Prediction of Crashworthiness Performance Using Multi-Fidelity Machine Learning Techniques

Master of Science Thesis

Panagiotis Koronaios

Delft University of Technology

TU Delft

Prediction of Crashworthiness Performance Using Multi-Fidelity Machine Learning Techniques

by

Panagiotis Koronaios

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on Thursday April 11, 2024 at 12:00 PM

Student number:	5690412	
Project duration:	October 11, 2023 - April 11, 2024	
Thesis commitee:	Dr. B. Chen,	TU Delft, chair
	Dr.ir. N. Eleftheroglou,	TU Delft, external evaluator
	Dr.-Ing. S. G. P. Castro,	TU Delft, supervisor
	MSc. H. Maathuis,	TU Delft, supervisor

TU Delft

kopter

"By believing passionately in something that still does not exist,
we create it"
Nikos Kazantzakis

Acknowledgements

During my postgraduate studies at TU Delft, I had the privilege of gaining hands-on experience in structural analysis within the aerospace industry. Completing my MSc thesis at Kopter was an invaluable experience that significantly contributed to my learning and growth. First, I would like to thank my supervisor and manager at Kopter Konstantinos Arvanitopoulos and the Chief Engineer of Structures Dr. Taqi Khan for all the insight they provided me during my graduation project. Additionally, I extend my gratitude to Hauke F. Mathius, PhD candidate and co-supervisor, for the guidance, thoughtful discussions, and exchange of ideas that contributed to the completion of the project. Moreover, I want to express my sincere appreciation to my supervisor at TU Delft, Dr. Saullo G. P. Castro for his valuable guidance and support during the project. Networking with other members of the CrashProof Knowledge Centre was also highly beneficial, particularly with Shreyas Anand, another PhD candidate that provided the low- and high-fidelity models that were used in this thesis. Finally, I would like to thank my family and friends for their constant encouragement and support that was the driving force behind my endeavors.

Panagiotis Koronaios
Zurich, April 2024

Preface

The motivation behind this thesis is to investigate crashworthiness assessment of helicopter structures. Given the scarcity of data from computationally expensive simulations and experiments, and the need of performing design optimization for crashworthiness efficiently, the focus is on meta-modeling techniques. The study compares machine learning approaches for predicting energy absorption and load-displacement curves in tubular metallic structures, highlighting the efficacy of machine learning to predict structural behaviors. After building surrogate models solely based on high-fidelity data from experimentally validated numerical models, the incorporation of low-fidelity data to enhance energy absorption predictions in a multi-fidelity framework is investigated. The methodologies and approaches presented in this research aim to construct surrogates that should be suitable for employment within crashworthiness optimization and preliminary design processes.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
EASA	European Aviation Safety Agency
FAA	Federal Aviation Administration
OEM	Original Equipment Manufacturer
CS27	Certification Specifications and Acceptable Means of Compliance for Small Rotorcraft
ATD	Anthropomorphic Test Dummy
CAD	Computer Aided Design
FEA	Finite Element Analysis
FEM	Finite Element Model
SO Regr.	Single-output Regressor
MO Regr.	Multi-output Regressor
MAPE	Mean Absolute Percentage Error

Symbols

Symbol	Definition	Units
M_0	Fully plastic bending moment per unit length	[N]
σ_0	Flow stress	[MPa]
ε	Strain	
ε_u	Ultimate strain	
N_c	Number of corners for a polygonal cross-section	
P_{mean}	Mean crushing load	[N]
h	Wall thickness	[mm]
κ	Effective crushing length	[mm]
c	Edge length of the cross-section in polygonal tubular cases	[mm]
$2H$	Initial distance between plastic hinges	[mm]
R^2	Coefficient of determination	
X, Y and Z	Coefficients of Equation 3.2.6	

Abstract

This study investigates the development and application of meta-models for crashworthiness assessment of helicopter structures and components. It aims to address the challenges associated with scarcity of data from computationally expensive simulations and experimental drop-tests, and enable the use of surrogates in a crashworthiness optimization framework. Two predictive approaches utilizing Machine Learning techniques are compared to predict and assess the energy absorption of tubular metallic structures for different cross-section configurations. The first approach directly predicts energy absorption, while the second predicts load-displacement curves, from which energy absorption is derived. Results indicate that certain regressors, such as the Transform Target Regressor, the Decision Tree Regressor and the Poisson Regressor, consistently achieve high accuracy in predicting load-displacement curves and energy absorption across the evaluated tubular samples. A low-fidelity model able to provide less accurate but computationally inexpensive information is then introduced. The influence of low-fidelity data is investigated when it serves as additional input alongside high-fidelity data during the training phase of the surrogate model, through a comparative analysis. The research's findings suggest the efficiency of Machine Learning in representing structural behaviour under crushing conditions and highlight the potential for further enhancements through the integration of low-fidelity data, thereby holding promise for extending the methodology to more complex structures.

1 Introduction

Helicopters are a type of aircraft that utilize rotating wings to provide lift force, propulsion, and control. In contrast to fixed-wing aircraft, helicopters are capable of generating aerodynamic force even without forward movement. This unique characteristic enables them to perform translational flight, hover flight, and vertical take-off and landing, making them extremely versatile among aircraft types.

Crash protection for occupants in aircraft has always been a significant concern, even from the early days of powered flight. Initially, measures such as helmets and leather jackets were introduced to prevent head injuries and abrasions during crashes. Seat belts were also designed to secure pilots during acrobatic maneuvers but were soon recognized as valuable for maintaining occupants in the event of a crash. However, it was not until the 1940s that researchers like Hugh DeHaven began to approach crash survivability as a comprehensive system [1]. These early efforts paved the way for the development of more advanced and standardized crashworthy designs in the subsequent years.

In recent decades, there has been an increased emphasis on helicopter safety due to a rise in accidents resulting from structural failures or human errors [2]. Helicopters have been found to have a 17.3 times higher risk of fatal accidents compared to passenger cars [3]. A fatal accident is defined as an operational incident wherein at least one individual sustains fatal or severe injuries due to various circumstances. These include being inside the aircraft during the incident, or direct contact with any part of the aircraft (including detached components). Notably, injuries resulting from natural causes, self-inflicted actions, or those inflicted by third parties are excluded from this definition. Additionally, injuries to stowaways concealed outside the typical passenger and crew accessible areas are not considered fatal accidents in this context [4]. Although traditional helicopter designs consider factors such as static strength, fatigue strength, aerodynamics, stability, and control capability to ensure flight safety, there still remains a risk of significant damage to major structural components and severe injury to occupants in the event of a crash. The impact force resulting from such crashes can directly impact passengers and impede crucial escape routes. When the impact energy exceeds human tolerance, the consequences can be grave, leading to severe injuries and casualties.

To address these issues, it is crucial to implement crashworthy design principles in helicopter structures. Analysis of helicopter crash accidents has identified several factors contributing to occupant injuries: inertial overload due to sudden acceleration or deceleration, contact injuries resulting from collisions with hard surfaces within the cabin, and environmental injuries such as fires caused by fuel leakage, asphyxiation from smoke and fumes, and drowning in water. Studies on helicopter accidents have shown that 90% of these incidents offer survival conditions for occupants [5]. Therefore, the significance of crashworthiness design has markedly increased in the advancement of helicopters.

The basic principles of crashworthiness design may be summarized by the acronym

"CREEP" as follows [1]:

- C – Container: should possess sufficient strength to prevent intrusion of structure into occupied spaces during a survivable crash.
- R – Restraint: seats, restraint systems and their attachments should have sufficient strength to retain all occupants for the maximum survivable crash pulse.
- E – Energy absorption: locations, where vertical energy absorbing capability may be integrated into a helicopter design, include landing gear, floor structure and the seats.
- E – Environment (local): any object within the passenger space may be considered an injury hazard.
- P – Postcrash factors: provide for the escape of occupants after the crash under a host of adverse conditions. Control or eliminate the hazard at the source or provide for more rapid egress, or a combination of both.

The Federal Aviation Administration (FAA) and European Aviation Safety Agency (EASA) are the regulatory agencies to address the crashworthiness principles and standards for aircraft structures. Specifically the AW09 aircraft designed by Kopter Group AG adheres to the EASA standards for small rotorcraft (CS27). The civil crashworthiness requirements for small rotorcraft CFR 27/CS 27 are provided in the in the Federal Aviation Administration (FAA) 14 CFR Airworthiness Standards [6] and EASA Certification Specifications [7].

