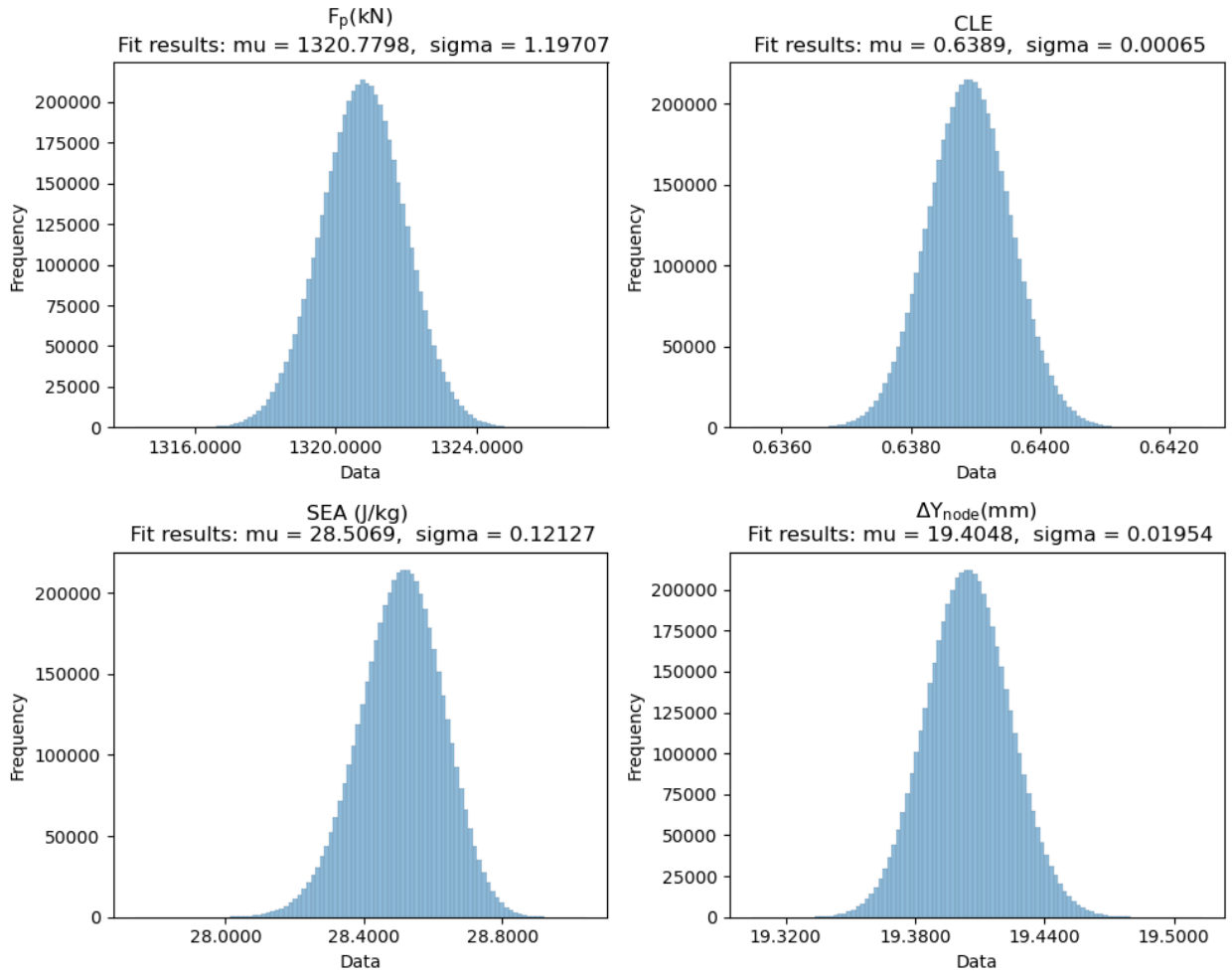


Figure 2.28: Results from MCS #3 with B configuration

Across all simulations, the resulting histogram of outputs was observed to follow normal distribution suggesting that the input uncertainties propagated through the model resulting in normally distributed outcomes with predictable variance. Additionally, it was observed that the variability in the results was relatively low, with standard deviations being a small fraction of the means.

2.5 Discussion

This study started with the data generation step that involved automating a complicated finite element simulation chain of thermoforming and crash simulations. To streamline the process few improvements to the simulation setup were added. These time and resources

intensive simulations were performed utilizing HPC clusters. Further, the results of one randomly selected simulations sample from the dataset was discussed in the preceding section. Further, the analysis of results demonstrated that the carbon fiber laminate conformed to shape of die without any fiber breakage, wrinkles, or distortions. A similar observation was made in the previous work by Kulkarni and Hale et. al. [9]

The results from all the simulations were post-processed and assembled into a dataset. Further, a GPR model was fitted and the results obtained were shared in the preceding section. From the analysis its was evident that GPR model was able to predict the output labels with high predictive accuracy, with R^2 greater than 0.980, MAE and RMSE less than 3.392 and 6.983 respectively, for all output labels.

In addition, GPR model was validated for the consistency and generalization of predictions using repeated K-fold cross validation. It was found that average value of evaluation metrics for different number of repeats was found to be close to obtained predictions.

Further, the GPR models predictive performance on the new dataset was tested. The model was able to provide predictions with absolute % error less than 14.64 % for all output labels. Additionally, MAPE for all output labels on the whole dataset was found to be less than 8.08 %. The model provided these approximate predictions in few milliseconds compared to actual simulations which took about 29000 sec (i.e. 8 hrs.) each.

Finally, a Monte Carlo uncertainty propagation was undertaken by incorporating uncertainty in inputs and propagating it through the surrogate model. Since do so within the GPR framework leads to additional complexity. It was observed the normally distributed uncertainty in the input variables propagated through the GPR model providing a normally distributed predictions. Further, it was observed that these uncertainty on input variables had overall small impact on the output labels.

In conclusion, it can be said that the GPR model was able better model the complex relationship within the dataset providing output labels with higher predictive accuracy compared to previous study [1].

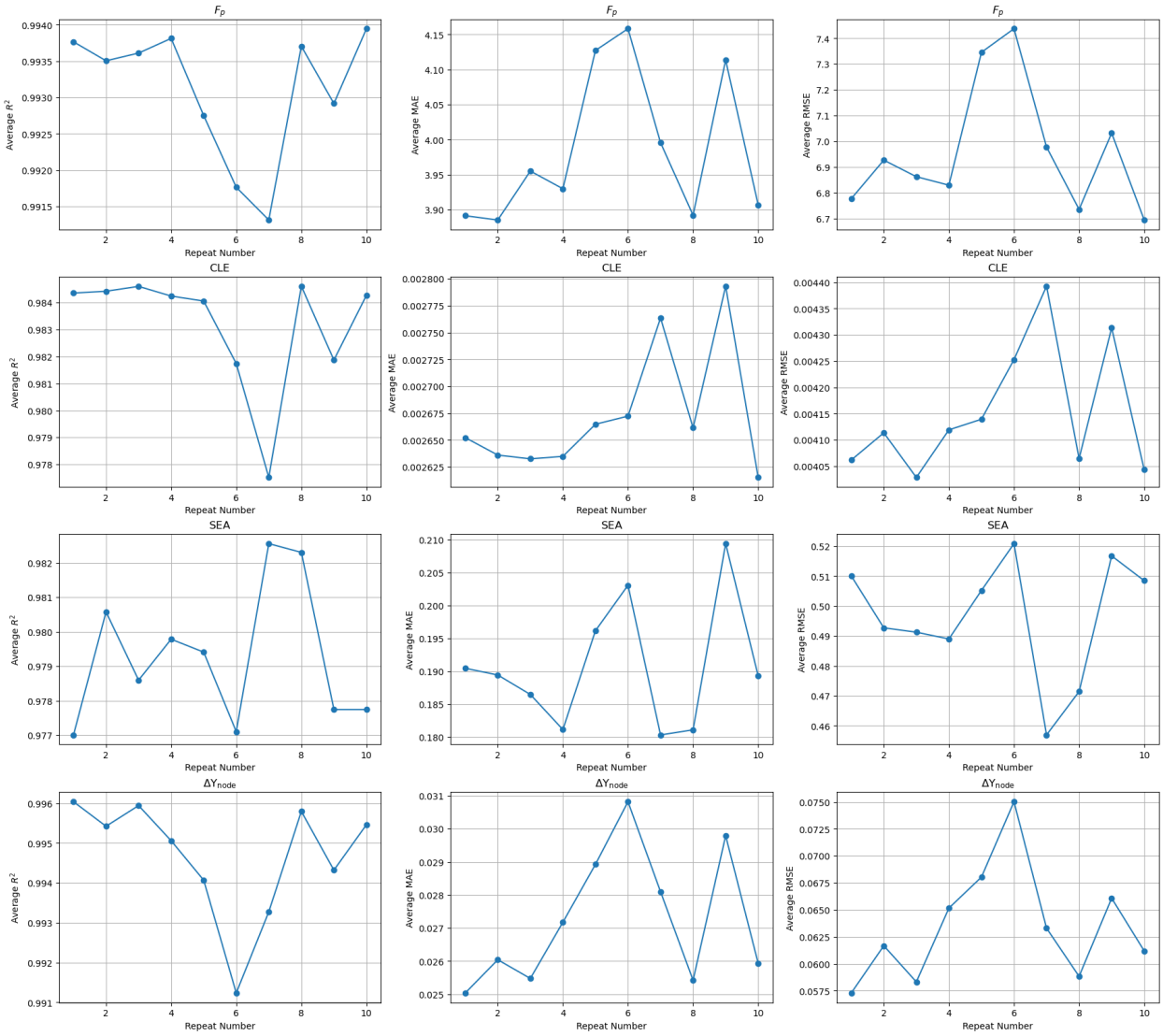


Figure 2.21: Average values of R^2, MAE, RMSE across different repeats for 5 - fold cross validation repeated 10 times

CHAPTER 3: SOLVING INVERSE SOURCE PROBLEM IN ENGINEERING DESIGN USING ML

3.1 Background

This section aims to provide background and insight into the second problem statement that this research work aims to tackle.

3.1.1 Inverse source problems in engineering

Inverse problems are a special class of mathematical problems aimed at inferring causal relationships from observational data. These problems are often ill-posed and plagued by numerical issues [11], yet they are widely encountered across various scientific and engineering disciplines. Recent decades have witnessed significant research efforts towards addressing these problems [101][102][103].

A subset of inverse problems, focusing on determining the 'source term' or right-hand side of a governing equation, is referred to as inverse source problems (ISP). ISPs are prevalent in various physics and engineering disciplines. An excellent example of the application of ISPs is found in Optical Molecular Imaging (OMI), where they enable the reconstruction of bio-luminescent and fluorescent marker distributions within human tissue from light intensity measurements. This application is pivotal for non-invasive exploration of cellular and molecular functions, facilitating early diagnosis of diseases and evaluation of treatment responses [104] [105].

In the field of Radiative Heat Transfer, ISPs play a significant role in deducing temperature distributions and material thermal properties. By interpreting radiation intensity and medium property measurements [106] [107]. The ability to accurately model and analyze heat transfer processes is essential for enhancing energy efficiency and ensuring the reliability

and safety of engineering products.

Moreover, ISPs are instrumental in neurology, particularly through Magnetoencephalography (MEG) and Electroencephalography (EEG), where they assist in localizing brain activity sources. This application is of paramount importance for advancing our understanding of brain function and diagnosing neurological disorders [108] [109]. The insights gained from these studies are fundamental to the field of neuroscience, offering new avenues for therapeutic interventions and cognitive research. These examples underscores the importance of ISPs from understanding complex biological systems to solving engineering challenges.

In the domain of physics and engineering, differential equations play a pivotal role in modeling physical phenomena. In their most general form these equations can be expressed as [110]:

$$\begin{aligned} \mathcal{N}(u(\mathbf{x},t);\boldsymbol{\gamma}) &= f(\mathbf{x},t) \quad \mathbf{x}\in\Omega, \quad t\in[0,T] \\ \mathcal{I}(u(\mathbf{x},0)) &= g(\mathbf{x}) \\ \mathcal{B}(u(\mathbf{x},t)) &= h(\mathbf{x},t) \quad \mathbf{x}\in\partial\Omega \end{aligned} \tag{3.1}$$

where $\Omega \subset \mathbf{R}^d$ with the boundary $\partial\Omega$, $\mathbf{x}$ indicates the space coordinate vector, t indicates the time, u represents the solution, $\boldsymbol{\gamma}$ are the parameters related to the physics, f is the source term, $\mathcal{N}$ is the non linear differential operator, $\mathcal{I}$ is the operator indicating arbitrary initial conditions, $\mathcal{B}$ is the operator indicating arbitrary boundary conditions.

In engineering we particularly deal with two principal types of problems: the forward problem and the inverse problem. The forward problem involves determining the solution, denoted as u, by considering the given values of $\boldsymbol{\gamma}$, f, $\mathcal{B}$, and $\mathcal{I}$. It essentially focuses on finding the resulting effect or output when the sources or inputs are known. On the other hand, the inverse problem, or more specifically the inverse source problem as its the main focus of this work, tackles the challenge of deducing the source term, represented as f, based on the known values of $\boldsymbol{\gamma}$, u, $\mathcal{B}$, and $\mathcal{I}$. This problem requires a deeper understanding and analysis of the relationship between the sources and the observed effects.

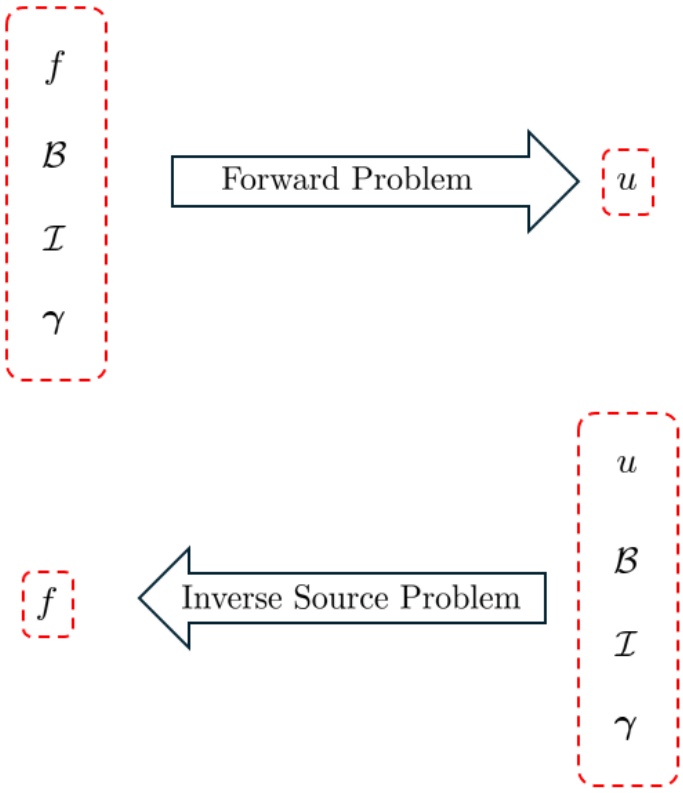


Figure 3.1: Schematic of forward and an inverse source problem

3.1.2 Physics Informed Neural Networks (PINNs)

PINNs are a class of deep learning models that integrated the physics into the training of neural networks to solve complex problems governed by differential equations [111]. Unlike traditional deep learning methods, PINNs embed the governing equations within the neural network's loss function to ensure better generalization and physically consistent predictions.

PINNs are designed to optimize a composite loss function that balances between fitting to data and adherence to physical laws, making them especially useful in fields like fluid dynamics [112], solid mechanics [113], and thermal sciences [114]. Their ability to deal with sparse or noisy data and to generalize well under physical constraints has positioned PINNs as a powerful tool for scientific computing and engineering applications. Despite challenges such as high computational demands and the complexity of loss function landscapes [115],

ongoing research is focused on enhancing their efficiency and applicability, promising to further bridge the gap between data-driven insights and physics-based modeling

Incorporating the description of the physical phenomenon from preceding section into the framework of PINNs. It can be extended to tackle both forward and inverse problems governed by partial differential equations. The objective is to approximate the solution $u(\mathbf{x}, t)$ by a neural network $\mathbf{NN}(\mathbf{x}, t; \theta)$, where θ encapsulates the network's parameters (weights and biases). The loss function for a PINNs in this context is constructed to ensure compliance with both the differential equation $\mathcal{N}$, the initial and boundary conditions $\mathcal{I}$ and $\mathcal{B}$ respectively and the data $\mathcal{D}$.

The loss function comprises four main components:

- **Data Loss**($\mathcal{L}_{\mathcal{D}}$): This component measures the discrepancy between the neural network's predictions and actual observed data. It is crucial for fitting the model to observations and is defined as:

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{NN}(x_i, t_i; \theta) - u_i \right\|^2, \tag{3.2}$$

  where $\{(x_i, t_i, u_i)\}_{i=1}^{N}$ represents observed data points with u_i being the observed values at spatial location x_i and time t_i, and $\mathbf{NN}(x_i, t_i; \theta)$ is the output of the neural network for parameters θ.

- **Physics loss**($\mathcal{L}_{\mathcal{N}}$): This component ensures that the neural network's output satisfies the differential equation across the domain Ω. It is defined as:

$$\mathcal{L}_{\mathcal{N}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathcal{N}(\mathbf{NN}(x_i, t_i; \theta); \boldsymbol{\gamma}) - f(x_i, t_i) \right\|^2 \tag{3.3}$$

- **Initial condition loss**($\mathcal{L}_{\mathcal{I}}$): This measures the discrepancy between the neural net-

work predictions and the specified initial conditions:

$$\mathcal{L}_{\mathcal{I}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathcal{I}(\mathbf{NN}(x_i, 0; \theta); \boldsymbol{\gamma}) - g(x_i) \right\|^2 \tag{3.4}$$

- **Boundary condition loss** ($\mathcal{L}_{\mathcal{B}}$): This ensures that the neural network respects the boundary conditions on ($\partial\Omega$):

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathcal{B}(\mathbf{NN}(x_i, t_i; \theta); \boldsymbol{\gamma}) - h(x_i, t_i) \right\|^2. \tag{3.5}$$

The total loss ($\mathcal{L}_{total}$) is a combination of these components, possibly with additional terms for any available data points:

$$\mathcal{L}_{total} = \lambda_{\mathcal{N}}\mathcal{L}_{\mathcal{N}} + \lambda_{\mathcal{I}}\mathcal{L}_{\mathcal{I}} + \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + \lambda_{\mathcal{D}}\mathcal{L}_{\mathcal{D}}, \tag{3.6}$$

where $\lambda_{\mathcal{D}}, \lambda_{\mathcal{N}}, \lambda_{\mathcal{I}}$, and $\lambda_{\mathcal{B}}$ are weighting factors.