The CS 27.561 paragraph specifies that the rotorcraft, must be designed to protect occupants during emergency landings on land or water, despite potential damage. The structure should provide occupants with the best possible chance of avoiding severe injury in the event of a crash landing given the proper use of seats, belts, and safety design features, especially when wheels are retracted.

Occupants and any cabin items with the potential to harm must be restrained under specified inertial load factors relative to the surrounding structure: upward 4 g, forward 16 g, sideward 8 g, downward 20 g considering the intended seat device displacement, rearward 1.5 g. The supporting structure must be designed to secure any potentially hazardous mass located above or behind the crew and passenger compartment during an emergency landing. These masses may include rotors, transmissions, and engines. The design must ensure they remain restrained for the following inertial load factors: upward 1.5 g, forward 12 g, sideward 6 g, downward 12 g, and rearward 1.5 g. Fuselage structures containing internal fuel tanks below the passenger floor level must be designed to withstand specified inertial factors and loads. Furthermore, they must protect the fuel tanks from rupture when subjected to these loads: upward 1.5 g, forward 4 g, sideward 2 g, and downward 4 g.

The emergency landing dynamic conditions are addressed in the CS 27.562 paragraph. The dynamic compliance tests should be conducted using a 77 kg anthropomorphic test dummy (ATD) in a normal upright sitting position. The tests should be conducted considering the following. Change of no less than 9.1 m/s (30 ft/s) in downward velocity with the seat inclined at a 60-degree angle to the direction of impact. The peak floor

deceleration must occur in not more than 0.031 seconds after impact and must reach a minimum of 30 g. Change of no less than 12.8 m/s (42 ft/s) in forward velocity with the seat in its nominal position. The maximum floor deceleration should be achieved within a time frame of 0.071 seconds following the impact, and it should reach a minimum value of 18.4 g. Where floor rails or attachment devices on the floor or sidewall are employed, they must exhibit a minimum misalignment of 10° in a vertical (pitch) direction and a 10° lateral roll to account for potential floor warping.

The seating device system and the attachment between the seating device and the airframe structure must remain intact, even if the structure has exceeded its limit load. The occupant retention system must effectively handle dynamic loads. Permanent seat deformations should remain within predefined limits, ensuring they do not significantly hinder an occupant's ability to release shoulder harnesses, stand, and exit the seat. In case the ATD's head is exposed to impact, the Head Injury Criterion (HIC) should not exceed 1000. The expression for HIC is provided in Equation 1.0.1:

$$HIC = (t_2 - t_1)\left[\frac{1}{t_2 - t_1}\int_{t_1}^{t_2} a(t)\,dt\right]^{2.5} \tag{1.0.1}$$

where $a(t)$ is the resultant acceleration at the centre of gravity of the head form expressed as a multiple of g and $t_2 - t_1$ is the time duration in seconds of major head impact, which should not exceed 0.05 seconds.

Provided that upper torso harness straps are employed, individual strap tension loads should not exceed 7.78 kN (1,750 lbs). If dual straps are used, the total strap tension load should not exceed 8.90 kN (2,000 lbs). The maximum compressive load measured between the pelvis and the lumbar column of the ATD should not exceed 6.67 kN (1,500 lbs). Each upper torso restraint strap must remain securely in place on the ATD's shoulder during the impact. The pelvic restraint must remain securely fastened to the ATD's pelvis throughout the impact.

These certification requirements serve as essential guidelines within the optimization framework for designers in order to achieve a crashworthy helicopter design. By adhering to EASA and FAA requirements, designers are compelled to integrate safety features, structural integrity, and performance considerations into their designs from the outset. This approach ensures that safety concerns are systematically addressed, resulting in helicopters that not only meet regulatory standards but also prioritize the safety of occupants in the event of a crash.

In the certification process of aircrafts, the following steps are undertaken:

- Type certification: this phase involves the certification of the design itself. In the context of a helicopter, this step necessitates the Original Equipment Manufacturer (OEM) to demonstrate compliance with crashworthiness requirements and type-certification standards, such as CS-27 (pertaining to small rotorcraft) and CS-29 (pertaining to large rotorcraft). The objective is to ensure initial air-

worthiness. According to the European Aviation Safety Agency (EASA), "Initial airworthiness is demonstrated by a certificate of airworthiness issued by the civil aviation authority in the state in which the aircraft is registered" [7].

- Continued airworthiness: this stage is characterized by the ongoing maintenance actions necessary to maintain airworthiness. It involves aspects such as damage tolerance and defect inspection, ensuring the sustained operational integrity of the aircraft.

Airworthiness requirements encompass various factors, including the legal and physical condition that an aircraft must meet to ensure its safe operation. According to ICAO Annex 8 [7], the term "airworthy" is defined as "the condition of an aircraft, engine, propeller, or part when it adheres to its approved design and is in a safe operational state." In order to assess the overall design and construction of an aircraft, certification authorities thoroughly review all aspects, even if there are perceived advancements beyond minimum standards. Once an aircraft is deemed compliant with all certification requirements, it is issued a Type Certificate (TC).

In the certification process for aerospace structures, Original Equipment Manufacturers (OEMs) employ physical tests, such as drop-tests, as a means to demonstrate compliance with type certification requirements. These tests also serve as validation for the numerical models developed. Finite Element Analysis (FEA) assumes a pivotal role in this process, simulating aircraft components and systems across various flight conditions. FEA evaluates landing gear integrity, aerodynamics, thermal stress, fatigue life, vibrations, and fuel usage, ensuring the safety and efficiency of aerospace structures and operations. In the quest for crashworthy designs, conducting finite element simulations and validating them through experimental drop-tests in accordance with safety standards and regulations outlined by EASA and FAA is imperative. The simulation environment is intricately linked with virtual testing, enabling the evaluation of numerous cases and facilitating the creation of more robust designs while simultaneously reducing the costs associated with the certification campaign.

Given the substantial computational resources required for experimentally validated numerical models and the formidable challenges associated with drop-tests, such as prolonged preparation periods, correlation with the respective numerical models and exorbitant costs, the data providing insights on the structural behavior under crash conditions remain scarce. Furthermore, employing computationally intensive finite element simulations within a design optimization framework presents a daunting challenge. It would be an arduous endeavor to conduct optimization (either for a single component or the whole helicopter subfloor) aimed at obtaining a crashworthy design, especially considering that running a single load case may require hours of explicit finite element analysis. An optimization aiming in enhancing crash performance would require large number of function evaluations, each representing the crash response of the structure. In pursuit of these goals, the development of highly efficient meta-models for prediction and design emerges.

1.1. Purpose of the Thesis

The present study focuses on addressing the challenges inherent in crashworthiness assessment for helicopter structures and components when these are represented by surrogate models. Rather than analysing an extensively detailed helicopter subfloor, the crashing of tubular structures is utilized in this dissertation. The main reason for choosing a tubular structure is that there are available experimentally validated data in the existing body of literature. Such data are not easily accessible for cruciforms or helicopter subfloors. This selection made the implementation of the methodology developed a computationally feasible task and valuable insight can be gained within the context of crash performance.

Taking advantage of a plethora of algorithms and tools recently available in the research area of Machine Learning, the focus is on employing Machine Learning techniques to predict the structural behavior under crashing conditions. Another reason behind this selection is that Machine Learning has been used before in the literature in the context of crashworthiness which means that this study could leverage previous findings.

The high-fidelity data for this case-study consist of load-displacement curves based on which the crashworthiness performance of the structure is evaluated. One approach is to perform this assessment based on the energy absorption, which corresponds to the integral of the load-displacement curve, as scalar value. Another approach is to evaluate the entire load-displacement curve and based on it evaluate crashworthiness. A first hypothesis can be formulated here: **the evaluation of the entire load-displacement curve can become a more valuable input for the Machine Learning models and give better insight on the structural behavior of the structure instead of utilizing the energy absorption as a scalar value.**

Due to the limited availability of high-fidelity data, there is a necessity to introduce low-fidelity models capable of producing less accurate information through computationally less demanding methods. Here a second hypothesis can be formulated: **by introducing low-fidelity data along with high-fidelity data in the training process of the surrogate model within a multi-fidelity framework the predictions of the energy absorption can become more accurate.**

The primary objective of the thesis is to identify the influence in the accuracy of the predictions by introducing low-fidelity information, which are less accurate but highly more efficient, in the training phase of the meta-models. The efficiency of surrogate models trained solely with expensive / accurate numerical data is compared with that of surrogate models trained with both expensive / accurate numerical data and cheap / less-accurate data, aiming to assess the impact of incorporating low-fidelity information when predicting the energy absorption. The ultimate goal is to contribute valuable insights that will enable crashworthy design based on predictions made via meta-modeling and the use of surrogates in crashworthiness optimization.