In the above equations, **NN** can represents any type of architecture such as CNNs, RNNs. However in case of fully connected network, this can be represented by the composite equation below,

$$\mathbf{x}^j = \sigma^j(\mathbf{W}^j \cdot \mathbf{x}^{j-1} + \mathbf{b}^j), \qquad j \in \{0, ..., L\} \tag{3.7}$$

where j is the layer number, $\sigma^j : \mathbb{R}^n \mapsto \mathbb{R}^n$ is the activation function which adds non-linearity to the NN, and $\mathbf{W}^j$ and $\mathbf{b}^j$ are weights and biases of the specific layer

For example, a 4-layer neural network, i.e. $L = 4$, can be represented by

$$\begin{aligned} \mathbf{x}^1 &= \sigma^1\left(\mathbf{W}^1 \cdot \mathbf{x}^0 + \mathbf{b}^1\right) \\ \mathbf{x}^2 &= \sigma^2\left(\mathbf{W}^2 \cdot \mathbf{x}^1 + \mathbf{b}^2\right) \\ \mathbf{x}^3 &= \sigma^3\left(\mathbf{W}^3 \cdot \mathbf{x}^2 + \mathbf{b}^3\right) \\ \mathbf{x}^4 &= \mathbf{W}^4 \cdot \mathbf{x}^3 + \mathbf{b}^4 \end{aligned} \tag{3.8}$$

where $\mathbf{x}^0$ is the input and $\mathbf{x}^4$ is the output.

In the forward problem, the PINN is trained to predict the solution $u(\mathbf{x}, t)$ by minimizing $\mathcal{L}_{total}$, given the parameters γ, source term f, and the conditions imposed by $(\mathcal{I})$ and $(\mathcal{B})$.

In the inverse problem, the challenge is to infer the source term (f) or the parameters $\boldsymbol{\gamma}$, given observations or measurements of u, along with the initial and boundary conditions. This requires adjusting the training process of the PINN to solve for $\boldsymbol{\gamma}$ or f that minimize the discrepancy between the observed data and the model's predictions, effectively turning the neural network into a tool for discovery and characterization of the underlying physical processes.

PINNs offer a versatile and powerful framework for addressing both forward and inverse problems in engineering and physics, leveraging the ability of neural networks to approximate complex functions and enforce compliance with physical laws through their training process [13].

3.1.3 Duffing's equation

Duffing's equation is a non-linear second-order differential equation used to model various physical phenomena, including the dynamics of damped and driven oscillators. The equation takes its name from German engineer George Duffing, who studied its properties in the early 20^{th} century. It is a quintessential example of a system that exhibits complex behavior, including chaos, under certain conditions [116].

Duffing's equation can model the behavior of a wide array of physical systems across

different scales and contexts. Among its notable applications are nano-mechanical resonators, as explored by Antonio et al. [117], which showcases its relevance in understanding the dynamics at the nano-scale. Similarly, in the analysis of ultrasonic cutting systems, a subject of preliminary investigation by Lim et al. [118]. Furthermore, Duffing's equation plays a vital role in understanding the interactions within piezo-ceramic materials subjected to electric fields [119]. Its utility is further evidenced by its application to biological systems, such as the complex mechanisms of insect flight motors discussed in [120]. These diverse applications underscore its ability to capture the complex dynamics of systems influenced by non-linear forces and external perturbations. For an in-depth exploration of Duffing's equation, readers are encouraged to consult [116]. It is expressed as,

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = f(t), \tag{3.9}$$

where $\ddot{x}$ represents the acceleration, $\dot{x}$ the velocity, x the displacement, δ the damping coefficient, β governs the degree of non-linearity, and $f(t)$ the external driving force. Further, the initial conditions are specified by $x(0) = x_0$ and $\dot{x}(0) = \dot{x}_0$, with δ representing damping, α the linear stiffness, β the non-linearity degree, and $f(t)$ the external driving force.

The non-linearity arises due to the x^3 term, which introduces complex dynamics not present in linear systems. Depending on the values of α, β and δ the Duffing oscillator can exhibit a wide range of behaviors from simple harmonic motion to chaotic dynamics.

This study focuses on determining $f(t)$ from the observations of $x(t)$, alongside initial conditions, by utilizing an artificial neural network (ANN) and the underlying equation. This approach contrasts with traditional forward-solving methods, which involve obtaining $x(t)$ from given $f(t)$ and initial conditions through analytical or numerical solutions.

3.2 Literature review

This dissertation work delves into a specific ISP, the dynamic load identification problem, aiming to infer the 'forcing function' or 'excitation force' of linear and non-linear oscillators

based on dynamic response data.

Over recent decades, extensive research has been conducted on diverse strategies for resolving this issue, with numerous significant contributions. Huang [121] employed a conjugate gradient approach for estimating time-dependent forces in non-linear oscillators. Ma et al. [122] introduced a recursive estimator using the Kalman filter for impulsive load determination from data for systems with single and multiple degrees of freedom. Azam et al. [123] suggested a dual Kalman filter for full state estimation of linear multi-degree freedom systems with unknown inputs, based on limited noisy acceleration data and a known physical model. [124] addressed the force identification problem in duffing oscillators through a Volterra-type integral equation, employing regularization for solution stabilization. Feldman [125] proposed force prediction from response data alone, without requiring parametric or governing equation knowledge, using the Hilbert transform. [126] tackled the non-linear force identification issue in the frequency domain through ordinary least squares and Tikhonov regularization. Liu et al. [127] addressed the non-linear vibration challenge by converting non-linear differential equations into more noise-resistant parabolic equations. Rice et al. [128] introduced a calibration-based integral approach for force function estimation in spring mass damper systems from response data. For a comprehensive review on dynamic load identification methodologies, readers are directed to [129].

Machine learning and deep learning have recently garnered attention for load identification, with Pravin and Rao [130] utilizing dynamic principal component analysis for input force recovery from acceleration time series. Zhou et al.[131] applied deep Recurrent Neural Networks (RNNs), including Long Short Term Memory (LSTM) variants, for impact load recovery on non-linear structures. Another study [132] explored RNNs for force recovery on beam structures under various excitation types. Rosafalco et al. [133] deployed a deep learning autoencoder for load identification in structural health monitoring, incorporating residual learning and inception modules. [134] presented an Artificial Neural Network (ANN) based on the Bayesian Probability Framework for displacement response-based force estimation.

Despite deep learning's substantial success across various domains, its solutions sometimes lack physical consistency and exhibit poor generalization [135]. This issue can be mitigated by integrating governing equations into neural network loss functions, as seen in "physics-informed neural networks (PINNs)" [111]. PINNs have been applied to inverse source problems, including He et al.'s work [136] on predicting varying heat sources from temperature data with notable accuracy. Our study employs PINNs to estimate forcing functions for one degree of freedom systems.

Recent works by Liu et al. [137] and Haghighat et al. [138] have combined machine learning with physics-based methods for vibration analysis. The latter employed PINNs for both forced vibration and plate vibration problems. Unlike Haghighat et. al., who focused on predictions of displacement and natural frequencies, the initial part of this study emphasizes excitation force estimation from noisy observational data using PINNs. Furthermore, the second part of this work focuses on the development of a physically consistent surrogate model that can predict forcing functions solely from noisy displacement data and initial conditions, without the need for physical parameters and repeated inverse solutions.

Although the primary focus is on mechanical oscillators, this methodology has a potential to applied to other system governed by linear or non-linear differential equations across other fields.

3.3 Methodology

Initial part of this work was focused on utilizing PINNs to estimate forcing function from the displacement and velocity data later leading up to the development of surrogate model which can predicting forcing functions from given displacement history and initial condition in a non parametric way. The methodology adopted to tackle each of these problems is discussed in detail in the following sections:

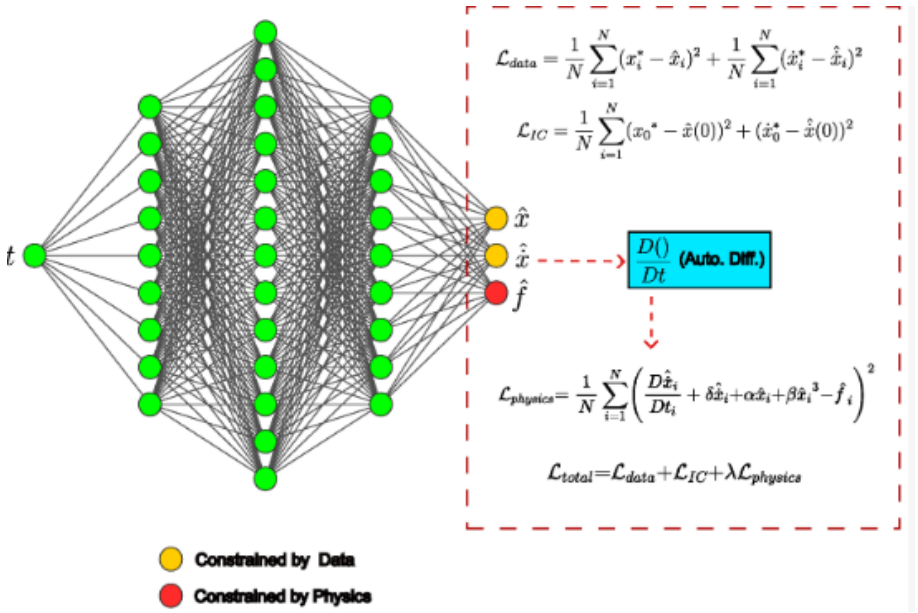


Figure 3.2: Proposed neural network architecture with input t and output $\hat{x}$, $\hat{\dot{x}}$ and $\hat{f}$ [10]

3.3.1 Estimating the $f(t)$ from $x(t)$, $\dot{x}(t)$ and initial conditions using PINNs

In this part of the study the emphasis was to develop a PINNs model that can recover different types of forcing function, from a given displacement, velocity histories and initial conditions, for both linear and non-linear oscillators. The mathematical underpinnings, training algorithm, choice of hyper-parameters are discussed in detail in the following subsections.

3.3.1.1 Neural network architecture

The architecture of the neural network is depicted in Figure 3.2 and is mathematically described as

$$\hat{f}, \hat{x}, \hat{\dot{x}} = \Phi^L(t; \mathbf{W}, \mathbf{b}) \tag{3.10}$$

where $\Phi^L : \mathbb{R}^+ \mapsto \mathbb{R}^3$ denotes the neural network characterized by L layers. Here, $t \in \mathbb{R}^+$ serves as the input, while the outputs are denoted by $\hat{x} \in \mathbb{R}$, $\hat{\dot{x}} \in \mathbb{R}$, and $\hat{f} \in \mathbb{R}$. The parameters of the neural network are represented by $\mathbf{W} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. To facilitate

the differentiation of neural networks output with respect to its input this architecture was used. Further, such differentiation is carried out through Automatic Differentiation (AD), utilizing functions from the TensorFlow [139] library.

The NN output $\hat{f}$ is constrained by the governing equation, further, the $\hat{x}$, $\hat{\dot{x}}$ are constrained by the displacement and velocity measurements respectively. A detailed explanation of this is provided in the section 3.3.1.2.

The proposed architecture was built using Keras[140] library with a TensorFlow backend. It NN was made up of $L = 10$ layers with [1,15,30,60,120, 240,120,60,30,15,3] units each. Additionally, each dense layer is passed through eLU activation function adding non-linearity to the architecture.

The best working hyper-parameters were identified by exploring various layer configurations, weight initialization, activation functions, learning rates and epochs. Initially, a shallow network featuring fewer trainable parameters and utilizing the ReLU activation function was developed which didn't provide good results. Further, eLU activation function was incorporated in the architecture providing further improvements. Consequently, a deeper architecture utilizing the eLU function was adopted. This methodological approach was also applied to determine other optimal hyper-parameters, with further details on the selected hyper-parameters presented in the section 3.3.1.3

3.3.1.2 Loss Function

The backbone of the approach lies in the definition of the neural network's loss function. The overall loss, denoted as $\mathcal{L}_{total}$, is the sum of the data loss $\mathcal{L}_{data}$, the initial condition loss $\mathcal{L}_{IC}$, and the loss associated with physics, $\mathcal{L}_{physics}$, expressed as:

$$\mathcal{L}_{total} = \mathcal{L}_{data} + \mathcal{L}_{IC} + \lambda \mathcal{L}_{physics} \tag{3.11}$$

such that

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{i=1}^{N}(x_i^* - \hat{x}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(\dot{x}_i^* - \hat{\dot{x}}_i)^2 \tag{3.12}$$

and

$$\mathcal{L}_{IC} = (x_0 - \hat{x}(0))^2 + (\dot{x}_0 - \hat{\dot{x}}(0))^2. \tag{3.13}$$

In this context, x_i^* and $\dot{x}_i^*$ represent the displacement and velocity obtained from the dataset, while $\hat{x}_i$ and $\hat{\dot{x}}i$ are the displacement and velocity as predicted by the neural network. The term λ is the regularization parameter, and x_0 and $\dot{x}_0$ refer to the initial conditions. The roles of $\mathcal{L}_{data}$ and $\mathcal{L}_{IC}$ are to ensure that the neural network's predictions adhere to the data.

The term for physics loss, $\mathcal{L}_{physics}$, is crucial for integrating physics into the neural network's structure, represented as follows:

$$\mathcal{L}_{physics} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{D\hat{\dot{x}}_i}{Dt_i} + \delta\hat{\dot{x}}_i + \alpha\hat{x}_i + \beta\hat{x}_i^3 - \hat{f}_i\right)^2. \tag{3.14}$$

This formula was derived by modifying equation 3.9 to incorporate the neural network's displacement and velocity estimations. To calculate the derivative of velocity with respect to time, automatic differentiation was utilized, denoted by $\frac{D}{Dt_i}$. The purpose of $\mathcal{L}_{physics}$ is to ensure that the predicted $\hat{f}_i$ aligns with the underlying physical laws.

3.3.1.3 Training

The goal of the proposed neural network architecture is to recover the forcing function, $f(t)$, using displacement and velocity measurement data. The procedure for training is detailed below (see algorithm 1). The inputs for this process are t, x^*,$\dot{x}^*$, which represent time, displacement, and velocity, respectively. The NN processes t and yields $\hat{f}, \hat{x}, \hat{\dot{x}}$, which are the estimates for the forcing function, displacement, and velocity. The initialization of the neural network's weights is carried out using the he-normal initialization.

Algorithm 1 Training Algorithm

Require: t, x^*, $\dot{x}^*$
Ensure: $\mathcal{L}_{total} \to 0$
  $n \leftarrow$ no. of epochs
  $\eta \leftarrow$ learning rate
  $N \leftarrow$ batch size
  $\lambda \leftarrow$ regularization
  while $n > 0$ **do**
    $\hat{f}, \hat{x}, \hat{\dot{x}} \leftarrow \Phi^L(t; \mathbf{W}, \mathbf{b})$
    $\mathcal{L}_{data}, \mathcal{L}_{IC}, \mathcal{L}_{physics}$ ▷ This is calculated using (3.12)–(3.14)
    $\mathcal{L}_{total} \leftarrow \mathcal{L}_{data} + \mathcal{L}_{IC} + \lambda \mathcal{L}_{physics}$
    $\mathbf{W}^*, \mathbf{b}^* \leftarrow \text{Adam}(\eta, \mathcal{L}_{total})$
    $\mathbf{W}, \mathbf{b} \leftarrow \mathbf{W}^*, \mathbf{b}^*$
    $n \leftarrow n - 1$
  end while

The neural network was trained with 500 data points, divided into batches of 250, on an NVIDIA GTX 2060 GPU over 60,000 epochs. The approximate duration for training spanned approximately from 3 to 3.5 hours. The learning rate, denoted as η, was set to 0.001, and the parameter, λ, was adjusted to 0.1 or 0.01 to achieve accurate results.

During each epoch, the total loss, denoted by $\mathcal{L}_{total}$, was calculated from the dataset and the predictions were made by the neural network. Subsequently, the Adam optimizer [141] was utilized to compute its gradients relative to the parameters of the neural network. These gradients are then propagated throughout the network via the back-propagation algorithm. This algorithm leverages these gradients to update the network's weights and biases at each epoch. The evolution of $\mathcal{L}_{total}$, along with $\mathcal{L}_{data}$ and $\mathcal{L}_{physics}$, over the epochs for a particular training scenario is depicted in figure 3.3. Ideally, for the neural network to learn successfully $\mathcal{L}_{total} \to 0$, which can be observed in the figure.

3.3.2 Surrogate model using PINNs for prediction of $f(t)$ from $x(t)$ and $\dot{x}_0$

The previous method entailed estimating the forcing function $f(t)$ based on the observed $\dot{x}(t)$ and $x(t)$ data for specific instances using PINN. This process resembled solving an optimization problem. However, this approach might fall short in predicting the forcing function when presented with new sets of response data.