1.2. Structure of the Report

The report comprises of 7 chapters. Chapter 1 refers to the present chapter and aims to introduce the reader to the report with important definitions and background crashworthiness information on helicopters. Chapter 2 includes the state of the art articles and studies focused on crashworthy helicopter subfloor designs, surrogate modelling and optimization, along with the research questions addressed. Chapter 3 describes a numerical and an analytical model for tubular metallic structures that was utilized within the study. The methodology adopted is thoroughly described in Chapter 4. The results are presented and discussed in Chapter 5 and the conclusions are documented in Chapter 6. Finally, recommendations on future work are given in Chapter 7.

2 Literature Review

The purpose of this chapter is to cover the relevant aspects of helicopter crashworthiness based on research done until the present dissertation is written. The research areas that are described and will be the basis to address the research questions and the objective of the thesis, are the crashworthy design of helicopter subfloor, crashworthiness requirements and evaluation criteria, surrogate modelling, and optimization for structural crashworthiness.

2.1. Crashworthy Design of Helicopter Subfloor

Aircraft crashworthiness refers to the aircraft system's ability to secure occupants from severe or fatal injuries in the event of a crash. It encompasses two crucial aspects of aircraft crash-resistant design, impact reduction through energy-absorbing structures and human tolerance [8].

When the vertical component of the impact energy is high the crash loads should be mainly absorbed by controlled structural deformation. A systems approach should be followed which consists of the landing gear, the subfloor, the mass retention structure, the seat, and the restrain systems as depicted in Figure 2.1. For helicopters, the overall crash behavior highly depends on the design of beams intersections, bulkheads (cruciforms), the beam webs, and the boxes in the floor structures. Typical helicopter subfloor elements are shown in Figure 2.2.

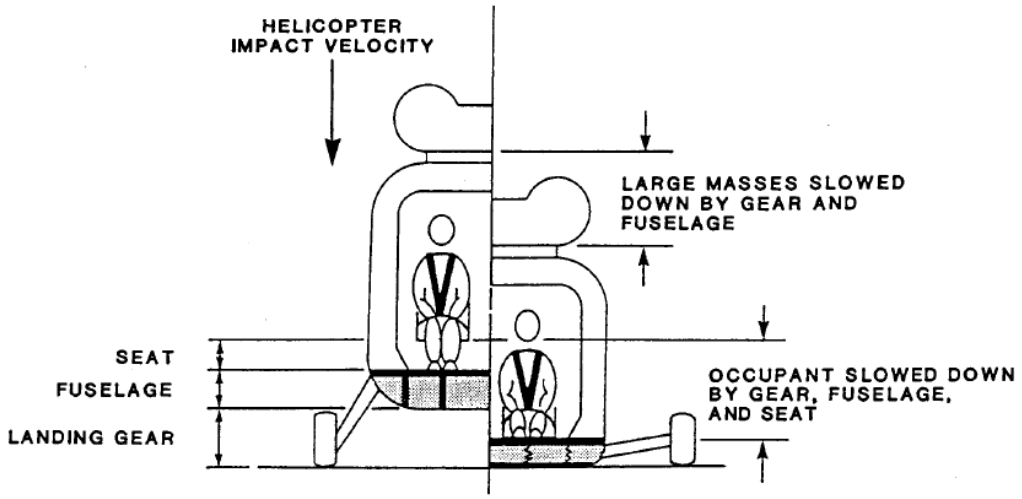


Figure 2.1: Rotorcraft crashworthiness concept [9]

Beam and frame elements for rotorcraft subfloor structures require a dual structural concept. These elements must be designed to carry flight loads during operation and crash loads in case of impact [10]. In other words, designing for crashworthiness comprises designing for structural integrity and energy absorption. The requirements for the structural design must be well-defined. For helicopters, survivability must be guaranteed

according to specified requirements which is essential to define relevant optimization objectives. A standard optimization objective for lightweight structures is to find minimum weight, in which case the crashworthiness requirements should be interpreted as constraints.

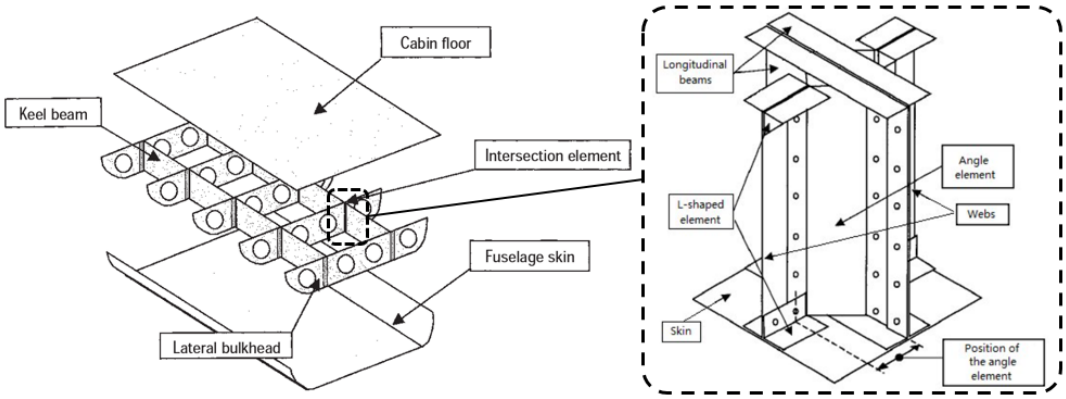


Figure 2.2: Typical helicopter subfloor and subfloor intersection element [11, 12]

It has been shown that composite materials show a significant capacity for absorbing kinetic energy in the event of a crash [13–15]. Using composites improved safety and lightweight designs can be obtained. Buckling and folding are the common failure modes for metallic structures under crash loads, which allow them to absorb crash energy through material yielding and plastification along the folding lines. In composite materials though the process of microfragmentation depends on various parameters. The aim is to initiate the failure mechanisms and to obtain a controllable crushing behavior [16].

Full-scale crash testing is one of the most important methods to assess the crashworthy design of helicopters. These tests require long preparation period and increased experimental costs. For these reasons, a meticulous experimental framework is imperative to capture an extensive range of structural responses. These responses play an important role in evaluating helicopter crashworthiness and validating numerical crash models. NASA Langley Research Center's Landing and Impact Research (LandIR) facility has conducted over 100 full-scale crash tests. Notable helicopter crash tests conducted at LandIR include those involving the ACAP, CH-47, UH-60, and AH-1, as depicted in Figure 2.3.

These tests were crucial for evaluating the crashworthiness characteristics and dynamic responses of small representative helicopters. They also provided empirical data to validate finite element models used in helicopter crash simulations. Validated computational tools offer a cost-effective means of simulating the majority of crash scenarios compared to conducting full-scale crash tests, contributing to the overall understanding of helicopter crash performance. Another objective of conducting such full-scale tests is to validate innovative energy absorption concepts on actual airframe structures.

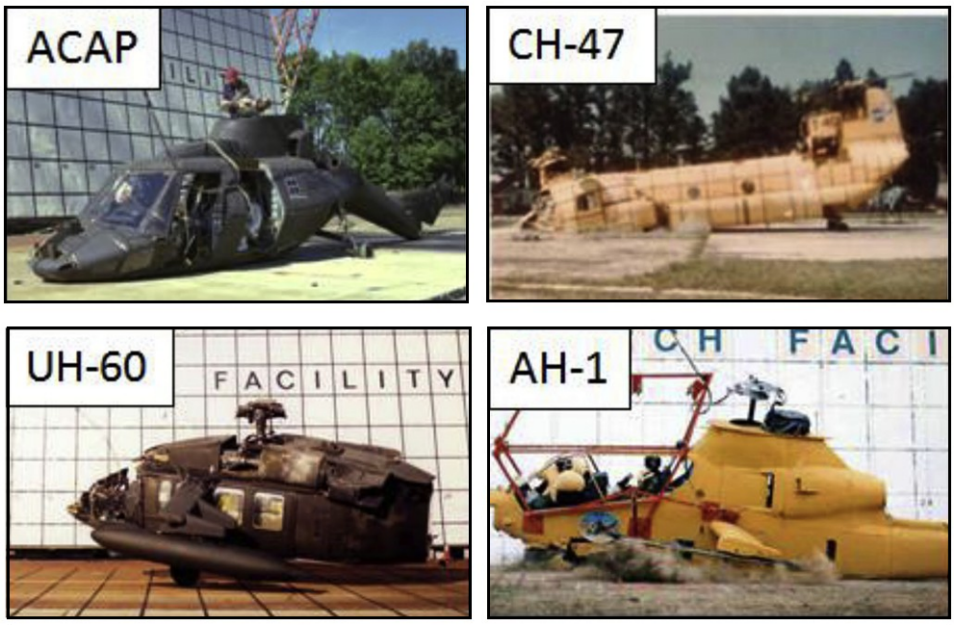


Figure 2.3: Full-scale helicopter crash tests [8]

Depending on the test conditions, the materials used, the assumptions made, and the strain rain regimes a classification of experimental techniques is made depicted in Figure 2.4. Helicopter crash tests belong to high strain rates regime of the graph while bird strike tests belong to very high strain rain rates regime of the graph.

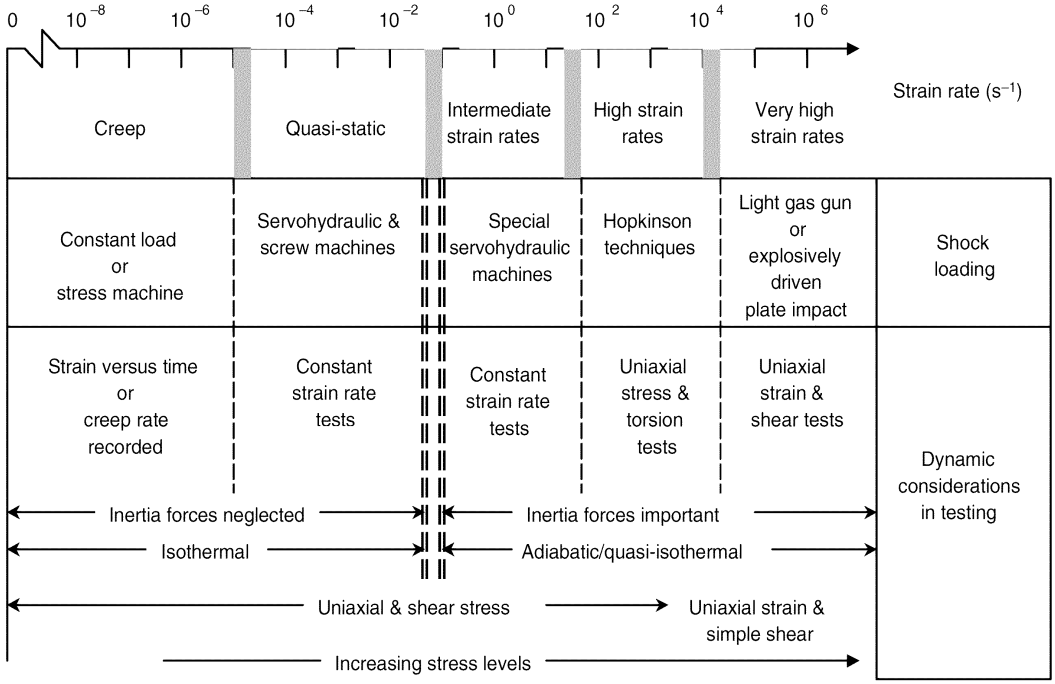


Figure 2.4: Classification of experimental techniques based on testing conditions and assumptions [17]

The analytical models are typically well correlated with experimental test results for global parameters, such as landing gear or engine response. However, the non-linear,

transient dynamic responses of helicopter structures hinders the development of analytical predictions for localized responses, such as stresses in components at a given time during a crash [18]. Non-linear responses may arise due to high peak loads to which structures are subjected, plastic deformations induced by inertial forces, alterations in boundary conditions, material and joint failures, high rates of deformation, contact forces, and large deformations which are commonly encountered in crash conditions [19]. Numerical simulations serve as an essential tool to evaluate crash scenarios not economically feasible with full-scale drop-tests. In the aerospace domain, the topic of helicopter crashworthiness has been a subject of in-depth deliberation, and comprehensive investigations have been diligently conducted.

Kermanidis et al. [20] in 1970 pioneered the study of the energy absorption behavior of helicopter subfloor components such as sin-wave beams and "tensor skin" panels with experiments and numerical simulations. The water impact scenario was mainly investigated while composite material damage models were developed capable of representing successfully the properties degradation. The failure modes of subfloor components necessitated the selection of appropriate contact algorithms and the creation of adequately refined meshes to accurately predict these failure modes during drop-tests.

Frese and Nitschke [21] investigated numerically and experimentally the structural behaviour of helicopter subfloor structures under crash impact loading. The performance of stiffened and honeycomb sandwich panels, made of metallic materials under quasistatic and dynamic conditions, was examined to obtain an optimized crashworthy structure, and to establish the non-linear characteristics of subfloor structures. An efficiency factor was introduced for the energy absorption evaluations, defined as the ratio of the absorbed energy to the product of the peak load and the stroke length. Within this paper was emphasized that in the design of crashworthy structures, it is essential to shift away from the conventional focus on achieving high specific strength and stiffness and, instead, prioritize the design for controlled failures and controlled deformations.

Farley et al. [13] developed analytical formulations to predict the energy-absorption capability of helicopter subfloor beams with circular and square cross sections. Sandwich, sine-wave and two integrally stiffened designs were evaluated in this study. From these concepts the sine-wave beams had the best energy absorption performance. One of the main outcomes of the study was that graphite/epoxy subfloor structures absorbed energy more efficiently compared to Kevlar or aluminum structures.

Boitnott and Kindervater [22] investigated the crush behavior of cruciforms and sine-wave beams considering them representative of keel beam and bulkhead intersections found in rotorcraft subfloor. Non-linear dynamic analysis with the finite element code DYCAST was performed and quasistatic tests on aluminum and composites cruciform elements were conducted aiming to initiate stable crushing failure modes. From the evaluated designs only one showed stable failure mode. It was a hybrid cruciform made of a mixture of carbon and aramid fibers. The design variants of the aluminium and composite cruciforms examined in this study are depicted in Figure 2.5. Implementation to the subfloor design, though, would require considerable manufacturing efforts. The

sine-wave subfloor beams were the most efficient in absorbing energy, result also shown in [13, 16].

Kindervater and Georgi [16] pointed out the limitations of analytical models stemming from the complexity and the multi-parameter nature of the problem. Frictional effects, viscoelastic and viscoplastic behavior of the elements, and the microcracking responses should be considered in order to have representative enough models. The load-deflection curves are extensively used for crashworthiness evaluation of the concepts proposed.

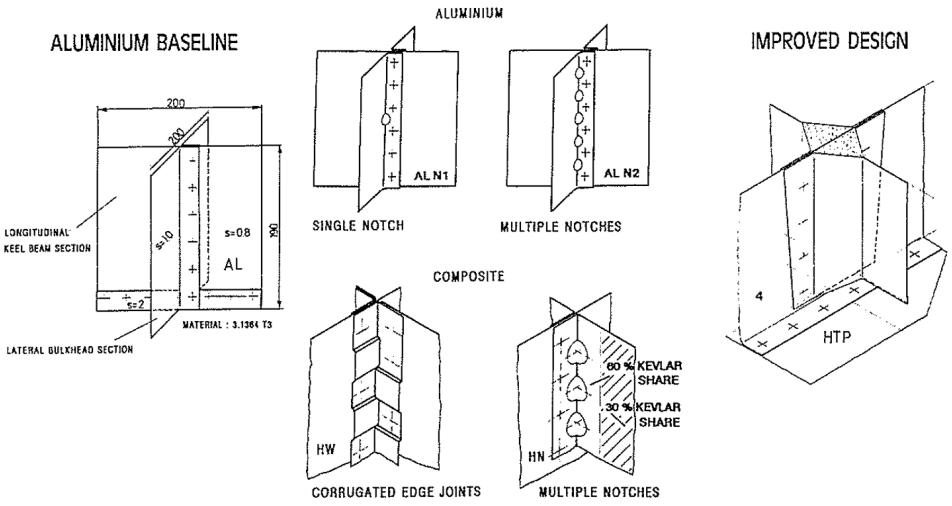


Figure 2.5: Cruciform design variants [22]