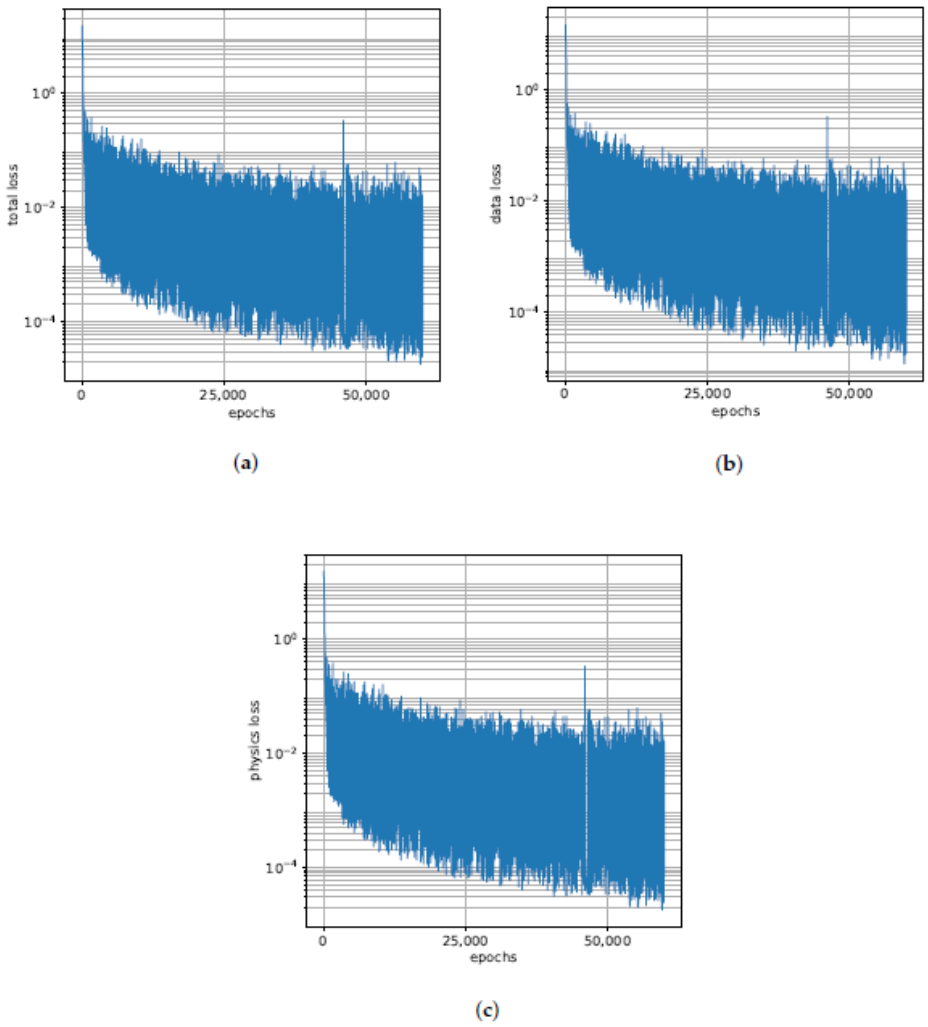


Figure 3.3: Figure (a)-(c) shows the total, data, and physics loss w.r.t the training epochs for one training instance [10]

This section elaborates on the technique utilized in constructing a PINN model capable of inferring the forcing function $f(t)$ directly from the available $x(t)$ measurements and the initial conditions.

The surrogate model was developed for the linear case i.e. $\beta = 0$. Further, with this modification the Equation 3.9 becomes an equation of simple harmonic oscillator with $\alpha = k/m$, $\delta = c/m$ and $f(t) = \gamma \sin(\omega t)$. It is given below as:

$$\ddot{x} + \delta\dot{x} + \alpha x = \gamma \sin(\omega t), \tag{3.15}$$

where m, k, c, γ and ω are the mass, spring stiffness, damping coefficient, amplitude and angular frequency of the simple harmonic oscillator a.k.a spring mass damper system.

Following the theory of spring mass damper system, ζ represents as damping ratio which is given by:

$$\zeta = \frac{c}{2\sqrt{km}}. \tag{3.16}$$

To be consistent with the notations following equation 3.9 and 3.15 the damping ratio can be rearranged giving:

$$\zeta = \frac{\delta}{2\sqrt{\alpha}}. \tag{3.17}$$

This dimensionless parameter governs the oscillatory behaviour of Equation 3.15. When $0 < \zeta < 1$ the system is known as under-damped. Further, when $\zeta = 1$ system is known as critically damped. Similarly, when $\zeta > 1$ system is known as an over-damped system.

The details of the methods used for development of surrogate model are further discussed in the following subsections.

3.3.2.1 Synthetic data generation

The synthetic data was generated by numerically the solving differential equation 3.15 using '**ode-int**' function of the SciPy library [142]. For different combination of parameters set compromising of ζ,γ, ω, $\dot{x}_0$, and x_0. This parameter set was created by employing Latin Hypercube (LH) sampling, which ensure uniform coverage of the parameter space, on the parameter ranges as follows: For the numerical solution of the differential equation across

Table 3.1: Parameter ranges used for synthetic data generation.

Parameter	**Range**
damping ratio (ζ)	[0.5, 1.5]
initial displacement (x_0)	[-1, 1]
initial velocity ($\dot{x}_0$)	[-1, 1]
angular frequency (ω)	[0.2π, 0.6π]
amplitude (γ)	[0.1 2]

all parameter sets, constants were set to $m = 1$, $k = 1$, implying $\alpha = 1$, and the simulation

duration was fixed at 20 seconds. The total dataset was compromised of 300 samples with each sample consisting of $\mathbf{x}$, $\dot{\mathbf{x}}$, $\mathbf{f}$ vectors of size 100. Further, with intial conditions and its parameter set.

3.3.2.2 Neural network architecture

The structure of the neural network is mathematically represented as follows:

$$\hat{\mathbf{f}}, \hat{\dot{\mathbf{x}}} = \Phi^L(\mathbf{x}, \dot{x}_0; \mathbf{W}, \mathbf{b}) \tag{3.18}$$

where Φ^L symbolizes the neural network comprised of L layers. Here, $\mathbf{x} \in \mathbb{R}^{1\times n}$ and $\dot{x}_0 \in \mathbb{R}^{1\times 1}$ represent the displacement input vector and the initial condition for velocity, respectively. The model's outputs include the history of the forcing function $\hat{\mathbf{f}} \in \mathbb{R}^{1\times n}$ and the velocity vector $\hat{\dot{\mathbf{x}}} \in \mathbb{R}^{1\times n}$ with n is number of points. This neural network configuration was defined to get rid of double differentiation with respect to t and enable the differentiation of $\hat{\dot{\mathbf{x}}}$ with respect to $\mathbf{x}$.

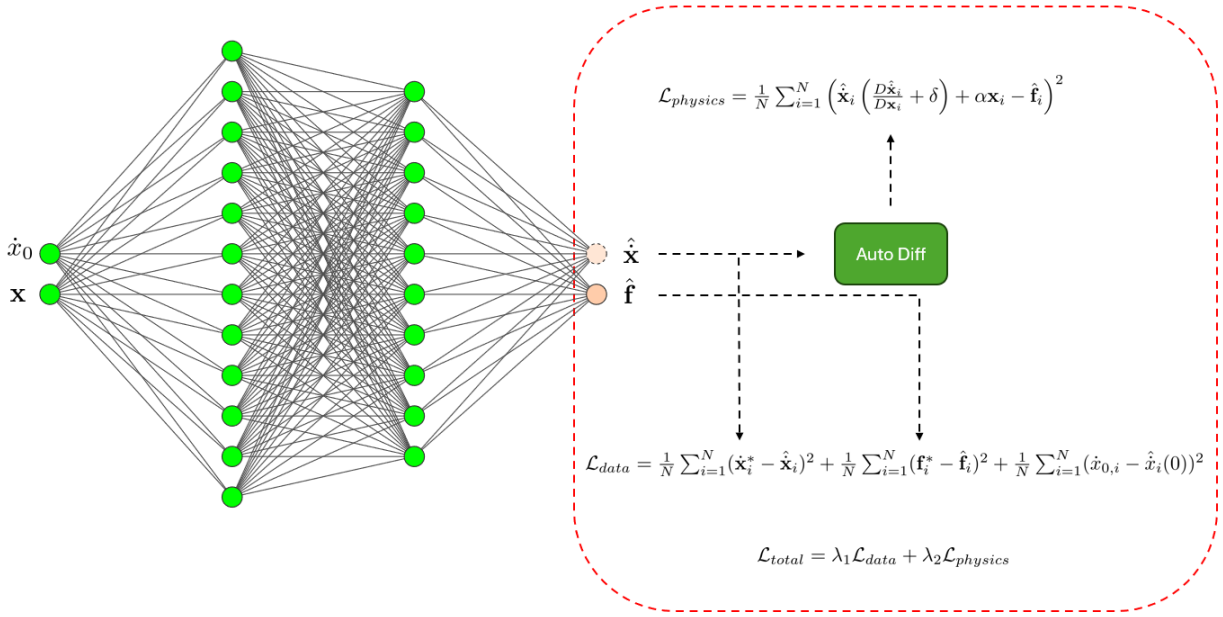


Figure 3.4: Proposed architecture for surrogate model

The above architecture was built using Keras library with a TensorFlow backend. The NN was made up of 4 layers with 20 units each. Additionally, each dense layer was passed through adaptive tanh activation function which was built using custom layer functionality of TensorFlow. The adaptive tanh function utilizes an additional trainable parameter governing the slope of activation function that is updated during the training process. This leads to improved PINNs performance and convergence during training [143].

The best working hyper-parameters were identified by exploring various layer configurations, weight initialization, different activation functions including their adaptive variant, combinations of weights of physics and data terms, learning rates and epochs.

3.3.2.3 Loss function

The formulation of the loss function used for the surrogate modeling tasks is detailed below. The overall loss, $\mathcal{L}_{total}$ is composed of the data term $\mathcal{L}_{data}$, and the physics loss term $\mathcal{L}_{physics}$:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{data} + \lambda_2 \mathcal{L}_{physics} \tag{3.19}$$

such that

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{i=1}^{N}(\dot{\mathbf{x}}_i^* - \hat{\dot{\mathbf{x}}}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{f}_i^* - \hat{\mathbf{f}}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(\dot{x}_{0,i} - \hat{\dot{x}}_i(0))^2, \tag{3.20}$$

and

$$\mathcal{L}_{physics} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\dot{\mathbf{x}}}_i\left(\frac{D\hat{\dot{\mathbf{x}}}_i}{D\mathbf{x}_i} + \delta\right) + \alpha\mathbf{x}_i - \hat{\mathbf{f}}_i\right)^2. \tag{3.21}$$

In these equations, $\dot{\mathbf{x}}_i^*$ and $\mathbf{f}_i^*$ represent the observed velocity and forcing function values, respectively, while $\hat{\dot{\mathbf{x}}}_i$ and $\hat{\mathbf{f}}_i$ are their neural network predictions. The coefficients λ_1 and λ_2 are weighting factors that balance the contribution of the data fidelity term and the physics-based term to the total loss, ensuring that the model predictions are both accurate and physically plausible. The term $\dot{x}_0$ specifies the initial condition for velocity, which is

crucial for accurately capturing the system dynamics from the start.

Moreover, the term $\frac{D}{D\mathbf{x}_i}$ signifies the process of automatic differentiation with respect to $\mathbf{x}$. This approach to differentiation was chosen to eliminate the requirement for double differentiation with respect to time t , to ensure stability during the training phase.

3.3.2.4 Training

The proposed architecture was trained on the dataset that was generated following process described in section 3.3.2.1. The dataset was divided into training, testing and validation sets following the 80%-10%-10% split. The input to the training process was $\mathbf{x}^*$ and $\dot{x}_0^*$ further, the output was $\hat{\mathbf{f}}$ and $\hat{\dot{\mathbf{x}}}$. The algorithm used for training is detailed below. The network was trained in batch sizes of 20 data points for 40,000 epochs on NVIDIA Tesla V100 GPU. The training spanned for about 6-7 hrs per 10,000 epochs approximately. Further, the learning rate η was set to 0.0008, the weights were initialized using "glorot uniform" initialization technique, and the data term and physics term contributions i.e. λ_1 and λ_2 in the loss function was set to 0.5 each.

Algorithm 2 Training Algorithm

Require: $\mathbf{x}^*$, $\dot{x}_0$
Ensure: $\mathcal{L}_{total} \to 0$
  $n \leftarrow$ no. of epochs
  $\eta \leftarrow$ learning rate
  $N \leftarrow$ batch size
  while $n > 0$ **do**
    $\hat{\mathbf{f}}, \hat{\dot{\mathbf{x}}} = \Phi^L(\mathbf{x}, \dot{x}_0; \mathbf{W}, \mathbf{b})$
    $\mathcal{L}_{data}, \mathcal{L}_{physics}$ ▷ This is calculated using 3.20, and 3.21
    $\mathcal{L}_{total} \leftarrow \lambda_1 \mathcal{L}_{data} + \lambda_2 \mathcal{L}_{physics}$
    $\mathbf{W}^*, \mathbf{b}^* \leftarrow \text{Adam}(\eta, \mathcal{L}_{total})$
    $\mathbf{W}, \mathbf{b} \leftarrow \mathbf{W}^*, \mathbf{b}^*$
    $n \leftarrow n - 1$
  end while

In each training epoch, the composite loss, represented as $\mathcal{L}_{total}$, was calculated for every batch size using the neural network's output. The Adam optimizer was then employed to determine the gradients in relation to the neural network's parameters. Following this, the

gradients were propagated across the network through the back-propagation algorithm. This algorithm utilizes the gradients to update the network's weights and biases with each epoch. Ideally, to ensure learning, $\mathcal{L}_{total}$ should approach towards 0.

3.4 Results

This section outlines the comprehensive findings from our investigation into the application of PINNs for identifying forcing functions from response data.

3.4.1 Recovering forcing function using PINNs

3.4.1.1 Linear case

The Equation 3.9 was transformed into a linear ODE by setting $\beta = 0$. By further setting $\alpha = k/m$, $\delta = c/m$, and adjusting $f(t) = f(t)/m$, the equation was simplified to resemble a spring-mass-damper system, characterized by a mass m, damping coefficient c, and spring stiffness k. In the discussions that followed, the values were set to $m = 1$, $c = 0.2$, and $k = 0.9$, which correspond to $\alpha = 0.9$ and $\delta = 0.2$. Data was then generated by solving the linear ODE when subjected to sinusoidal, piece-wise, and step forcing functions.

The neural network was trained using these generated datasets to assess its capability in inferring the forcing functions from the data. The subsequent sections detail the findings obtained after the training process.

Sinusoidal function

To assess the neural network's capability in accurately reconstructing a smooth, periodic function, it was trained using data generated from the application of a harmonic force to the spring-mass-damper system. This force was described by

$$f(t) = \gamma \cos \omega t \tag{3.22}$$

with the initial conditions $x_0 = 4.9, \dot{x}_0 = -2.2$, frequency $\omega = 0.4$, and amplitude $\gamma = 3$. The results obtained were promising, as depicted in figure 3.5(a). An excellent agreement

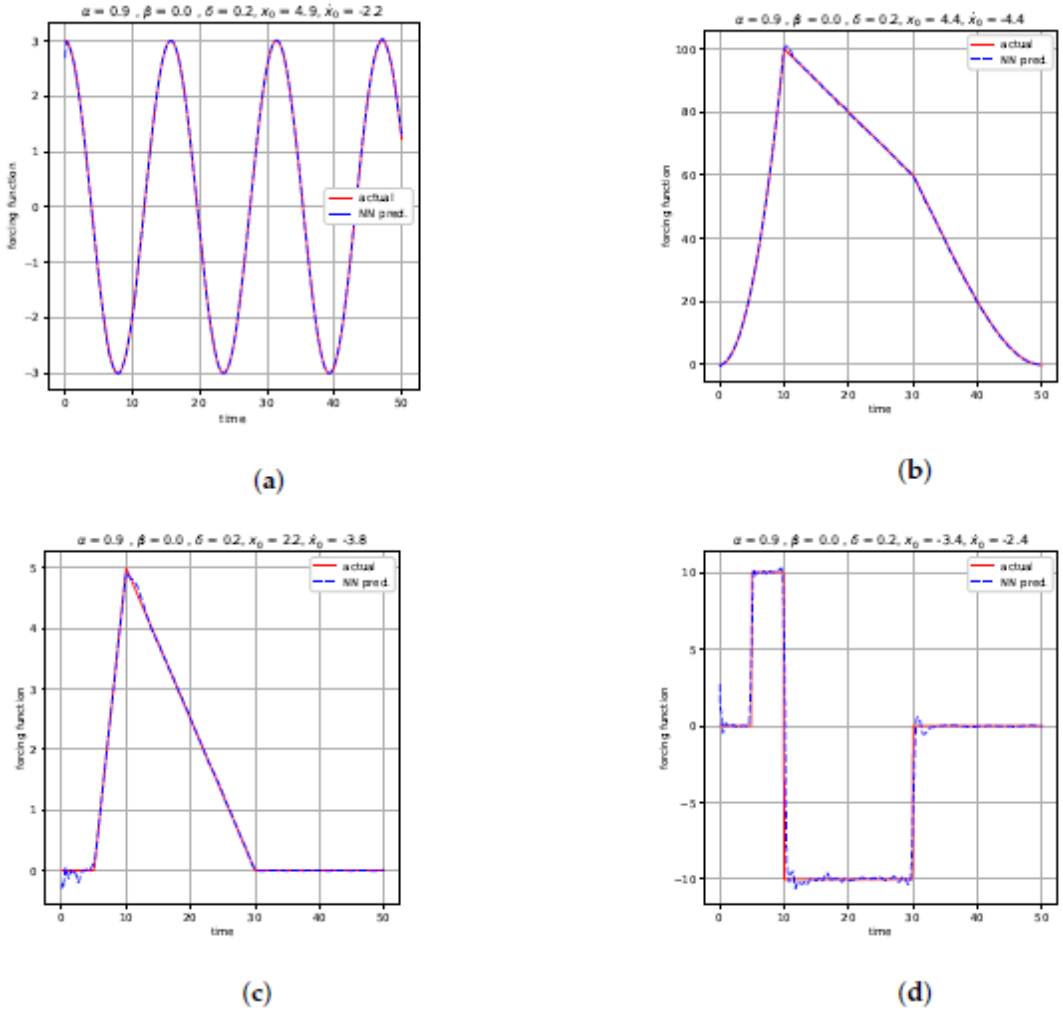


Figure 3.5: Predictions for (a) sinusoidal (b) combination of parabolic, linear and cubic (c) triangular (d) step functions [10]

between the neural network's predictions and the actual function was observed

Piece-wise function

In this part of the study, the spring mass damper system was subjected to piece-wise forcing functions as represented by the equations,

$$f(t) = \begin{cases} 0 & 0 \leq t < 5 \\ t - 5 & 5 \leq t < 10 \\ \frac{5}{20}(30 - t) & 10 \leq t < 30 \\ 0 & 30 \leq t \leq 50 \end{cases} \tag{3.23}$$

and,

$$f(t) = \begin{cases} t^2 & 0 \leq t < 10 \\ -2t + 120 & 10 \leq t < 30 \\ \frac{1}{200}t^3 - \frac{1}{2}t^2 + \frac{25}{2}t & 30 \leq t \leq 50 \end{cases} \tag{3.24}$$

where the first equation describes a triangular function, which is essentially a sequence of linear increases and decreases over specific intervals, and the second equation combines parabolic, linear, and cubic functions. These functions are defined by their sudden changes in the gradient magnitudes.