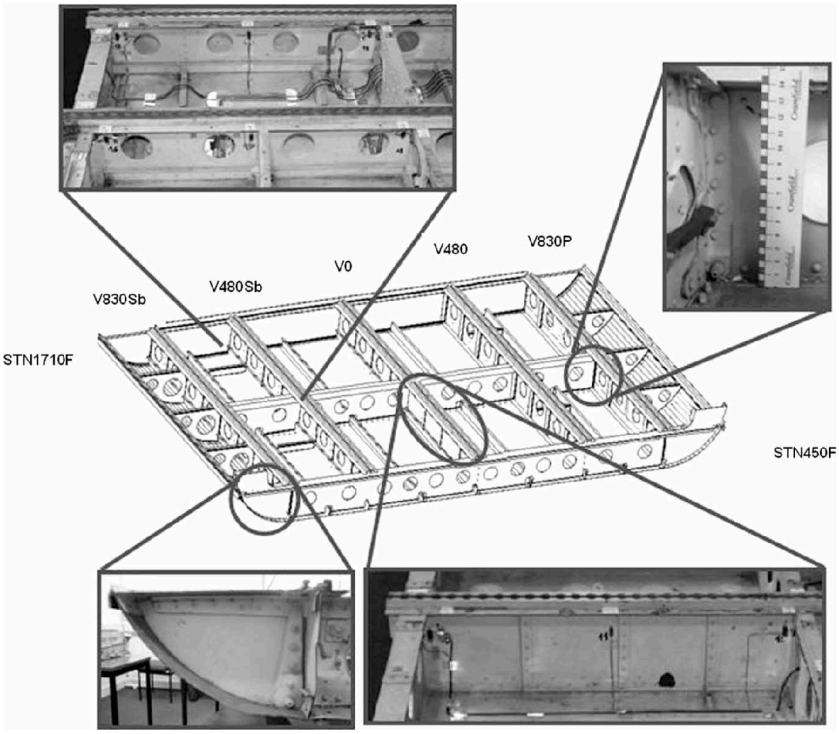
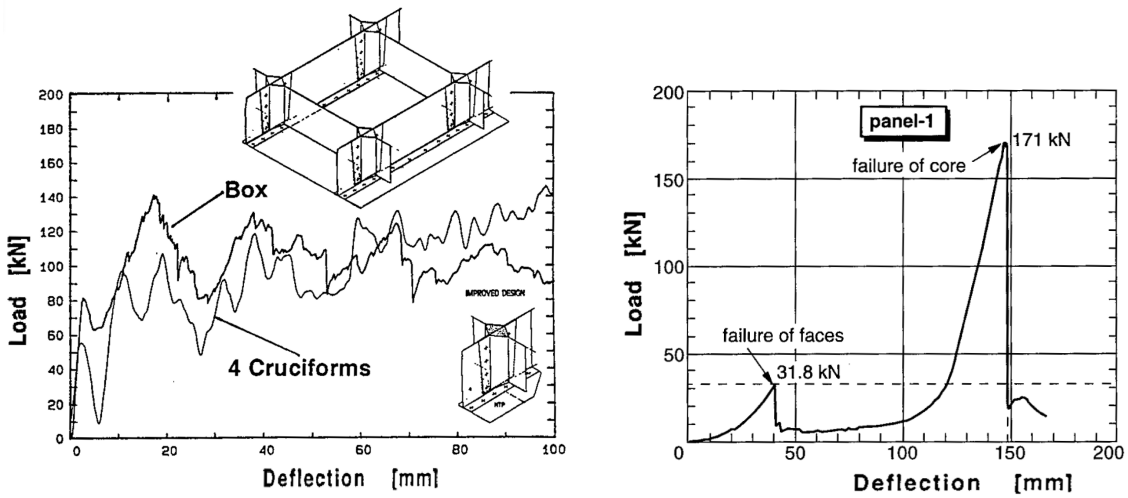


Figure 2.6: Schematic diagram of simple box-beam underfloor structure [23]

Johnson et al. [10] described the incorporation of composites to the design of crash resistant beam and frames for helicopter subfloor structures. The most efficient yet evaluated subfloor beams were composite beams with sinusoidal or trapezoidal web profiles. Their main drawbacks, however, include the high manufacturing costs, the interface issues with other structural components, and the difficulty to trigger failure mechanisms. The load-deflection curves for subfloor box and cruciforms of the tests conducted are shown in Figure 2.7. Other studies that conducted experiments in box structures include [23]. The schematic diagram of the simple box-beam underfloor structure that was experimentally evaluated with a drop-test is shown in Figure 2.6. Observing the form of these curves is crucial in the training phase of the surrogate model when employed to represent the structural behavior of a crushing structure.



(a) Cruciform and subfloor box **(b)** Tensor-skin panel

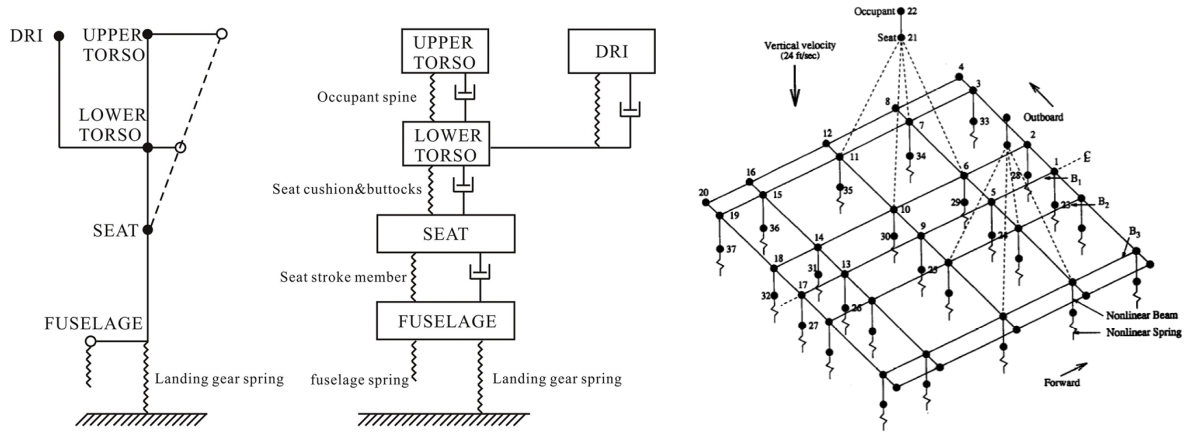
Figure 2.7: Load-deflection curves for subfloor box and cruciforms [10]

The previous studies involved both numerical simulations and experimental analyses to explore the crash performance of helicopter subfloor elements. The goal was to gain a deeper comprehension of failure modes and the energy absorption performance of these components. The findings aimed to provide practical guidance and recommendations for the implementation of crashworthy design concepts in helicopters.

A different approach was proposed by Hajela and Lee [24] who addressed a topological optimization of the subfloor structure using genetic algorithms for improved crash performance approximating response models with multilayer perceptron Neural Networks (NN). NN predicted the Dynamic Response Index (DRI) values, in a range of 0.7-4.7 % compared to the values derived from the numerical model. The subfloor was considered as a crush zone on top of which a rigid floor structured was installed. The DRI is an indicative of the likelihood of severe spinal injury to the occupants [25]. The definition of DRI injury model is mentioned in [26] as a one-dimensional spring-mass-damper system.

The approach adopted in this study was aiming to determine both the optimum load-deflection response of a structural element and its location in the subfloor structure. The

KRASH analysis program was used to simulate the structural behavior of the subfloor. Lumped masses, linear and non-linear beam elements, and spring elements were employed to obtain a representative model depicted in Figure 2.8 (b). A simplified model that includes the landing gear, the fuselage, the seat, and the lower and upper torso can be seen in Figure 2.8 (a).



(a) Simplified model [27] **(b)** Subfloor model [24]

Figure 2.8: Simplified helicopter numerical models

Kohlgruber and Kamoulakos [28] examined the crash behavior of composite box structures in helicopter subfloor assemblies, focusing on numerical simulations with PAM-CRASH. Their work encompasses material data, validation, and numerical simulations. The initial and the improve concepts can be seen in Figure 2.9. Future advancements was recomended to focus on integrating strain-rate effects and delamination processes into the shell materials model. It was underscored that these enhancements are important, particularly for shells, as they are the sole elements viable for analyzing larger structures, primarily due to computational expenses.

Kindervater et al. [14, 29] presented crashworthy design principles for aircrafts and discussed the energy dissipation at structural element level. The helicopter frame structure consists of frames and longitudinal beams to support the the outer skin and cabin floor. The subfloor structure is composed of keel beams and lateral bulkheads which form the subfloor boxes. Beam elements are designed to deform and absorb the energy during crash conditions.

The frames and shell structures in the cabin should ensure survivable conditions for the occupants, preventing collapse and potential intrusion from other components. Finally, the subfloor must provide structural integrity for the cabin, absorb the impact energy, and reduce the crash vibrations transmitted to the occupants. This system approach with reference to the key design concepts and structural elements for a crashworthy aircraft is illustrated in Figure 2.10. The deceleration forces should be limited by structural deformation through a controlled load concept shown in Figure 2.11 where the the botom part of the rotorcraft aiframe consists of a high strength cabin floor and an energy absorbing crushing zone.

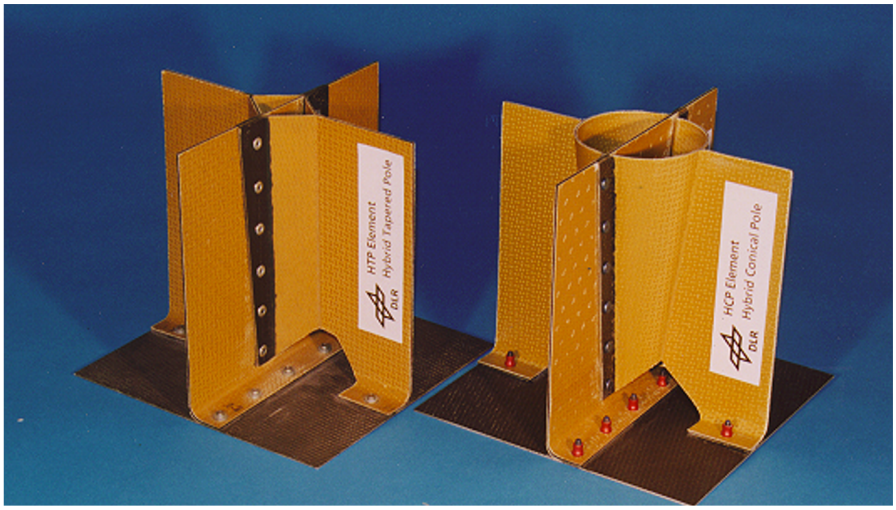


Figure 2.9: Initial (left) and improved (right) cruciform element design [28]

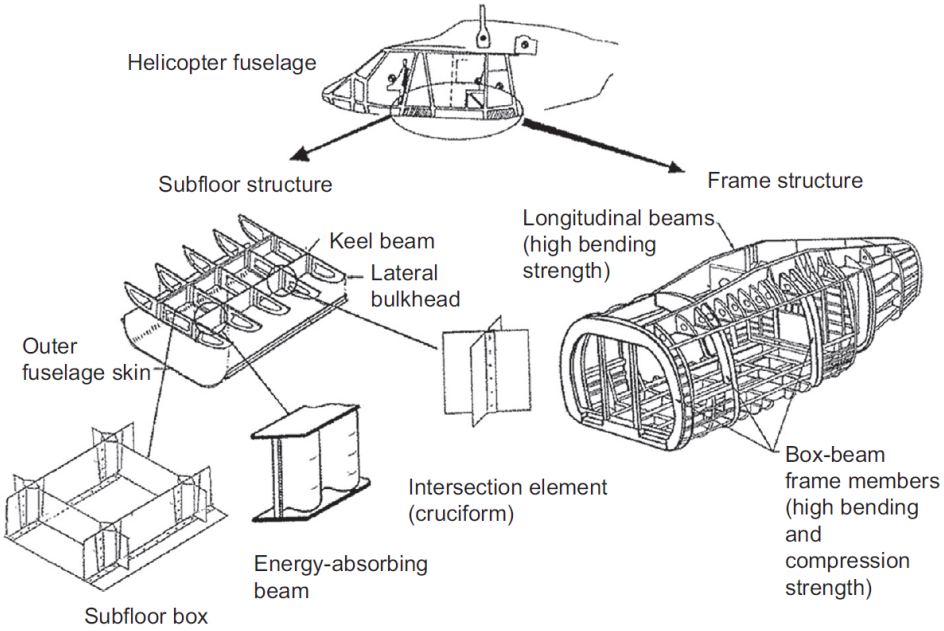


Figure 2.10: Crashworthiness design concepts for helicopter structures [14]

Bisagni [11, 30, 31] investigated the energy absorption of helicopter subfloor made of aluminum aloy in cooperation with Agusta. Experimental drop-tests were conducted and finite element analyses of the subfloor riveted intersections were performed since these elements can be responsible for high deceleration peak loads at the cabin floor and then to the occupants. The analysis with finite elements included detailed geometrical models, materials models and the proper modeling of contact forces and rivets. While these studies contribute significantly to understanding the crashworthiness performance of subfloor elements, they are limited and they focus solely on aluminum materials, neglecting the exploration of composites and their relevant failure modes.

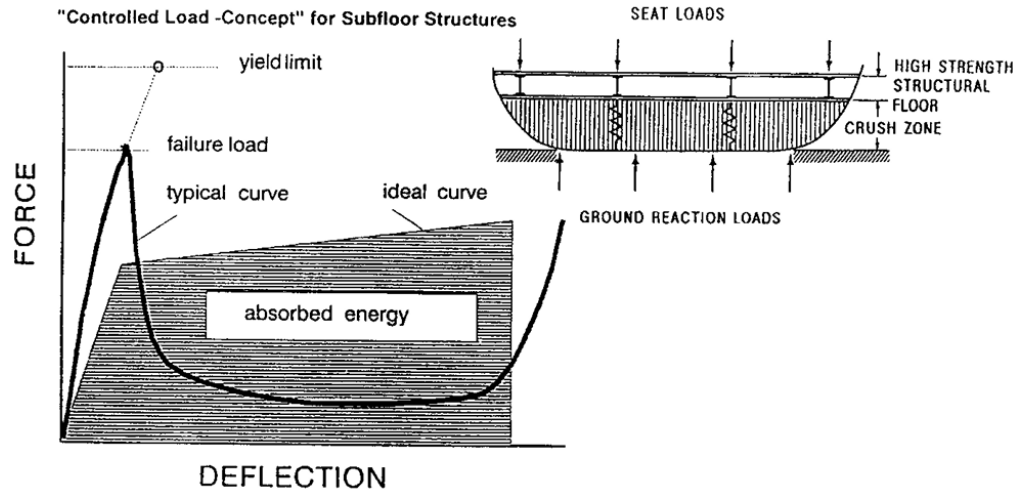


Figure 2.11: Control load concept for helicopter subfloor [14]

Mccarthy and Wiggenraad [32] investigated the response of composite helicopter subfloor subjected to crushing loading conditions with explicit finite element code and drop-tests. It was found that using coupon data as an input provided a reasonable overall representation. The structure used for experimental validation together with the numerical model can be seen in Figure 2.12. The kinetic energy was considered to be mainly converted to the internal energy absorbed by the box elements, the friction energy between the subfloor box elements, and the the friction energy between the subfloor box elements and the top plate of the box. Thus, modeling the frictional phenomena was of outmost importance.

Fasanella et al. [18] developed a finite element model for the Sikorsky Advanced Composite Airframe Program (ACAP) helicopter, using explicit transient dynamic code. They used a two-stage rigid-to-flexible modeling approach to obtain numerical predictions which were later correlated with the full-scale tests. The aim was to verify the capabilities of the explicit code to capture accurately the structural behavior. The forces, velocities, and accelerations were analyzed and correlated between the experimental and numerical models at various significant locations of the aircraft. The last timestep of the experimental and numerical simulation is depicted in Figure 2.13.