Figures 3.5 (b) and (c), illustrates the finding where the network was trained using data generated by simulating the spring mass damper system under initial conditions $x_0 = 4.4, \dot{x}_0 = -4.4$, and $x_0 = 2.2, \dot{x}_0 = -3.8$, subjected to forces depicted by equations 3.24 and 3.23, respectively.

It was observed that the actual forcing functions and the neural network predictions matched quite closely. However, the network encountered some difficulties in accurately predicting values at the transitions of both functions. Specifically, for the triangular function, it tended to under-predict, while for the composite of linear, parabolic, and cubic functions, it

tended to over-predict. Furthermore, during the interval where the function's value was zero, marking the start of the triangular function, some oscillations in the network's predictions were observed.

Step function

After evaluating the architecture's ability to identify piece-wise forcing functions in the preceding section,in the current section, an attempt was made to solve much more challenging problem. The aim of this part of the work was to ascertain whether the architecture would be able to recover the functions characterized by discontinuities. To address this the neural network was trained on data generated from simulation the spring mass damper system with initial conditions $x_0 = -3.4, \dot{x}_0 = -2.4$ and the following step function,

$$f(t) = \begin{cases} 0 & 0 \leq t < 5 \\ 10 & 5 \leq t < 10 \\ -10 & 10 \leq t < 30 \\ 0 & 30 \leq t \leq 50 \end{cases} \tag{3.25}$$

step functions are characterized by maintaining constant values over certain intervals, then exhibiting abrupt changes at points of transition.

Figure 3.5 (d) displayed our results. It was evident that the neural network was able to recover the majority of the forcing function from the data. The constant segments of the step function were matched with oscillatory predictions. These oscillations resembles the Gibbs phenomenon, commonly encountered when jump discontinuities are approximated by the Fourier series.

3.4.1.2 Non-linear case

Following the success with linear ODEs in the preceding sections, the performance of proposed architecture was evaluated against non-linear ODEs. The equation 3.9 was subjected

to a range of smooth forcing functions, such as sinusoidal, a combination of two sinusoidal, and impulse functions. Details and results of the numerical experiments are presented in the subsequent subsections.

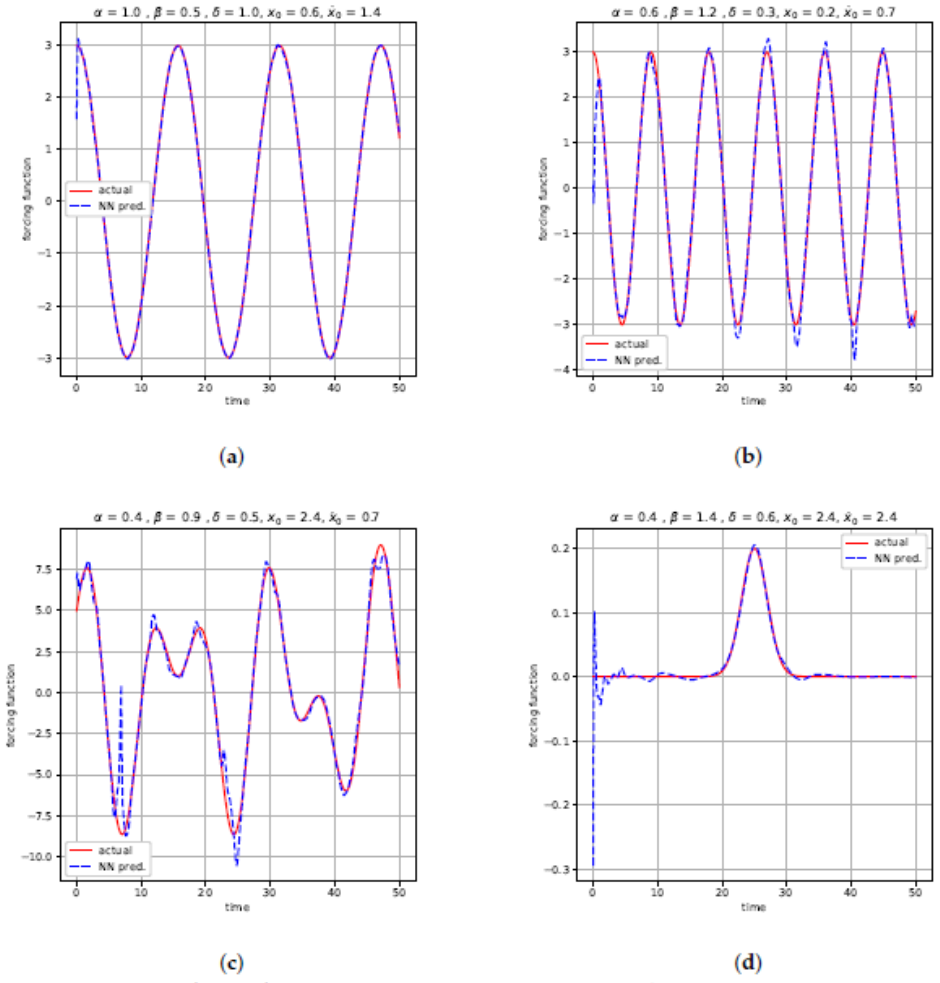


Figure 3.6: Predictions for (a) sinusoidal (b) sinusoidal with increased non-linearity and frequency (c) sum of two sinusoidal (d) impulse functions for non linear case [10]

Sinusoidal Function

The governing equation (3.9) was solved by applying a sinusoidal forcing function defined as,

$$f(t) = \gamma \cos \omega t \tag{3.26}$$

with parameters set to $\alpha = 1, \delta = 1, \beta = 0.5, \gamma = 3, \omega = 0.4$, and initial conditions $x_0 = 0.6, \dot{x}_0 = 1.4$ to produce response data.

Figure 3.6 (a) illustrates the neural network's performance post-training with the generated data. It was found that the neural network was successful in recovering the sinusoidal function from the response data with minimal issues, although some instability was present at the onset.

The complexity was increased by adjusting the parameters to $\alpha = 0.6, \delta = 0.3, \beta = 1.2, \gamma = 3, \omega = 0.7$, and initial conditions $x_0 = 0.2, \dot{x}_0 = 0.7$. Notably, the non-linearity parameter $\beta = 1.2$ and frequency $\omega = 0.7$ were increased compared to the previous scenario.

The outcomes for this case are depicted in Figure 3.6 (b). While the neural network's predictions mostly aligned well with the actual function, it tended to overestimate at the peaks and troughs, alongside showing some instabilities at the beginning of the function.

Sum of Two Sinusoidal Functions

In this section, the focus was on recovering the forcing function characterized by the sum of two sinusoidal functions, as described by the equation,

$$f(t) = \gamma_1 \cos \omega_1 t + \gamma_2 \cos \omega_2 t. \tag{3.27}$$

The neural network was trained using data generated after setting $\alpha = 0.4, \beta = 0.9, \delta = 0.5$, and solving the duffing equation under the influence of equation 3.27 with $\gamma_1 = 5, \gamma_2 = 4, \omega_1 = 0.4, \omega_2 = 0.7$, and initial conditions $x_0 = 2.4, \dot{x}_0 = 0.7$. This task proved to be more challenging than the previous case that involved a single sinusoidal function. The findings, illustrated in the figure 3.6 (c), revealed that the neural network was able to accurately predict the forcing function from the response data with reasonable accuracy. Although the neural network did under-predict and over-predict at some peaks and troughs of the function, the overall alignment between the actual function and the predictions was nearly perfect.

Impulse function idealized by normal distribution

In the final section, the capability of our network to predict an impulsive function modeled by a normal distribution was assessed. It is detailed as follows:

$$f(t) = \frac{e^{-(t-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}. \tag{3.28}$$

The values $\mu = 25$, and $\sigma = 2$ in equation 3.28 were employed to generate data by simulating equation 3.9 with parameters $\alpha = 0.4, \beta = 1.4, \delta = 0.6$ and initial conditions $x_0 = 1, \dot{x}_0 = -2.2$. This case was explored to test if the network could accurately recover the impact excitation force for non-linear oscillators. To ensure a smooth representation, the impact force was depicted using a normal distribution equation. As illustrated in figure 3.6 (d), the network's predictions initially exhibited numerical oscillations which subsequently subsided, and for the most part, aligned well with the actual forcing function that was utilized to generate the data.

3.4.1.3 Estimating forcing functions from noisy data

As observed in the preceding sections, the neural network demonstrated impressive performance in predicting various types of forcing functions. Nonetheless, it's important to note that in all cases, the data was synthetically generated through numerical simulations of the ODEs. In practical applications, the available data is often corrupted by noise of varying degrees.

This section explores the neural network's ability to predict all the previously discussed forcing functions using data plagued with noise. The following equations were employed to add noise to simulated measurements:

$$\mathbf{x}^*_{\text{noisy}} = \mathbf{x}^* + \xi \max(|\mathbf{x}^*|) \cdot \mathcal{N}(0, 1) \tag{3.29}$$

and,

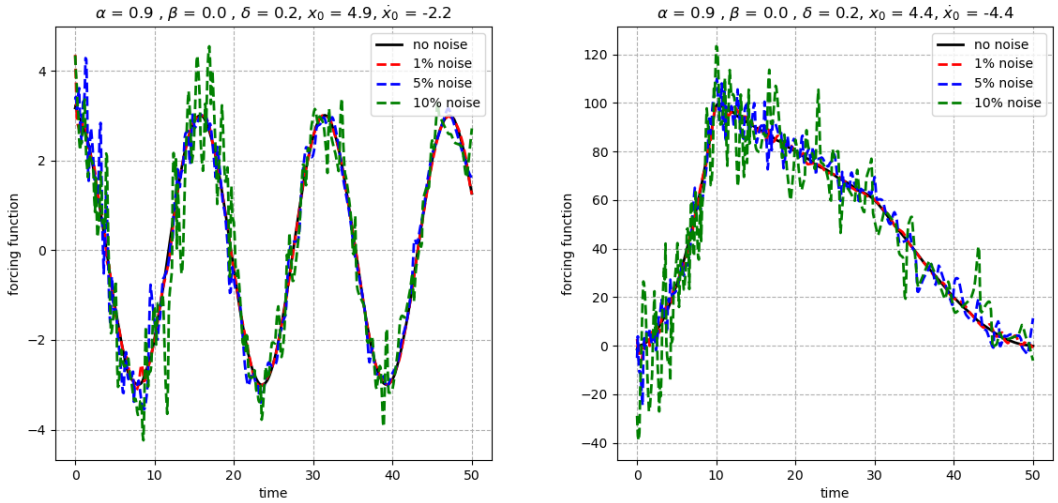
$$\dot{\mathbf{x}^*}_{\text{noisy}} = \dot{\mathbf{x}}^* + \xi \max(|\dot{\mathbf{x}}^*|) \cdot \mathcal{N}(0,1) \tag{3.30}$$

where, ξ represents the percentage of noise, $\mathcal{N}(0,1)$ is the Gaussian random number generator.

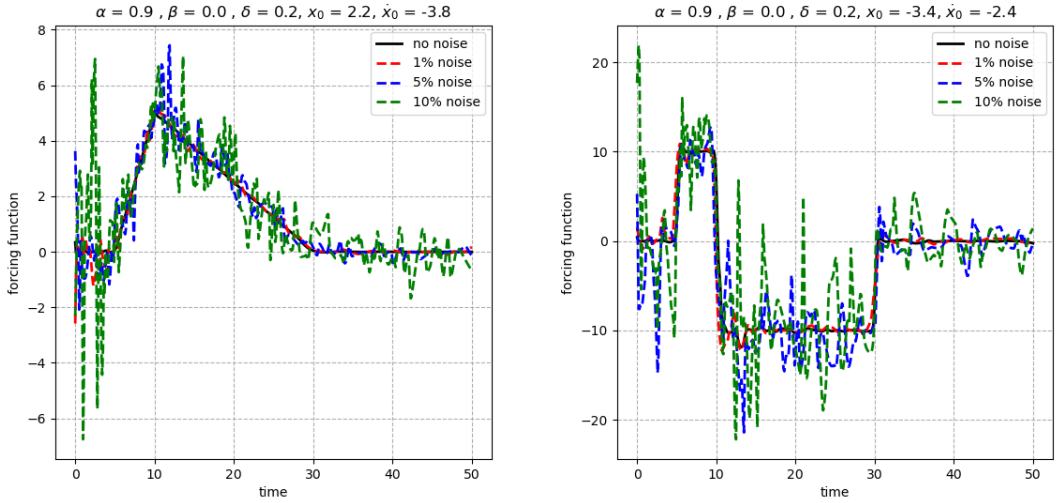
Each of the dataset discussed in the previous section was corrupted $\xi = [1\%, 5\%, 10\%]$ noise. The effect of this varying degree of noise was super-imposed on the predictions without noise. Figure 3.7 shows the prediction results for the linear case. Additionally, figure 3.8 illustrates the findings for non-linear case.

Following the figure 3.7 (a) to (d) the model's predictions for the linear case against data with 1%, 5%, and 10% noise levels. It is observed that:

At 1% noise, the model maintains a high level of accuracy, with all the predicted forcing functions slightly deviating from the noise-free predictions, yet still capturing the essential characteristics. With 5% noise, there is a noticeable deviation from the noise-free predictions, particularly at points where the forcing function exhibits sharp or sudden transitions. Despite this, the model retains a reasonable approximation of the underlying function.



(a) (b)



(c) (d)

Figure 3.7: Figure demonstrating predictions for (a) sinusoidal (b) combination of parabolic, linear and cubic (c) triangular (d) step functions from noisy data

Furthermore, at 10% noise, the model's predictions demonstrate increased deviation from the noise free predictions becomes more pronounced. However, the general trend of the forcing functions is still recognizable, suggesting that the model has a degree of resistance to noise but also highlighting the limits of its predictive capability under higher noise levels.

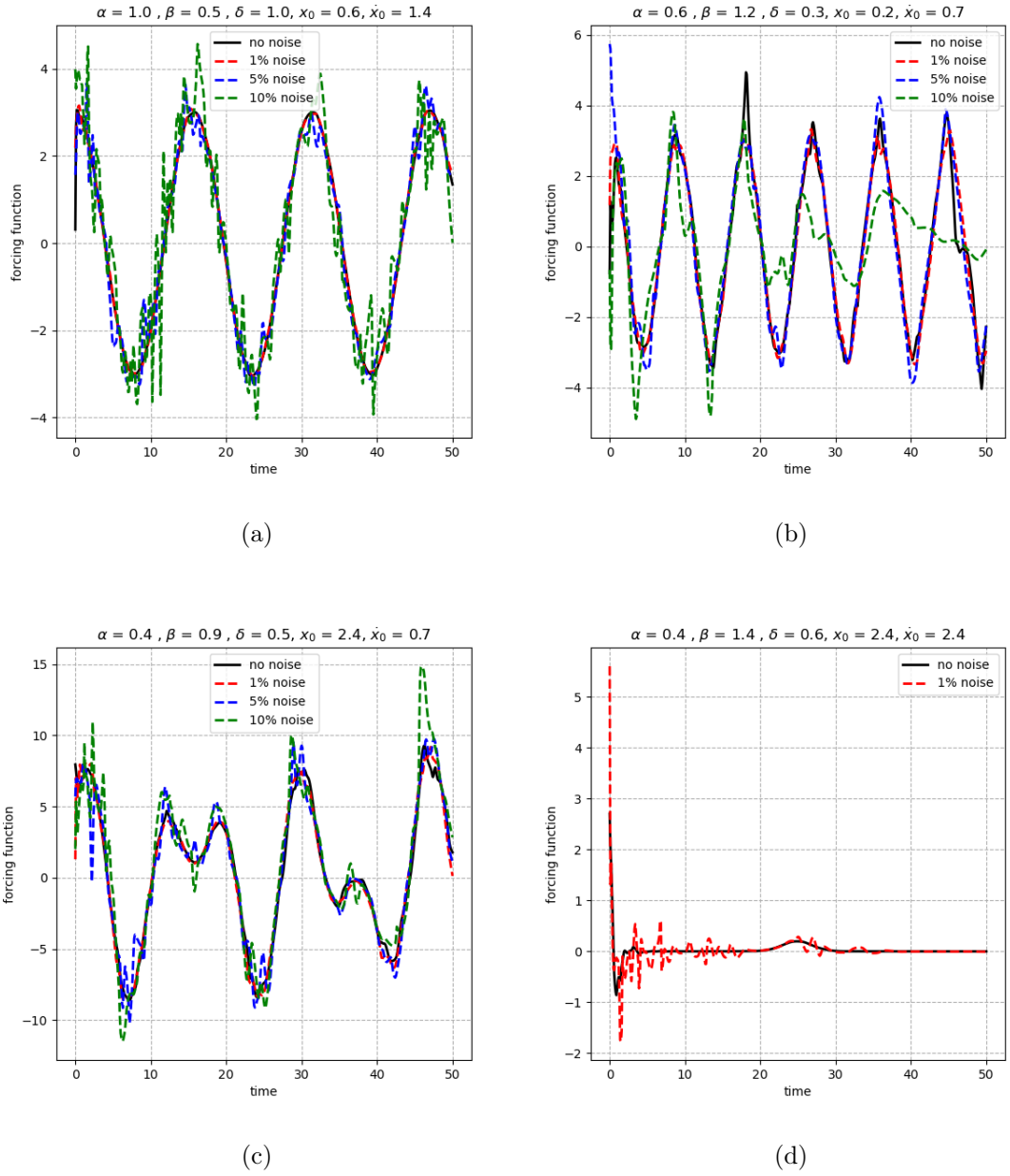


Figure 3.8: Figure demonstrating predictions for (a) sinusoidal (b) sinusoidal with increased non-linearity and frequency (c) sum of two sinusoidal (d) impulse functions from noisy data

For the non-linear case, as illustrated in figure 3.8(a)-(c), the model retains a high degree of agreement at 1% noise. Although slight variations are introduced, the overall shape and key features of the forcing function are well-preserved. At 5% noise, discrepancies between the model's predictions and the actual data become more noticeable. The model struggles

slightly with the increased complexity introduced by the noise but generally follows the trend.