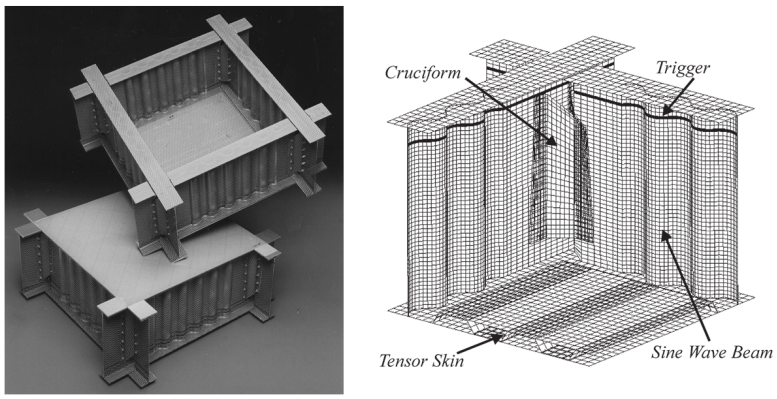


Figure 2.12: Experimental subfloor structure along with the finite element models [32]

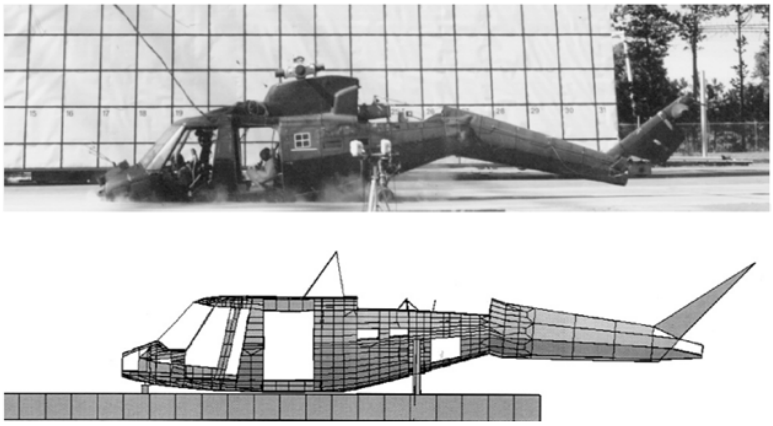


Figure 2.13: Experimental and numerical model of the helicopter structure [18]

Bisagni et al. [33] approximated the structural behavior of helicopter subfloor's structural elements with Neural Networks at crushing conditions trained by finite element analyses. Sequential Quadratic Programming and Genetic Algorithms were implemented to evaluate the objective function after training samples were generated with finite element analyses. The objective function considered was the sum of the crush force efficiency and the specific energy absorbed. Response surfaces generated by Neural Networks were able to reproduce scalar values of maximum and mean forces in order to evaluate the crashworthy efficiency of the configurations proposed by the optimization algorithms.

The same researchers [19, 34] conducted a topology optimization to a helicopter subfloor made of aluminum alloy. To overcome the irregular design domain and the non-linear structural response a decomposition procedure was developed. Global approximation strategies with Neural Networks as basis were again employed for the evaluation of the response of each subsystem. Then a Genetic Algorithm was used to obtain the optimal configuration overcoming the non-linearities and the discrete variables. Even though the outlined approach appears comprehensive in its application to subfloor's structural component, it neglects the broader structural response of the aircraft and its interaction with components that were not modeled, such as the occupants' seats or the landing gear.

Lanzi et al. [35] approximated the structural response under crushing loading of composite cylindrical absorbers with Radial Basis Functions (RBF). Then Genetic Algorithms were coupled with the generated response surfaces to conduct constrained single and multi-objective optimizations. Two years later, a numerical optimization method for a composite intersection element of a helicopter subfloor was developed [36]. Quasi-Newton algorithms and Radial Basis Functions were used together with high-fidelity Finite Element Analyses to optimize the efficiency of the absorber. Effective formulation of the optimization problem necessitated proper parameterization of the shape and precise selection of sample points within the design domain. Throughout the optimization process, a significant decrease in computational effort is attained by substituting the high-fidelity model with a set of response surfaces able to reproduce crash load-deflection curves using bi-linear laws. Implementing this approach in a global model of a helicopter structure could be advantageous in the training phase of the surrogate model employed.

Joosten et al. [37] validated with experiments a numerical design of a composite helicopter airframe and subfloor. In their paper the building block approach was introduced for numerical simulation and experimental validation depicted in Figure 2.14. It represents the level of complexity depending on the stage of the pyramid the analysis or experiment is conducted.

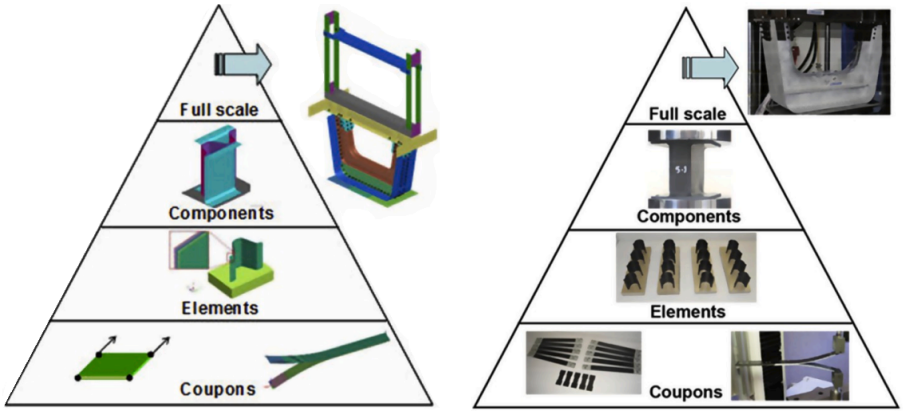


Figure 2.14: Building block approach for numerical models **(a)** and experimental validation **(b)** [37]

Prusty et al. [38] introduced a novel concept with variable crushing loading conditions to improve the energy absorption efficiency of crushing element, using pressurized composite tubes, found in helicopter subfloors. The motivation was to overcome the disadvantage of constant or fixed load energy absorbing systems. A simple analytical model provided preliminary results verifying the adopted variable loading concept. Quasistatic crash tests were conducted the results of which verified an explicit finite element study.

Anghilery et al. [39] on a study with AgustaWestland developed a numerical model of a helicopter subfloor. The research focused on enhancing the correlation between tests on individual intersections and entire four-walled subfloor cells. Critical factors for the model's robustness included using incremental damage materials, optimizing mesh size around joints, and determining friction and viscous friction coefficients. While complete subfloor cell results didn't perfectly match experimental data, the model accurately represented energy absorption, crash efficiency, and collapse mode evolution for intersections. These positive outcomes led to the model's application in simulating a complete test involving the subfloor cell, a helicopter seat, and an anthropomorphic crash test dummy.

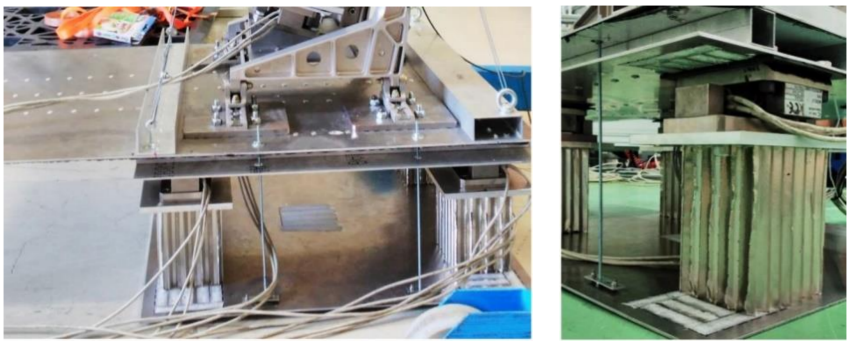
Subbaramaiah et al. [40] developed a retrofittable solution of energy-absorbing composite structures to enhance the crashworthiness performance of metallic helicopter subfloors. The study explores the potential of layered aluminum and glass fiber-reinforced composite hybrids for reinforcing subfloor components, employing LSDYNA and PAM-CRASH for explicit finite element analysis. Modeling strategies are discussed and the advantages of hybrid materials, by comparing mean crushing force and specific energy of absorption, are highlighted. These retrofits offer a promising way to boost crashwor-

thiness with minimal impact on structural weight and reduced complexity compared to other design solutions.

Zhou and Wang [12] studied the intersection element (see Figure 2.2) of a helicopter subfloor structure under crushing loading by conducting numerical simulations and experiments. This led to the redesign of the intersection element, integrating fold core sandwich structures as buffering and energy-absorbing components in helicopter subfloor structures. A comparative analysis revealed that the redesigned structure, with fold cores, exhibited reduced mass, lower peak force, and increased crush force compared to the conventional intersection element. This signified an enhancement in buffering and energy-absorbing capabilities.

In a subsequent research, Astori et al. [41] validated numerical models for a rotorcraft seat and subfloor equipped with energy-absorbing stages, focusing on vertical impact scenario for crashworthiness assessment, including an anthropomorphic dummy. The subfloor testing setup can be seen in Figure 2.15. The method developed began with a preliminary lumped mass model used to guide the design of the experimental drop-test. Additional static and dynamic tests were conducted at the coupon and sub-component levels to characterize the seat cushion, seat pan, and the honeycomb elements that were integrated into the structure for energy absorption. Then, a finite element model representing the full drop-test was created. The experimental test validated the original lumped mass model outlining the advantages of using both numerical techniques and experimental data for design assistance.