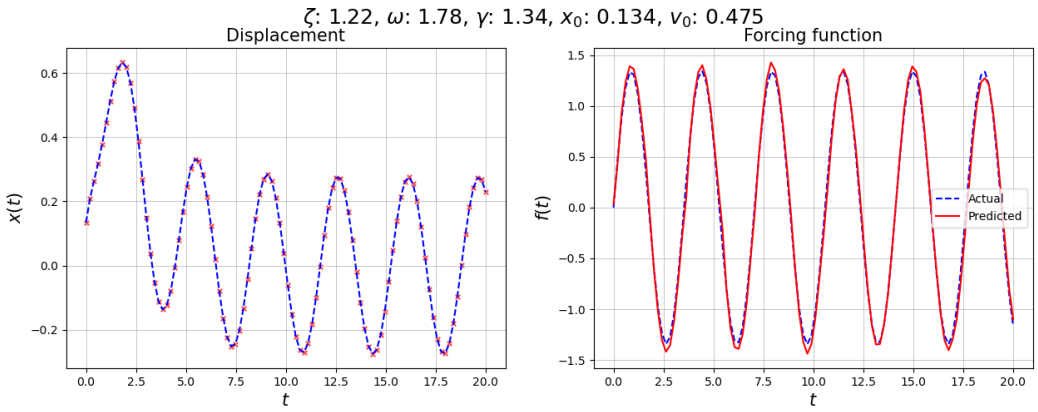
A 10% noise level presents a more challenging scenario. Here, the model's predictions, as observed in 3.8(a) and (c), show evident deviations, However, the some pattern remains discernible, suggesting that the model can still extract some meaningful information from the data despite significant noise interference.

In the figure 3.8(b) and (d), the presence of noise results in predictions that exhibit a higher degree of discrepancy when compared to their noiseless predictions. Specifically, for the case illustrated in figure 3.8(b), the model managed to accurately predict only a small segment of the forcing function. Additionally, in the case of figure 3.8(d), the model's predictive capability was confined to datasets corrupted with 1% noise. Predictions for datasets with higher noise levels, specifically 5% and 10%, were significantly impaired and hence are not depicted in the figure.

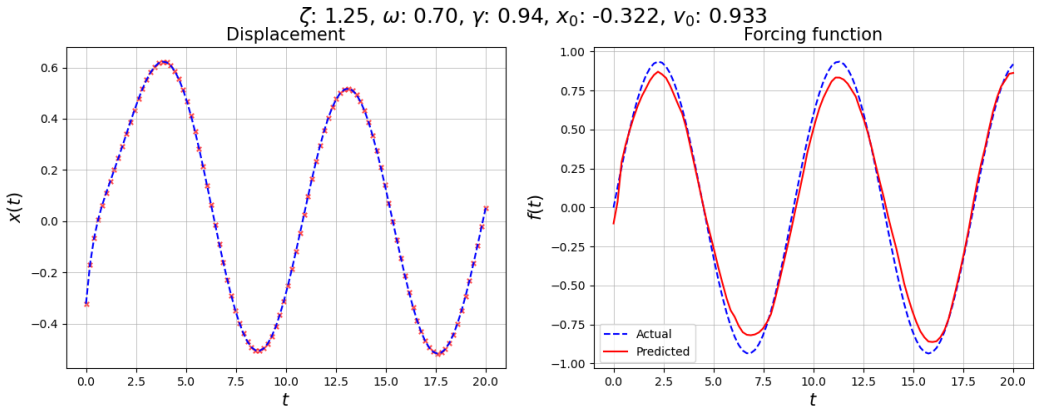
3.4.2 Prediction from PINNs surrogate model

3.4.2.1 Prediction on the test dataset

The model was trained following the process outlined in section 3.3.2.4, and evaluated on the test set compromising of 30 samples. However for sake of brevity, the results are shown for the cases involving high and low damping with increased and decreased frequency.



(a)

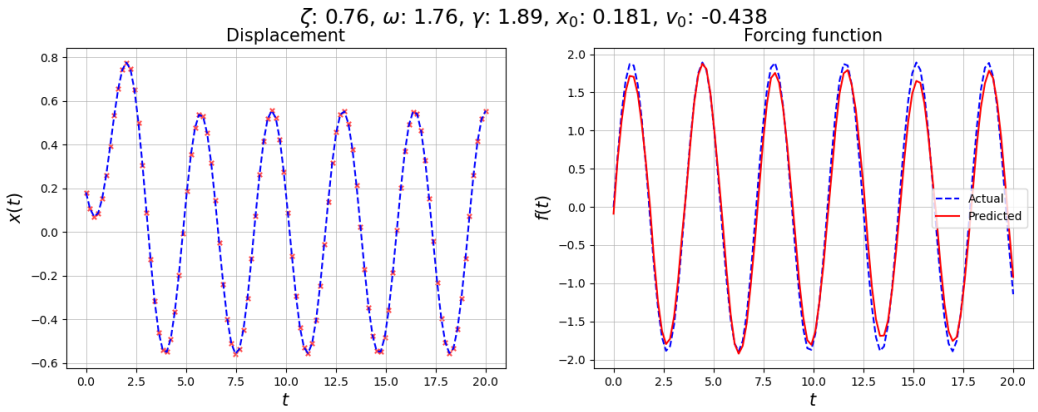


(b)

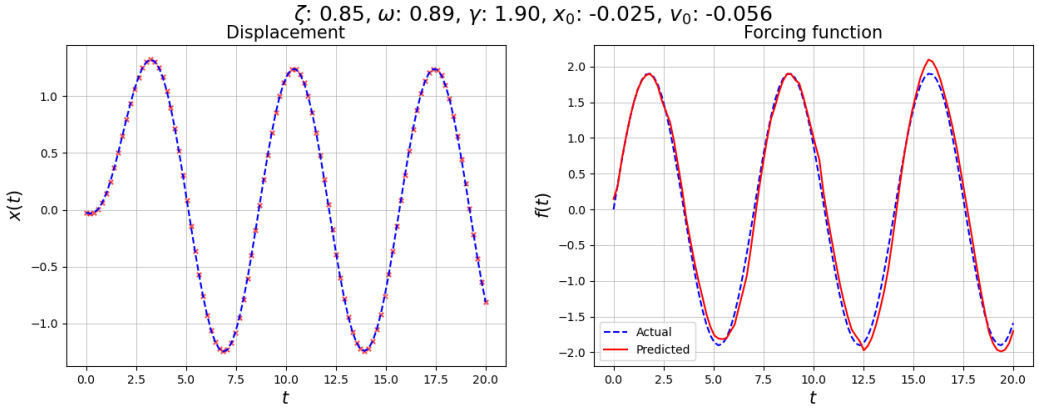
Figure 3.9: Samples from surrogate model's predictions on test dataset, case with high damping (a) high frequency (b) low frequency

As illustrated in figure 3.4, the input to the model was only displacement history $\mathbf{x}$ and velocity initial condition $\dot{x}_0$ with no physical parameters. The surrogate model demonstrated adeptness in predicting forcing functions across a variety of cases. In a particular case with high damping and high frequency, shown in 3.9 (a), where $\zeta = 1.22$, $\omega = 1.78$, $\gamma = 1.34$, $x_0 = 0.134$, and $\dot{x}_0 = 0.475$, the predicted forcing function closely followed the actual function with significant accuracy. Further, similar level of agreement with slight under-prediction was observed in 3.9 (b) for the case with a high damping and lower frequency $\zeta = 1.25$, $\omega = 0.70$, $\gamma = 0.94$, $x_0 = -0.322$, and $\dot{x}_0 = 0.933$.

Further, in cases, as shown in figure 3.10 with low damping and (a) high characterized



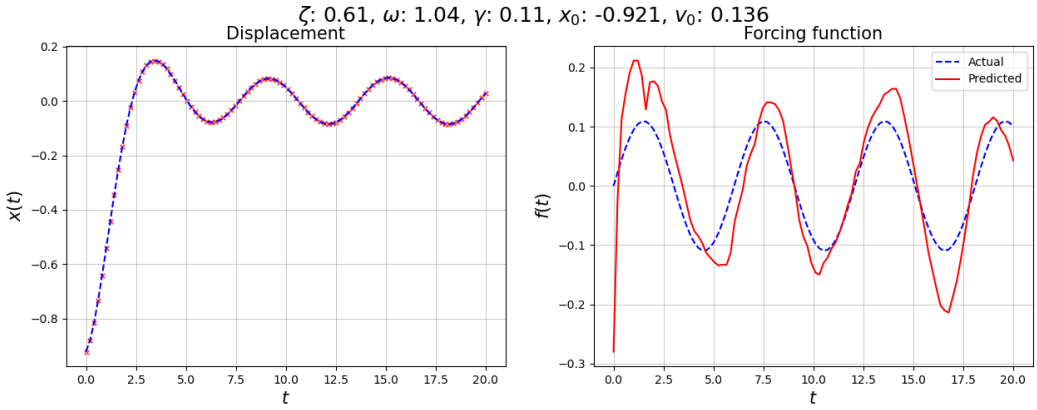
(a)



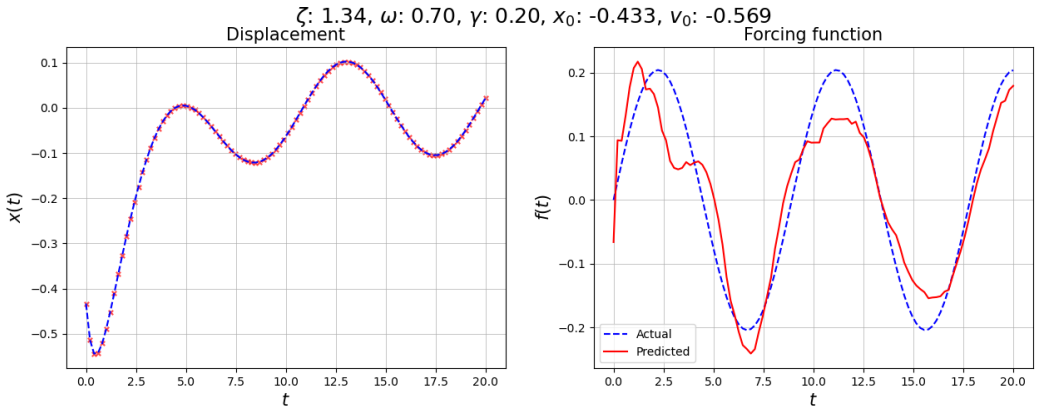
(b)

Figure 3.10: Samples from surrogate model's predictions on test dataset, case with low damping (a) high frequency (b) low frequency

by $\zeta = 0.76$, $\omega = 1.76$, $\gamma = 1.89$, $x_0 = 0.181$, and $\dot{x}_0 = -0.438$ and (b) low frequency with $\zeta = 0.85$, $\omega = 0.89$, $\gamma = 1.90$, $x_0 = -0.025$, and $\dot{x}_0 = -0.056$ respectively, the model's predictions mirrored the actual forcing function with negligible shift and under-predictions.



(a)



(b)

Figure 3.11: Samples demonstrating poor predictions in test dataset

Conversely, certain cases highlighted the model's limitations. As observed in figure 3.11, discrepancies between the predicted and actual forcing functions were evident in cases with (a) $\zeta = 0.61$, $\omega = 1.04$, $\gamma = 0.11$, $x_0 = -0.921$, $\dot{x}_0 = 0.136$ and (b) $\zeta = 1.34$, $\omega = 0.70$, $\gamma = 0.20$, $x_0 = -0.433$, and $\dot{x}_0 = -0.569$. In (a) model was unable to match the actual forcing function however it was able to capture the overall nature of the function. Similarly, in (b) the model faced some issues in predictions however it was able to predict overall characteristics.

3.4.2.2 Predictions against noisy data

Following the successful evaluation of surrogate model on the testing data in preceding section. The robustness of the surrogate model was further tested under conditions where the input displacement data is corrupted by noise. This is an essential consideration since the measurements from sensors in real world is often corrupted with noise. The model's predictions were assessed across various noise levels to determine the impact on its predictive accuracy.

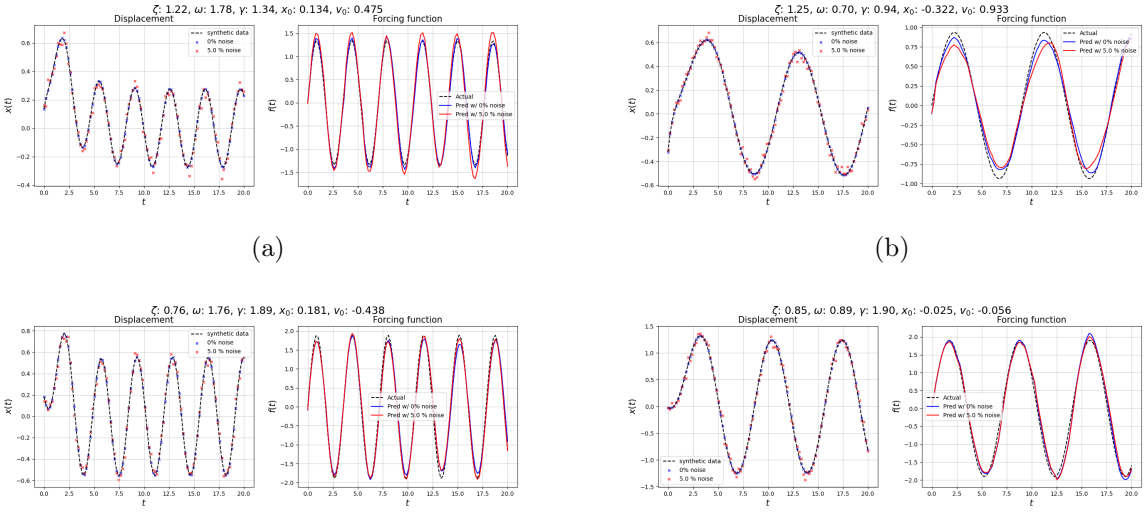


Figure 3.12: Predictions on test data with 5 % noise

Noise was artificially introduced to the test data following equation 3.29 with different noise levels such as 5%, 10%, and 15%. Further, the model was evaluated on these noisy dataset. To be consistent, cases form previous section evaluated on noisy inputs are shown in figure 3.12, 3.12 and 3.13. As observed in the figure the predictions of the forcing function by the surrogate model exhibited a certain degree of resilience to the presence of noise.

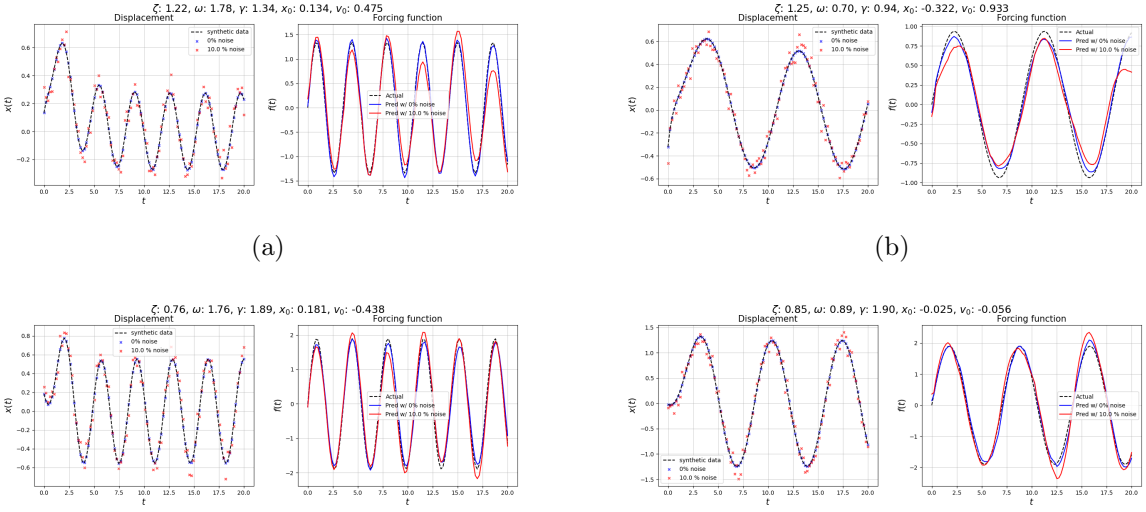


Figure 3.13: Predictions on test data with 10 % noise

At 5% and 10% noise level, as seen in Figure 3.12 and 3.13, the model's predictions were almost aligned with the actual data, with occasional shifts, over and under predictions observed at the peaks and valleys of the function. This demonstrated the model's ability to filter out minor inconsistencies and learn the underlying patterns of the system.

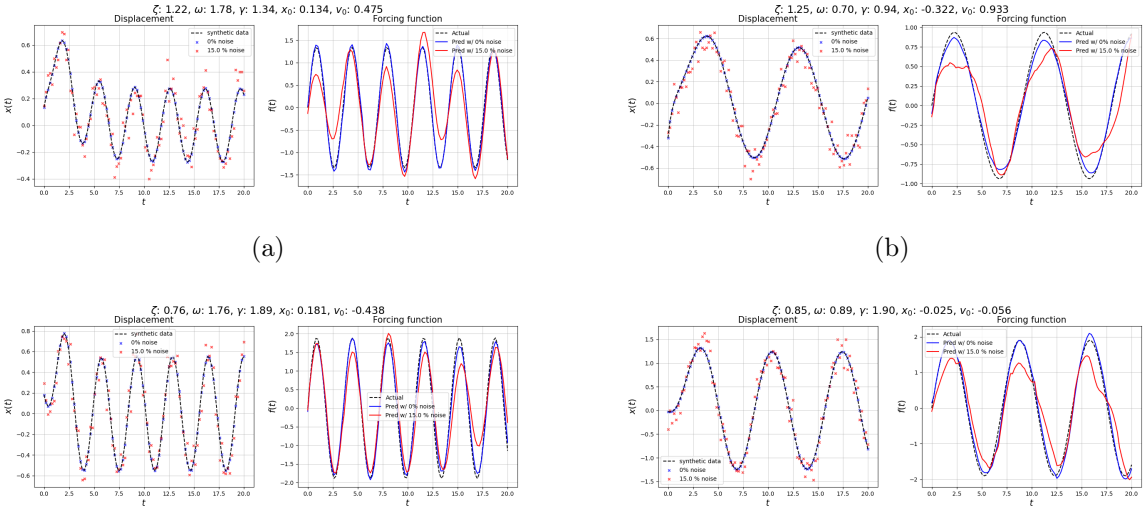


Figure 3.14: Predictions on test data with 15 % noise

Further, at 15% noise, as shown in figure 3.14, the deviations were found to be much more pronounced. The model's ability to predict the functions from the noisy data was challenged, leading to predictions that, while still capturing the general trend of the system's behavior, were less precise in matching the actual forcing functions.

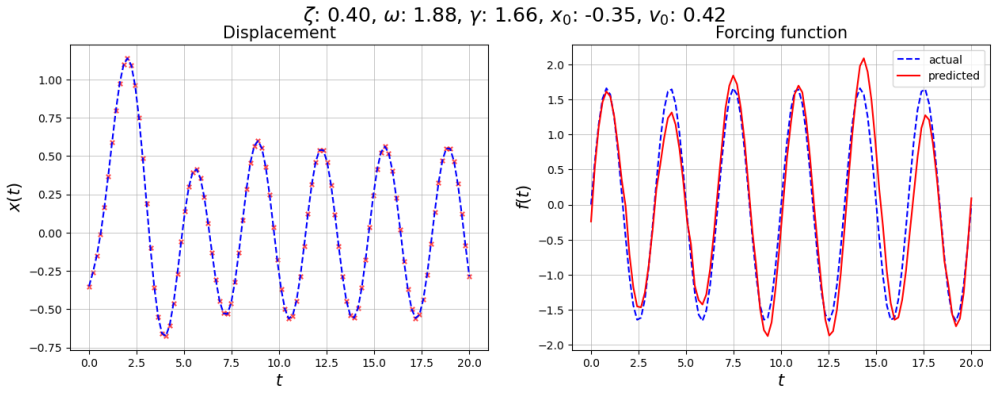
3.4.2.3 Prediction on the new cases

Following the evaluation of surrogate model against the noisy data in the preceding section. The surrogate model was further tested for generalization and its understanding of physics by inputting new cases that were not part of the training, testing or validation datasets. It is important to note that for each of the cases the input to the model was only displacement history, and velocity initial condition.

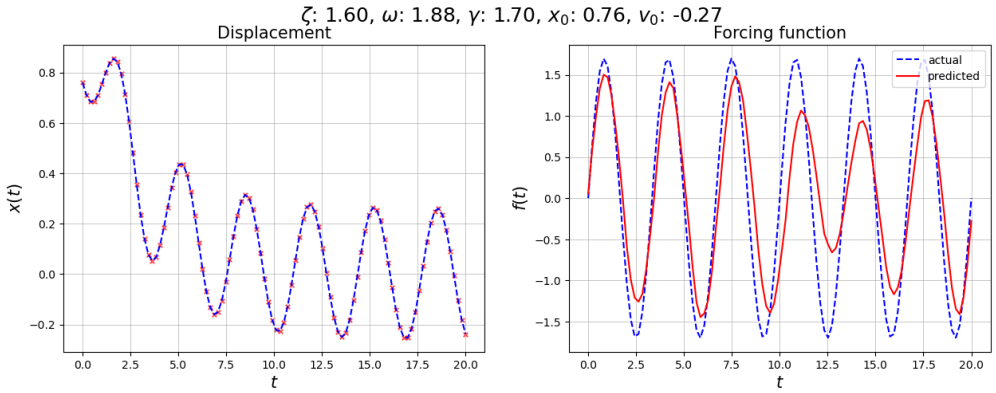
The data for evaluation was generated by simulating the with: a) $\zeta = 0.40$, $\omega = 1.88$, $\gamma = 1.66$, $x_0 = -0.35$, $\dot{x}_0 = 0.42$ characterizing the under-damped case, and b)) $\zeta = 1.60$, $\omega = 1.88$, $\gamma = 1.70$, $x_0 = 0.76$, $\dot{x}_0 = -0.27$ characterizing over-damped case.

These parameters, apart from ζ and ω, were chosen randomly from the range in table. The value of ω was fixed at highest frequency i.e. 0.6π and ζ was set to 0.4 and 1.6. The selected values of ζ were different from the range used for generating the dataset.

For a case characterized by low damping, shown in figure the model yielded predictions that closely tracked the actual forcing function data with occasional small over and under predictions at peaks and shifts. The accuracy of the model in predicting both the amplitude and the frequency underscores its understanding of physics.



(a)



(b)

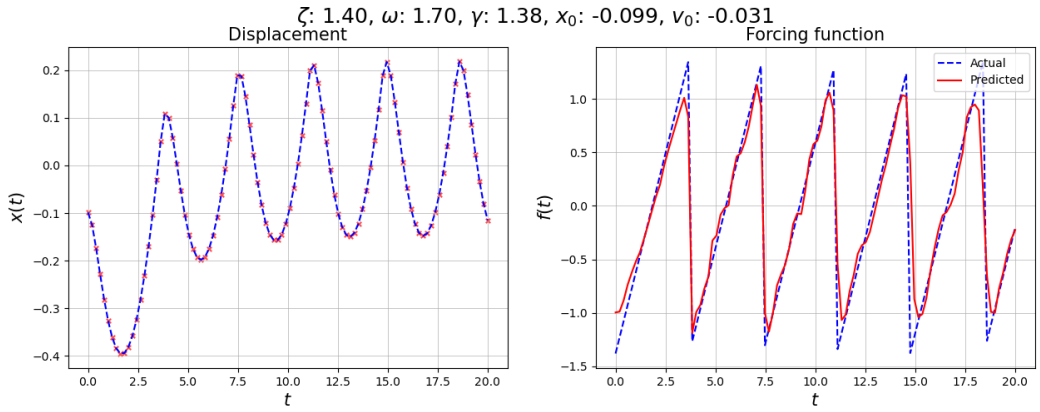
Figure 3.15: Figure demonstrating PINNs model generalization on new cases

For the other case, the surrogate model's predictions mirrored the general oscillatory trend. However, they exhibited more pronounced deviations compared to the previous case. It was able to capture the overall dynamics of the system. This demonstrates the models adeptness at handling new cases and degree of generalization

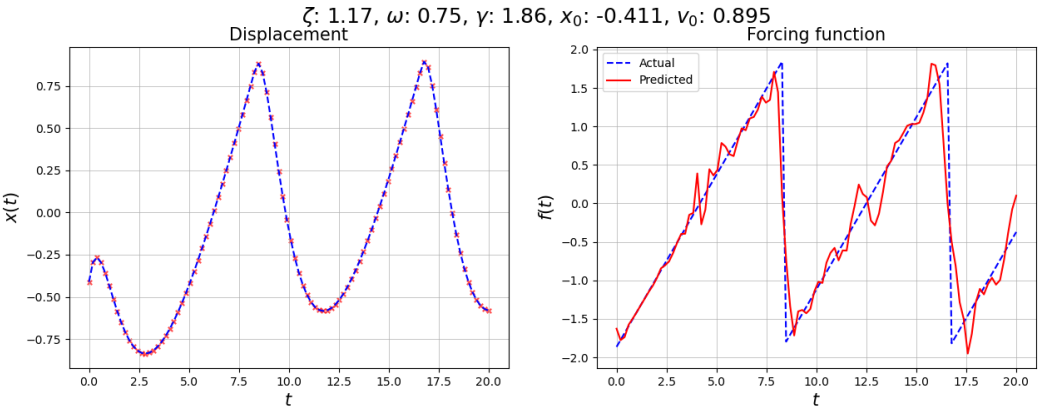
3.4.2.4 Predictions of $f(t)$ involving sharp gradients via transfer learning

Leveraging the success of the surrogate model in accurately predicting the sinusoidal function $f(t)$ as discussed in previous sections, this section explores its application to datasets characterized by sharply changing gradients. Specifically, a new dataset was generated following the methodology described in Section 3.3.2.1, utilizing a saw-tooth forcing function marked by abruptly changing gradients.

To enhance model convergence and capitalize on the knowledge acquired from prior training, the weights of the previously trained surrogate model were adopted as the initial starting point. This transfer learning approach allows the model to leverage previously learned features without necessitating modifications to the architecture. The model underwent further training adhering to the training process and hyper-parameter specifications detailed in Section 3.3.2.4.



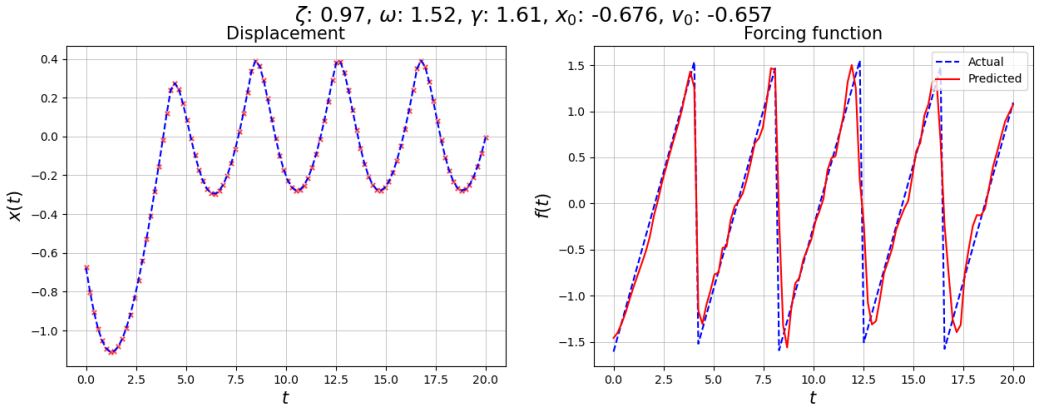
(a)



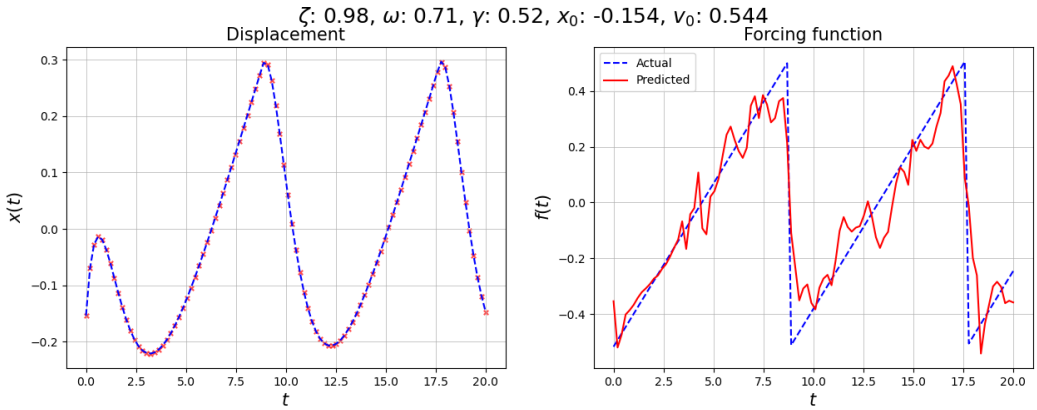
(b)

Figure 3.16: Samples from surrogate model's predictions on abruptly changing gradient test dataset, case with high damping (a) high frequency (b) low frequency

The efficacy of this approach is illustrated in Figures 3.16 and 3.17, showcasing the model's capability to accurately predict functions with sharp gradients by utilizing the knowledge gained from earlier sinusoidal function predictions.



(a)



(b)

Figure 3.17: Samples from surrogate model's predictions on abruptly changing gradient test dataset, case with low damping (a) high frequency (b) low frequency

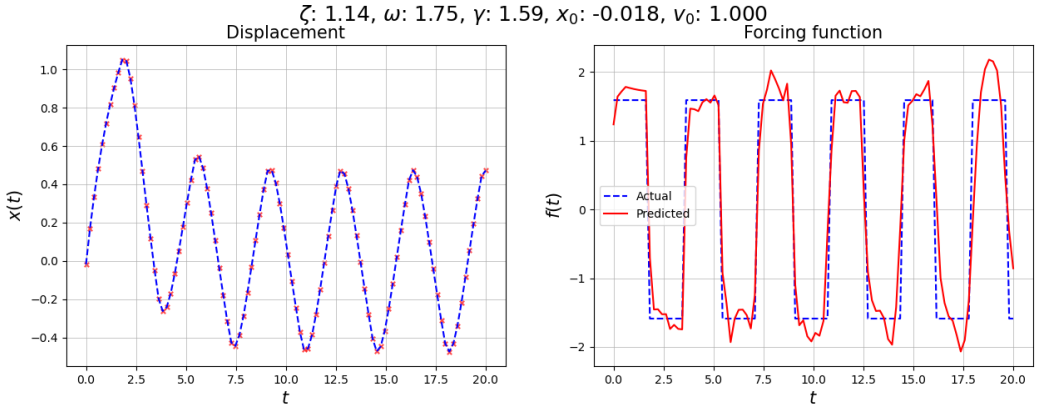
From the figures, it is evident that the predictions from the surrogate model closely follow the actual forcing function, illustrating a well-fitted model for most of the data points. However, there exists a small under-prediction at the cusps of the forcing functions in the case of Figure 3.17(a). Furthermore, for Figure 3.17(b), several oscillations were observed at the section of the function with increasing magnitude.

In Figure 3.17(c), the model prediction for the forcing function is consistent, however some deviations are observed, similar to previous case. Further, in Figure 3.17(d), the model seems capture the overall trend of the forcing function. However, the model does not precisely align with the actual function. A more pronounced oscillations compared to previous cases can

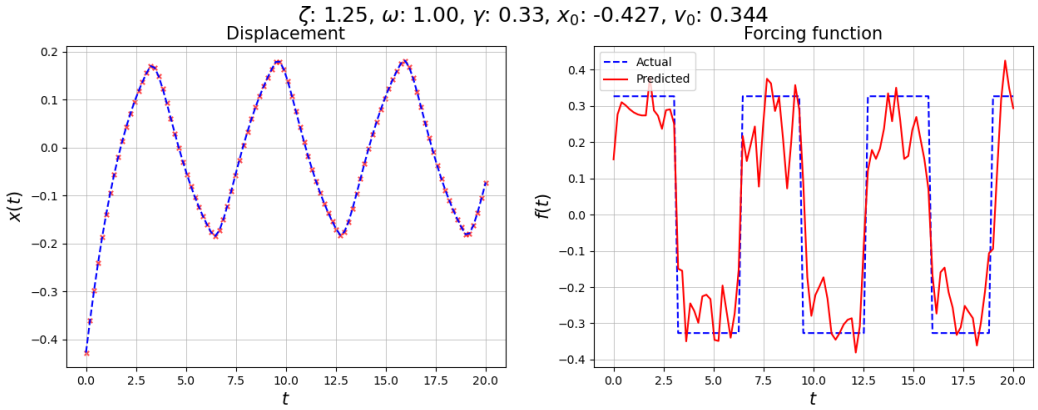
be observed.

3.4.2.5 Predictions of $f(t)$ involving jump discontinuities via transfer learning

Building on the surrogate model's success in handling functions with sharply changing gradients, as discussed in the previous sections, it application was extended to datasets featuring jump discontinuities. Following the methodology outlined in Section 3.3.2.1, a new dataset was generated incorporating a step function that exhibits sudden changes and discontinuous jumps.



(a)

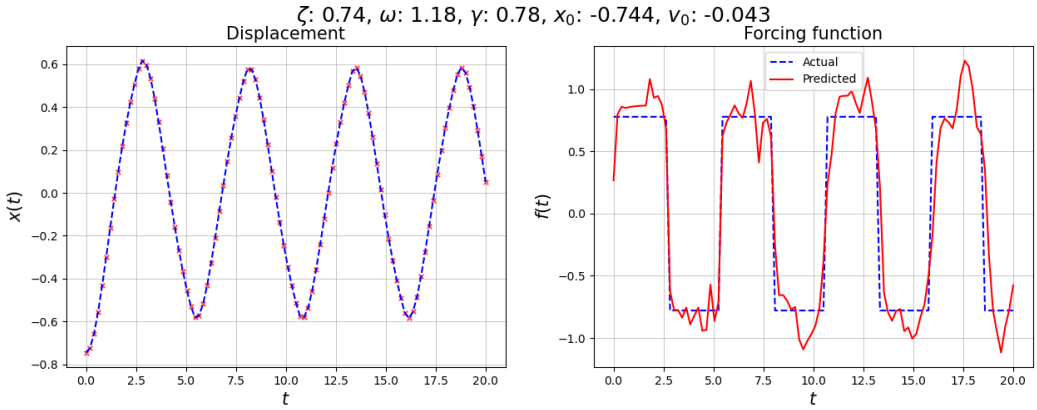


(b)

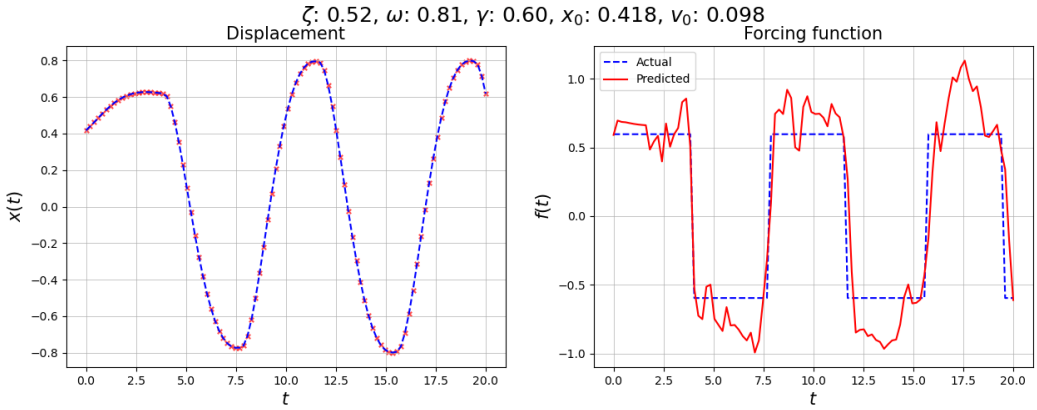
Figure 3.18: Samples from surrogate model's predictions on jump discontinuity test dataset, case with high damping (a) high frequency (b) low frequency

The weights from the surrogate model trained on the sharply changing gradient data served as the starting point for the this task, incorporating the transfer learning technique.

This strategy enables the model to apply its learned behavior from the previous tasks to the new dataset. Consistent with the procedures discussed in Section 3.3.2.4, the model was further trained to fine-tune its predictions for the current dataset's characteristics.



(a)



(b)

Figure 3.19: Samples from surrogate model's predictions on jump discontinuity gradient test dataset, case with high damping (a) high frequency (b) low frequency

The results presented in Figures 3.18 and 3.19 validate the surrogate model's capability to adapt to the abrupt changes represented by jump discontinuities. It can be seen in both the cases that that the model was able to predict the overall shape, i.e. frequency and amplitude, of the forcing functions. However, the predictions at the peaks were complemented by oscillations. A similar behaviour was observed when recovering step function from displacement and velocity data, as shown in Figure 3.5 (d).

3.5 Discussion

The focus of this study was to tackle a special type of ISP known as the dynamic load identification problem. Ultimately, leading up to the development of a surrogate model which is capable of predicting forcing functions from a given displacement data and initial conditions without the need for physical parameters.

The initial part of the work involved the development of a PINN model for recovering the $f(t)$ from displacement $x(t)$, velocity data $\dot{x}(t)$, and initial conditions, both without and in the presence of noise. The data instances were generated by simulating Duffing's equations for both linear and non-linear cases. For linear cases, forcing functions involving smooth, abruptly changing gradients, and jump discontinuities were considered. Similarly, for non-linear cases, only smooth functions were considered.

The PINN performance for all linear cases, in the absence of noise, was excellent, with slight overshoot and undershoot at the cusp of changing gradients and some numerical oscillations in the case of jump discontinuities resembling the Gibbs phenomenon. For the nonlinear cases, the model was still effective, with small amounts of initial instability and minor overshoot and undershoot at the peak and trough of the functions.

In the presence of noise for both linear and non-linear cases, the model demonstrated a degree of robustness in recovering the forcing function with an overall discernible shape. However, in a non-linear case with a high amount of non-linearity and presence of noise, significant deviations in the predictions were observed.

The second part of this work focused on the development of a surrogate model which can predict $f(t)$ only from displacement data $x(t)$ and initial conditions without utilizing physical parameters. A new PINN model with a different architecture was developed and trained on a synthetic dataset that was generated by simulating Duffing's equation without nonlinearity and with a sinusoidal forcing function. For generating data, different initial conditions, damping ratios, frequencies, and amplitudes were used.

Overall, the developed PINN model demonstrated an impressive performance on the hold-

out set. The model predictions were almost aligned with the actual forcing function in most cases. However, in a few cases, slight under and over predictions at the peaks and troughs were observed.

The model was further evaluated for generalization for cases different than those used for training. It was found that the model was able to predict the overall shape of the forcing function, with predictions aligned with the actual forcing function in most of the places. Following this, to emulate realistic measurements, the test dataset was corrupted with varying degrees of noise, and the PINN model's predictive performance was evaluated. It was found that the PINN model demonstrated a high degree of robustness against noise.

CHAPTER 4: CONCLUSIONS

This dissertation has successfully demonstrated the application of machine learning-based approaches to address both forward and inverse problems in engineering design with uncertainty analysis.

The research started by exploring the potential of lightweight materials like carbon fiber composites to replace traditional metallic alloys in electric vehicle battery enclosures, aiming for improvements in crash performance and overall vehicle efficiency.

To accelerate the design space exploration for EV battery enclosures, a novel machine learning-based surrogate model using Gaussian processes was developed. The model outperformed the models from the previous study and further demonstrated robustness against noisy inputs. This model significantly reduced the time and computational resources required for the simulation chain, consisting of virtual manufacturing and crash-testing simulations. The surrogate model's ability to quickly approximate results can assist in selecting optimal design parameters. This showcases the effectiveness of integrating machine learning with traditional engineering simulations. This integration not only streamlines the design process but also opens new avenues in engineering design.

Furthermore, the dissertation introduced a machine-learning method to solve the dynamic load identification problem — a specific type of inverse source problem. The method harnessed the power of PINNs to identify unknown source terms from noisy observational data for mechanical systems. Further, leading to the development of a physically consistent surrogate model that can predict forcing functions solely from noisy displacement data and initial conditions without need of physical parameters. This demonstrated PINNs capability to manage ill-posed problems and handle noisy data efficiently.

The methodologies and insights gained from this research lay a solid foundation for future

work. The machine learning-based approaches developed herein have the potential to be applied to a wider range of problems in engineering design and materials science. As the automotive industry continues to evolve towards electric mobility, the contributions of this dissertation will be helpful in future vehicle design, emphasizing safety, and performance.

CHAPTER 5: FUTURE WORKS

While this research work has shown promise, several key areas have been identified for future exploration:

1. This work investigated the performance of the enclosure in the event of a side pole impact test. However, other modes of failure may occur during a collision. Therefore, this study could be extended by incorporating other types of impacts to ensure a comprehensive assessment of the safety and performance of battery enclosures.

2. In this study, the fiber volume fraction was kept constant. However, varying the fiber volume may also impact the energy absorption capabilities of the composite structure. Future studies should incorporate this factor into their analyses.

3. Another direction for future research is to conduct experimental validations of the surrogate model's predictions to investigate their applicability and accuracy in practical engineering scenarios.

4. The geometry of the battery enclosure used in this study was simplified to make the analysis manageable. However, a realistic battery enclosure may include additional structural features and mounts that could impact the overall structural integrity of the battery. These should be considered in future work.

5. Future studies should explore models that account for varying enclosure geometries. This involves developing adaptable machine learning models that can efficiently process changes in design parameters, potentially leading to optimized structures.

6. PINNs can be further utilized to develop surrogate models for inferring the source term of nonlinear and multi-degree of freedom problems. This can be utilized in engineering

applications that require repeated identification of the source term.

By pursuing these directions, future research can build upon the findings of this dissertation.

REFERENCES

[1] S. A. Shaikh, M. Taufique, S. S. Kulkarni, F. Hale, J. Oleson, R. Devanathan, A. Soulami, *et al.*, "Finite element analysis and machine learning guided design of carbon fiber organosheet-based battery enclosures for crashworthiness," *arXiv preprint arXiv:2309.00637*, 2023.

[2] P. C. Hansen, *Discrete inverse problems: insight and algorithms*. SIAM, 2010.

[3] "https://electrichasgoneaudi.net/models/e-tron/drivetrain/battery/."

[4] "Automotive composites applications." https://www.huntsman-transportation.com/EN/applications/applications-for-composites.html, 2024. Accessed: 2024-03-13.

[5] B. Geiselman, "Plastics trim ev batteries' weight, boost safety," *Plastics Machinery Manufacturing*, February 2022. Accessed: 2024-03-13.

[6] E. group, "Pam-form user manual," 2019.

[7] Kecia, "Euro ncap: Leaving us in the dust (part i)." https://carseatblog.com/1537/euro-ncap-leaving-us-in-the-dust-part-i/, March 2009. Accessed: 2024-03-13.

[8] V. K, "Side pole car crash simulation of dodge neon biw using radioss and hypercrash." https://skill-lync.com/student-projects/assignment-7-87, February 2021. Accessed: 2024-03-13.

[9] S. S. Kulkarni, F. Hale, M. Taufique, A. Soulami, and R. Devanathan, "Investigation of crashworthiness of carbon fiber-based electric vehicle battery enclosure using finite element analysis," *Applied Composite Materials*, pp. 1–27, 2023.

[10] S. A. Shaikh, H. Cherukuri, and T. Khan, "Recovering the forcing function in systems with one degree of freedom using ann and physics information," *Algorithms*, vol. 16, no. 5, p. 250, 2023.

[11] S. I. Kabanikhin, "Definitions and examples of inverse and ill-posed problems," 2008.

[12] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models," *Acta Numerica*, vol. 28, p. 1–174, 2019.

[13] 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, pp. 422–440, 2021.

[14] A. Eftekhari, "Lithium batteries for electric vehicles: from economy to research strategy," 2019.

[15] D. Berjoza, I. Jurgena, *et al.*, "Effects of change in the weight of electric vehicles on their performance characteristics," *Agron. Res*, vol. 15, no. 1, pp. 952–963, 2017.

[16] X. Zhang, Y. Chen, and J. Hu, "Recent advances in the development of aerospace materials," *Progress in Aerospace Sciences*, vol. 97, pp. 22–34, 2018.

[17] T. K. Das, P. Ghosh, and N. C. Das, "Preparation, development, outcomes, and application versatility of carbon fiber-based polymer composites: a review," *Advanced Composites and Hybrid Materials*, vol. 2, pp. 214–233, 2019.

[18] I. Swentek, C. A. Ball, S. Greydanus, and K. R. Nara, "Phenolic smc for fire resistant electric vehicle battery box applications," tech. rep., SAE Technical Paper, 2020.

[19] A. Sayam, A. M. Rahman, M. S. Rahman, S. A. Smriti, F. Ahmed, M. F. Rabbi, M. Hossain, and M. O. Faruque, "A review on carbon fiber-reinforced hierarchical composites: Mechanical performance, manufacturing process, structural applications and allied challenges," *Carbon Letters*, vol. 32, no. 5, pp. 1173–1205, 2022.

[20] A. M. Almushaikeh, S. O. Alaswad, M. S. Alsuhybani, B. M. AlOtaibi, I. M. Alarifi, N. B. Alqahtani, S. M. Aldosari, S. S. Alsaleh, A. S. Haidyrah, A. A. Alolyan, *et al.*, "Manufacturing of carbon fiber reinforced thermoplastics and its recovery of carbon fiber: A review," *Polymer Testing*, p. 108029, 2023.

[21] "Pam-form." https://www.esi.com.au/software/pamform/ (accessed apr. 24, 2023).."

[22] "Virtual performance solution (vps) | crash simulation software." https://www.esi-group.com/products/virtual-performance-solution (accessed apr. 24, 2023).."

[23] P. M. Bean, *Modeling and Simulation of the Thermoforming Process in Thermoplastic-Matrix Composite Materials*. The University of Maine, 2018.

[24] E. group, "Vps solver reference manual," 2020.

[25] "Side impact protection | nhtsa." accessed Apr. 24, 2023).

[26] "Euro ncap | the european new car assessment programme." accessed Apr. 24, 2023).

[27] P. Ladeveze and E. L.-C, "science and technology, and undefined 1992, "damage modelling of the elementary ply for laminated composites," *Elsevier, Accessed: Apr*, vol. 24. Online]. Available:.

[28] A. F. Johnson, A. K. Pickett, and P. Rozycki, "Computational methods for predicting impact damage in composite structures," *Composites Science and Technology*, vol. 61, no. 15, pp. 2183–2192, 2001.

[29] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[30] J. Wang, "An intuitive tutorial to gaussian processes regression," *Computing in Science & Engineering*, 2023.

[31] C. B. Do and H. Lee, "Gaussian processes," *Stanford University, Stanford, CA, accessed Dec*, vol. 5, p. 2017, 2007.

[32] C. Williams and C. Rasmussen, "Gaussian processes for regression," *Advances in neural information processing systems*, vol. 8, 1995.

[33] V. Nemani, L. Biggio, X. Huan, Z. Hu, O. Fink, A. Tran, Y. Wang, X. Zhang, and C. Hu, "Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial," *Mechanical Systems and Signal Processing*, vol. 205, p. 110796, 2023.

[34] J. Zhang, "Modern monte carlo methods for efficient uncertainty quantification and propagation: A survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 13, no. 5, p. e1539, 2021.

[35] B. Sudret, S. Marelli, and J. Wiart, "Surrogate models for uncertainty quantification: An overview," in *2017 11th European conference on antennas and propagation (EUCAP)*, pp. 793–797, IEEE, 2017.

[36] C. Iclodean, B. Varga, N. Burnete, D. Cimerdean, and B. Jurchiş, "Comparison of different battery types for electric vehicles," *IOP Conf Ser Mater Sci Eng*, vol. 252, pp. 012058,.

[37] J. Zhu, T. Wierzbicki, and W. Li, "A review of safety-focused mechanical modeling of commercial lithium-ion batteries," *J Power Sources*, vol. 378, pp. 153–168,.

[38] X. Zhang, Y. Chen, and J. Hu, "Recent advances in the development of aerospace materials," *Progress in Aerospace Sciences*, vol. 97, pp. 22–34,.

[39] G. V. Research, "Automotive polymer composites market size report by resin (epoxy, polyurethane, polyamide, polypropylene, polyethylene), by application, by product, by end use, by manufacturing, and segment forecasts, 2018–2025."

[40] "Solvay." accessed Feb 2024).

[41] J. Zhu, X. Zhang, T. Wierzbicki, Y. Xia, and G. Chen, "Structural designs for electric vehicle battery pack against ground impact," *Apr.*

[42] A. Navale, S. Chippa, D. Chougule, and P. Raut, "Crashworthiness aspects of electric vehicle design," *International journal of crashworthiness*, vol. 26, no. 4, pp. 368–387, 2021.

[43] P. Victor Chombo, Y. Laoonual, and S. Wongwises, "Lessons from the electric vehicle crashworthiness leading to battery fire," *Energies*, vol. 14, no. 16, p. 4802, 2021.

[44] L. Shui, F. Chen, A. Garg, X. Peng, N. Bao, and J. Zhang, "Design optimization of battery pack enclosure for electric vehicle," *Structural and Multidisciplinary Optimization*, vol. 58, pp. 331–347, 2018.

[45] Y. Pan, Y. Xiong, L. Wu, K. Diao, and W. Guo, "Lightweight design of an automotive battery-pack enclosure via advanced high-strength steels and size optimization," *International Journal of Automotive Technology*, vol. 22, no. 5, pp. 1279–1290,.

[46] Y. Kim, Y. Kim, C. Yang, K. Park, G. X. Gu, and S. Ryu, "Deep learning framework for material design space exploration using active transfer learning and data augmentation," *npj Computational Materials 2021 7:1*, vol. 7, pp. 1–7, 9 2021.

[47] A. Coppola, "Validation of material models for automotive carbon fiber composite structures via physical and crash testing (vmm composites project." United States), Sep.

[48] W. Joost, "Reducing vehicle weight and improving u.s. energy efficiency using integrated computational materials engineering," *JOM*, vol. 64, no. 9, pp. 1032–1038,.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[50] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, vol. 12, 2016.

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[52] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "TabDDPM: Modelling tabular data with diffusion models," in *Proceedings of the 40th International Conference on Machine Learning* (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds.), vol. 202 of *Proceedings of Machine Learning Research*, pp. 17564–17579, PMLR, 23–29 Jul 2023.

[53] R. Richman and M. V. Wüthrich, "Localglmnet: interpretable deep learning for tabular data," *Scandinavian Actuarial Journal*, vol. 2023, no. 1, pp. 71–95, 2023.

[54] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 6679–6687, 2021.

[55] L. Katzir, G. Elidan, and R. El-Yaniv, "Net-dnf: Effective deep modeling of tabular data," in *International conference on learning representations*, 2020.

[56] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[57] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," *Advances in neural information processing systems*, vol. 35, pp. 507–520, 2022.

[58] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[59] J. Gubernatis and T. Lookman, "Machine learning in materials design and discovery: Examples from the present and suggestions for the future," *Physical Review Materials*, vol. 2, no. 12, p. 120301, 2018.

[60] J. Wei, X. Chu, X.-Y. Sun, K. Xu, H.-X. Deng, J. Chen, Z. Wei, and M. Lei, "Machine learning in materials science," *InfoMat*, vol. 1, no. 3, pp. 338–358, 2019.

[61] D. Morgan and R. Jacobs, "Opportunities and challenges for machine learning in materials science," *Annual Review of Materials Research*, vol. 50, pp. 71–103, 2020.

[62] T. Zhou, Z. Song, and K. Sundmacher, "Big data creates new opportunities for materials research: a review on methods and applications of machine learning for materials design," *Engineering*, vol. 5, no. 6, pp. 1017–1026, 2019.

[63] J. G. Hoffer, B. C. Geiger, and R. Kern, "Gaussian process surrogates for modeling uncertainties in a use case of forging superalloys," *applied sciences*, vol. 12, no. 3, p. 1089, 2022.

[64] G. Tapia, S. Khairallah, M. Matthews, W. E. King, and A. Elwany, "Gaussian process-based surrogate modeling framework for process planning in laser powder-bed fusion additive manufacturing of 316l stainless steel," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 3591–3603, 2018.

[65] J. Zhou and L.-S. Turng, "Process optimization of injection molding using an adaptive surrogate model with gaussian process approach," *Polymer Engineering & Science*, vol. 47, no. 5, pp. 684–694, 2007.

[66] M. I. Radaideh and T. Kozlowski, "Surrogate modeling of advanced computer simulations using deep gaussian processes," *Reliability Engineering & System Safety*, vol. 195, p. 106731, 2020.

[67] R. Saunders, C. Butler, J. Michopoulos, D. Lagoudas, A. Elwany, and A. Bagchi, "Mechanical behavior predictions of additively manufactured microstructures using functional gaussian process surrogates," *npj Computational Materials*, vol. 7, no. 1, p. 81, 2021.

[68] M. M. Noack, G. S. Doerk, R. Li, J. K. Streit, R. A. Vaia, K. G. Yager, and M. Fukuto, "Autonomous materials discovery driven by gaussian process regression with inhomogeneous measurement noise and anisotropic kernels," *Scientific reports*, vol. 10, no. 1, p. 17663, 2020.

[69] B. Chen, L. Shen, and H. Zhang, "Gaussian process regression-based material model for stochastic structural analysis," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, vol. 7, no. 3, p. 04021025, 2021.

[70] X. Liu, S. Tian, F. Tao, H. Du, and W. Yu, "How machine learning can help the design and analysis of composite materials and structures?," *arXiv preprint arXiv:2010.09438*, 2020.

[71] X. Liu, S. Tian, F. Tao, H. Du, and W. Yu, "Machine learning-assisted modeling of composite materials and structures: a review," in *AIAA Scitech 2021 Forum*, p. 2023, 2021.

[72] C.-T. Chen and G. X. Gu, "Machine learning for composite materials," *MRs Communications*, vol. 9, no. 2, pp. 556–566, 2019.

[73] A. Sharma, T. Mukhopadhyay, S. M. Rangappa, S. Siengchin, and V. Kushvaha, "Advances in computational intelligence of polymer composite materials: machine learning assisted modeling, analysis and design," *Archives of Computational Methods in Engineering*, vol. 29, no. 5, pp. 3341–3385, 2022.

[74] C. Zhang, Y. Li, B. Jiang, R. Wang, Y. Liu, and L. Jia, "Mechanical properties prediction of composite laminate with fea and machine learning coupled method," *Composite Structures*, vol. 299, p. 116086, 2022.

[75] H. E. Balcıoğlu and A. Ç. Seçkin, "Comparison of machine learning methods and finite element analysis on the fracture behavior of polymer composites," *Archive of Applied Mechanics*, vol. 91, pp. 223–239, 2021.

[76] A. Milad, S. H. Hussein, A. R. Khekan, M. Rashid, H. Al-Msari, and T. H. Tran, "Development of ensemble machine learning approaches for designing fiber-reinforced polymer composite strain prediction model," *Engineering with Computers*, vol. 38, no. 4, pp. 3625–3637, 2022.

[77] R. L. Iman, "L atin hypercube sampling," *Wiley StatsRef: Statistics Reference Online*, 2014.

[78] R. B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. CRC press, 2020.

[79] M. Baudin, M. Christopoulou, Y. Colette, and J. Martinez, "pydoe: The experimental design package for python," *URL https://pythonhosted. org/pyDOE/index. html*, 2013.

[80] X. Cui, B. Panda, C. M. M. Chin, N. Sakundarini, C.-T. Wang, and K. Pareek, "An application of evolutionary computation algorithm in multidisciplinary design optimization of battery packs for electric vehicle," *Energy Storage*, vol. 2, no. 3, p. e158, 2020.

[81] W. Lu, C. Xiao-kai, and Z. Qing-hai, "Muti-objective topology optimization of an electric vehicle's traction battery enclosure," *Energy Procedia*, vol. 88, pp. 874–880, 2016.

[82] Y. Ren, H. Jiang, B. Gao, and J. Xiang, "A progressive intraply material deterioration and delamination based failure model for the crashworthiness of fabric composite corrugated beam: Parameter sensitivity analysis," *Composites Part B: Engineering*, vol. 135, pp. 49–71, 2018.

[83] S. K. Singh, R. Pandey, and A. Upadhyay, "A numerical study on combined effects of groove shape and numbers on crashworthiness characteristics of thin-walled tube," *Materials Today: Proceedings*, vol. 44, pp. 4381–4386, 1 2021.

[84] D. H.-J. Lukaszewicz, "Automotive composite structures for crashworthiness," *Advanced composite materials for automotive applications: structural integrity and crashworthiness*, pp. 99–127, 2013.

[85] J. Xu, Y. Ma, Q. Zhang, T. Sugahara, Y. Yang, and H. Hamada, "Crashworthiness of carbon fiber hybrid composite tubes molded by filament winding," *Composite Structures*, vol. 139, pp. 130–140, 4 2016.

[86] J. S. Kim, H. J. Yoon, and K. B. Shin, "A study on crushing behaviors of composite circular tubes with different reinforcing fibers," *International Journal of Impact Engineering*, vol. 38, pp. 198–207, 4 2011.

[87] Q. Liu, H. Xing, Y. Ju, Z. Ou, and Q. Li, "Quasi-static axial crushing and transverse bending of double hat shaped cfrp tubes," *Composite Structures*, vol. 117, pp. 1–11, 11 2014.

[88] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[89] S. M. L. Group, "Gpy: A gaussian processes framework in python." https://sheffieldml.github.io/GPy/, 2024. Accessed: 2024-03-01.

[90] K. Liu, Y. Li, X. Hu, M. Lucu, and W. D. Widanage, "Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3767–3777, 2019.

[91] Y. Wang, J. Feng, J. Wu, and D. Hu, "Effects of fiber orientation and wall thickness on energy absorption characteristics of carbon-reinforced composite tubes under different loading conditions," *Composite Structures*, vol. 153, pp. 356–368, 2016.

[92] B. Xu, R. Kuplicki, S. Sen, and M. P. Paulus, "The pitfalls of using gaussian process regression for normative modeling," *Plos one*, vol. 16, no. 9, p. e0252108, 2021.

[93] H. Bijl, "Gaussian process regression techniques with applications to wind turbines," *Delft University of Technology, Doctoral degree*, 2016.

[94] A. McHutchon and C. Rasmussen, "Gaussian process training with input noise," *Advances in neural information processing systems*, vol. 24, 2011.

[95] A. Girard, *Approximate methods for propagation of uncertainty with Gaussian process models*. University of Glasgow (United Kingdom), 2004.

[96] J. E. Johnson, "Literature notes on uncertain gaussian processes." https://jejjohnson.github.io/uncertain$_g$ps/Notes/literature/, 2023. *Accessed* : $2024 - 03 - 14$.

[97] K. Rahul, D. A. Jack, and D. E. Smith, "A statistical approach for failure analysis involving uncertainty in determining ply orientation," *Polymer Composites*, 2024.

[98] P. Han, J. Butterfield, M. Price, S. Buchanan, and A. Murphy, "Experimental investigation of thermoforming carbon fibre-reinforced polyphenylene sulphide composites," *Journal of Thermoplastic Composite Materials*, vol. 28, no. 4, pp. 529–547, 2015.

[99] Calex Electronics Limited, *PyroMini Manual*, March 2018. Accessed: 2024-03-04.

[100] R. I. Corporation, "Thermocouple accuracies." https://www.thermocoupleinfo.com/thermocouple-accuracies.htm, 2023. Accessed: 2024-03-04.

[101] P. C. Sabatier, "Past and future of inverse problems," *Journal of Mathematical Physics*, vol. 41, no. 6, pp. 4082–4124, 2000.

[102] F. Yaman, V. G. Yakhno, and R. Potthast, "A survey on inverse problems for applied sciences," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–19, 2013.

[103] G. Uhlmann and S. G. F. Uhlmann, "Inverse problems: seeing the unseen," *Bulletin of Mathematical Sciences 2014 4:2*, vol. 4, pp. 209–279, 6 2014.

[104] S. R. Arridge, "Optical tomography in medical imaging," *Inverse problems*, vol. 15, no. 2, p. R41, 1999.

[105] P. Stefanov and G. Uhlmann, "Volume 1 no. 1 2008 an inverse source problem in optical molecular imaging an inverse source problem in optical molecular imaging," *ANALYSIS AND PDE*, vol. 1, 2008.

[106] N. J. McCormick, "Inverse radiative transfer problems: A review," *http://dx.doi.org/10.13182/NSE112-185*, vol. 112, pp. 185–198, 2017.

[107] H. Ertürk, K. Daun, F. H. França, S. Hajimirza, and J. R. Howell, "Inverse methods in thermal radiation analysis and experiment," *ASME Journal of Heat and Mass Transfer*, vol. 145, no. 5, p. 050801, 2023.

[108] H. Ammari, G. Bao, and J. L. Fleming, "An inverse source problem for maxwell's equations in magnetoencephalography," *https://doi.org/10.1137/S0036139900373927*, vol. 62, pp. 1369–1382, 7 2006.

[109] R. Grech, T. Cassar, J. Muscat, K. P. Camilleri, S. G. Fabri, M. Zervakis, P. Xanthopoulos, V. Sakkalis, and B. Vanrumste, "Review on solving the inverse problem in eeg source analysis," *Journal of NeuroEngineering and Rehabilitation*, vol. 5, pp. 1–33, 11 2008.

[110] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: where we are and what's next," *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.

[111] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[112] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (pinns) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021.

[113] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 379, p. 113741, 2021.

[114] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer*, vol. 143, no. 6, p. 060801, 2021.

[115] S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, p. 110768, 2022.

[116] I. Kovacic and M. J. Brennan, *The Duffing equation: nonlinear oscillators and their behaviour*. John Wiley & Sons, 2011.

[117] D. Antonio, D. H. Zanette, and D. López, "Frequency stabilization in nonlinear micromechanical oscillators," *Nature communications*, vol. 3, no. 1, p. 806, 2012.

[118] F. Lim, M. Cartmell, A. Cardoni, and M. Lucas, "A preliminary investigation into optimising the response of vibrating systems used for ultrasonic cutting," *Journal of Sound and Vibration*, vol. 272, no. 3-5, pp. 1047–1069, 2004.

[119] U. von Wagner and P. Hagedorn, "Nonlinear effects of piezoceramics excited by weak electric fields," *Nonlinear Dynamics*, vol. 31, pp. 133–149, 2003.

[120] Q. Cao, Y. Xiong, and M. Wiercigroch, "A novel model of dipteran flight mechanism," *International Journal of dynamics and control*, vol. 1, pp. 1–11, 2013.

[121] C. H. Huang, "A generalized inverse force vibration problem for simultaneously estimating the time-dependent external forces," *Applied Mathematical Modelling*, vol. 29, pp. 1022–1039, 11 2005.

[122] C. K. Ma, P. C. Tuan, D. C. Lin, and C. S. Liu, "A study of an inverse method for the estimation of impulsive loads," *http://dx.doi.org/10.1080/00207729808929559*, vol. 29, pp. 663–672, 1998.

[123] S. E. Azam, E. Chatzi, and C. Papadimitriou, "A dual kalman filter approach for state estimation via output-only acceleration measurements," *Mechanical systems and signal processing*, vol. 60, pp. 866–886, 2015.

[124] T. S. Jang, H. Baek, H. S. Choi, and S. G. Lee, "A new method for measuring nonharmonic periodic excitation forces in nonlinear damped systems," *Mechanical Systems and Signal Processing*, vol. 25, 2011.

[125] M. Feldman, "Mapping nonlinear forces with congruent vibration functions," *Mechanical Systems and Signal Processing*, vol. 37, pp. 315–337, 5 2013.

[126] M. Chao, H. Hongxing, and X. Feng, "The identification of external forces for a nonlinear vibration system in frequency domain," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 228, pp. 1531–1539, 2014.

[127] C. S. Liu and C. W. Chang, "A real-time lie-group differential algebraic equations method to solve the inverse nonlinear vibration problems," *Inverse Problems in Science and Engineering*, vol. 24, 2016.

[128] C. Rice and J. I. Frankel, "Estimating the forcing function in a mechanical system by an inverse calibration method," *JVC/Journal of Vibration and Control*, vol. 28, 2022.

[129] R. Liu, E. Dobriban, Z. Hou, and K. Qian, "Dynamic load identification for mechanical systems: A review," *Archives of Computational Methods in Engineering*, vol. 29, 2022.

[130] J. Prawin and A. R. M. Rao, "An online input force time history reconstruction algorithm using dynamic principal component analysis," *Mechanical Systems and Signal Processing*, vol. 99, pp. 516–533, 1 2018.

[131] J. M. Zhou, L. Dong, W. Guan, and J. Yan, "Impact load identification of nonlinear structures using deep recurrent neural network," *Mechanical Systems and Signal Processing*, vol. 133, 11 2019.

[132] H. Yang, J. Jiang, G. Chen, M. S. Mohamed, and F. Lu, "A recurrent neural network-based method for dynamic load identification of beam structures," *Materials 2021, Vol. 14, Page 7846*, vol. 14, p. 7846, 12 2021.

[133] L. Rosafalco, A. Manzoni, S. Mariani, and A. Corigliano, "An autoencoder-based deep learning approach for load identification in structural dynamics," *Sensors*, vol. 21, no. 12, p. 4207, 2021.

[134] Y. Liu, L. Wang, K. Gu, and M. Li, "Artificial neural network (ann) - bayesian probability framework (bpf) based method of dynamic force reconstruction under multi-source uncertainties," *Knowledge-Based Systems*, vol. 237, 2022.

[135] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics 2021 3:6*, vol. 3, pp. 422–440, 5 2021.

[136] Z. He, F. Ni, W. Wang, and J. Zhang, "A physics-informed deep learning method for solving direct and inverse heat conduction problems of materials," *Materials Today Communications*, vol. 28, p. 102719, 2021.

[137] Q. Liu, Z. Zhao, Y. Zhang, J. Wang, and J. Cao, "Physics-informed sparse identification of bistable structures," *Journal of Physics D: Applied Physics*, vol. 56, p. 044005, 12 2022.

[138] E. Haghighat, A. C. Bekar, E. Madenci, and R. Juanes, "Deep learning for solution and inversion of structural mechanics and vibrations," *Modeling and Computation in Vibration Problems, Volume 2*, vol. 1, 12 2021.

[139] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. D. et. al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[140] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[141] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization,"

[142] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[143] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics*, vol. 404, p. 109136, 2020.

APPENDIX A: Material properties used in thermoforming and crash simulations

Table A.1: Material properties for the die and punch

Property	Value
Mechanical properties	Rigid material
Convection coefficient	10 W/m^2 K
Conductivity	0.45 W/m.K

Table A.2: Material properties of composite sheets

Property	Value
Density	1×10^{-6} kg/mm^3
Initial angle between fibers	90 degrees
Thickness	0.25 mm
Fiber content	0.5
Tension compression stiffness (Fiber 1)	20 GPa
Tension compression stiffness (Fiber 2)	20 GPa
Bending stiffness (Fiber 1)	0.03 GPa
Bending stiffness (Fiber 2)	0.03 GPa
Conductivity	2.3×10^{-6} kW/mm °C
Specific heat	1150 J/kg °C
In plane shear	2.5×10^{-5} GPa
Sheet orientation	(90, 45-, 45, 0)
Layer separation stress	0.005 GPa
Convection coefficient	10 W/m^{2}K

Table A.3: Mechanical properties for the lid and rib

Property	Value
Density	1.8×10^{-6} kg/mm^3
Young's modulus	125 GPa
Yield stress	3.5 GPa
Poisson's ratio	0.33
Max plastic strain for element removal	0.014
Plastic tangent modulus	8 GPa

Table A.4: Material properties of enclosure used for crash simulations

Property	Value
Density	1.8×10^{-6} kg/mm^3
Young's modulus parallel to fiber	125 GPa
Young's modulus perpendicular to fiber	8 GPa
Critical shear damage limit	0.114 GPa
Initial shear damage limit	0.02 GPa
Initial strain of tensile fiber	0.012
Ultimate strain of tensile fiber	0.014
Tensile fiber ultimate damage	0.99
Initial strain compressive fiber	0.008
Ultimate strain compressive fiber	0.009
Compressive fiber ultimate damage	0.99
Initial yield stress	0.02 GPa
Hardening law exponent	0.64
Hardening law multiplier	1.3
Shear modulus 1,2 plane	7 GPa
Shear modulus 2,3 plane	4 GPa
Shear modulus 1,3 plane	4 GPa
Poisson's ratio	0.33
Critical transverse damage limit	1
Initial transverse damage limit	0.02

APPENDIX B: Predictions on remaining new dataset

Table B.1: $n_{ls} = 4$; t_l, v_p, T_i, T_{pd}, T_{air} randomly selected (from table 2.1); $\mathbf{\Phi}_{fib}$ different from training and testing

Outputs	0.5 mm, 5.8 m/s, 239 °C, 68 °C , 16 °C, (0, 45, -45, 60)			0.1 mm, 4.9 m/s, 206 °C, 219 °C , 25 °C, (0, 45, -45, 60)		
	Sim.	GPR	% err.	Sim.	GPR	% err.
F_p	1010	976.2	**3.35**	1010	998.6	**1.12**
CLE	0.545	0.583	**6.97**	0.566	0.607	**7.08**
SEA	14.17	12.91	**8.88**	15.46	13.73	**11.16**
ΔY_{node}	16.72	17.31	**3.50**	16.81	18.39	**9.40**

Table B.2: $n_{ls} = 4$; v_p, T_i, T_{pd}, T_{air} and $\mathbf{\Phi}_{fib}$ randomly selected (from table 2.1); t_l different from training and testing

Outputs	0.7 mm, 5.8 m/s, 239 °C, 68 °C , 16 °C, (0, 45, -45, 90)			0.8 mm, 4.9 m/s, 206 °C, 219 °C , 25 °C, (0, 45, -45, 90)		
	Sim.	GPR	% error	Sim.	GPR	% error
F_p	974.0	1017.6	**4.43**	971.0	1051.4	**8.28**
CLE	0.562	0.579	**3.11**	0.569	0.578	**1.74**
SEA	14.79	12.62	**14.64**	14.46	13.46	**6.90**
ΔY_{node}	16.33	17.29	**5.86**	16.34	17.50	**7.11**

Table B.3: $n_{ls} = 4$; v_p, T_i, T_{pd}, T_{air} and $\mathbf{\Phi}_{fib}$ randomly selected (from range in table 2.1); t_l different from training and testing

Outputs	0.7 mm, 6.5 m/s, 318 °C, 131 °C , 24 °C, (30, -30, 60, -60)			0.9, 5.7 m/s, 304 °C, 91 °C , 22 °C, (30, -30, 60, -60)		
	Sim.	GPR	% error	Sim.	GPR	% error
F_p	1020.0	1037.9	**1.76**	1010.0	1091.2	**8.04**
CLE	0.540	0.577	**6.9**	0.554	0.582	**4.98**
SEA	13.61	12.21	**10.26**	13.51	14.26	**5.57**
ΔY_{node}	16.84	17.49	**3.89**	16.65	17.82	**7.10**