Al-Fatlawi et al. [42] aimed to optimize helicopter floors using fiber-reinforced plastic (FRP) composites to reduce weight and improve performance. They explored 46 FRP layer combinations while addressing nine design constraints, including deflection, facesheet stress, stiffness, buckling, coreshear stress, skin wrinkling, intracell buckling, and shear crimping. The fiber-reinforced plastic (FRP) honeycomb core sandwich plate optimized can be seen in Figure 2.16. Matlab's Interior Point Algorithm and Excel Solver's Generalized Reduced Gradient (GRG) Non-linear Algorithm were used as single-objective optimization tools.



(a) **(b)**

Figure 2.15: (a) Energy absorbers in helicopter structure and **(b)** load cells on the top of the subfloor crushable columns [41]

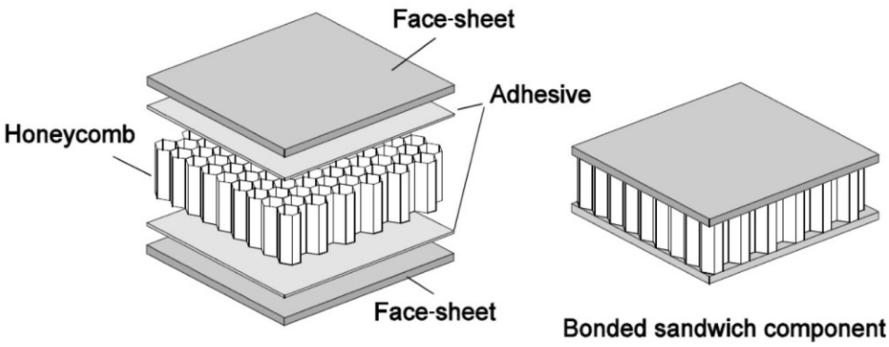


Figure 2.16: Sandwich structure of the helicopter subfloor [42]

A significant number of the aforementioned studies either aimed to optimize a specific component of the helicopter subfloor in terms of energy absorption and weight or to validate existed numerical models with experiments. Load-deflection curves along with energy absorbed-deformation curves are primarily used to evaluate the crashworthiness of the investigated concepts.

It is important to emphasize that relying solely on these studies does not ensure compliance with crashworthiness standards set by relevant authorities such as EASA and FAA. This is primarily due to the fact that the primary objective of these studies is to optimize specific components within the assembly, rather than comprehensively assessing the crashworthiness attributes of the entire helicopter structure. As a result, vital information concerning passenger loading, accelerations, or injury criteria may not always be provided. Nevertheless, it should be noted that despite these limitations, this step remains a highly significant aspect of crashworthiness evaluation.

Hajela et al. [24] addressed the subfloor crashworthiness problem quite effectively by employing Neural Networks as an approximation method in conjunction with Genetic Algorithms. However, it's worth noting that the models utilized for explicit analysis were considerably simplified, as depicted in Figure 2.8 (b). This is not always negative though, especially if the response of the structure can sufficiently be predicted.

A considerable number of papers [33, 35, 36] have been conducted to evaluate the crashworthiness of helicopters, utilizing surrogate modeling techniques to reduce computational costs. One of the main disadvantages of these models is that they often require extensive training to accurately represent high-fidelity models. Exploring the integration of low-fidelity data, which are less accurate and computationally less intensive to obtain, within surrogate modeling holds significant potential to advance the field of crashworthiness optimization in helicopters. Not only does it offer the possibility of improving model development, efficiency, and accuracy, but it also has the capability to significantly reduce training requirements.

2.2. Crashworthiness Evaluation Criteria

Fang et al. [43] summarized the energy-based metrics to asses crashworthiness and energy absorption. One of the most-common energy-based metric used in crashworthiness analysis [35, 44–50] is the energy absorption level expressed as:

$$EA(d) = \int_0^d F(s)\, ds \tag{2.2.1}$$

where $F(s)$ is the impact force, s is the crash distance and d is the total crash displacement considered. To consider mass efficiency, the specific energy absorption (SEA), defined as the EA per unit mass, has been used in literature [34, 51–55]. It is defined as:

$$SEA(d) = \frac{EA(d)}{M} \tag{2.2.2}$$

where M represents the mass of the structure. Crash force efficiency (CFE) is another criteria to design crashworthy structures. This metric is defined in Equation 2.2.4 as the ratio of the mean crash load (F_{avg}), given in Equation 2.2.3, to maximum load (F_{max}).

$$F_{avg}(d) = \frac{EA(d)}{d} \tag{2.2.3}$$

$$CFE(d) = \frac{F_{avg}(d)}{F_{max}(d)} \tag{2.2.4}$$

Mean crushing force was used by [33, 34] who focused specifically on helicopter subfloor crashworthiness.

The complex nature of crashworthiness problems arises from the intricate interplay of various structural phenomena involved in collisions, including high rates of deformation, contact forces, and other non-linearities. These complexities pose significant challenges in accurately modeling and predicting the behavior of structures under crush conditions. Compounding these challenges is the scarcity of experimental data necessary to develop precise predictive models. Considering these limitations, the need of surrogate models emerges. Surrogate models offer a representation of the complex crashworthiness phenomena, allowing for efficient exploration of the design space and optimization of structural configurations.

2.3. Surrogate Modelling in Crashworthiness

Surrogate modeling, also known as meta-modeling, is a technique used to create a representative model of a more complex model. It is often employed when the original

model is deemed impractical or difficult to use in a specific context. This may be due to factors such as the complexity of the model's implementation, the need for use in many-query applications such as optimization or real-time simulations, or the desire to conceal proprietary information or expertise.

To efficiently perform structural optimizations in crashworthiness problems, it is essential to establish a robust and flexible approximation system that requires a minimum number of exact points while considering large domains. Promising methodologies to achieve this goal include meta-model techniques such as Response Surface Methodologies (RSM).

The schematic process of implementing a Response Surface Methodology for meta-modeling optimization is depicted in Figure 2.17. The dimension of the dataset can be expressed as $n \times (p+k)$, where n is the number of observations, p is the independent variables, and k is the response variables number. Meta-modeling starts with goal definition, where the objectives, independent variables, response variables, and validation measures are clearly established. To reduce computational time, Design of Experiments (DoE) strategies are applied.

The subsequent stage is resource-intensive, involving the execution of experiments or simulations to gather response data for all design variables. Meta-models, aiming to approximate system behavior, are fitted, validated and analysed. Incorporating physics-based information into the construction of surrogate models presents a challenging but essential endeavor. The intricate nature of this challenge arises from the absence of a straightforward relationship between design variables and the corresponding structural responses, particularly in the context of crash conditions.

Nevertheless, the inclusion of low-fidelity information holds great promise as it has the potential to significantly reduce the training demands imposed on the meta-model. In the literature, artificial Neural Networks, radial basis functions, polynomial response surfaces, and kriging methods have been used in modeling helicopter subfloor for crashworthiness.

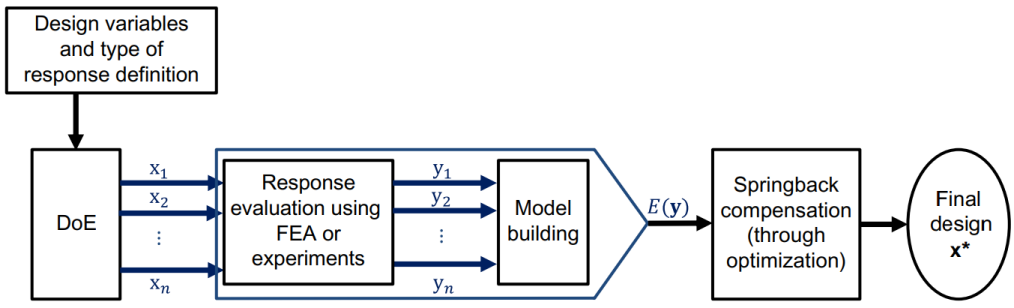


Figure 2.17: The schematic process of implementing a Response Surface Methodology for meta-modeling optimization [56]

2.3.1. Artificial Neural Networks

Neural Networks try to emulate the human brain's problem-solving approach. The human brain relies on intricate neural cells, neurons, to efficiently transmit electric-chemical signals. In Neural Networks, nodes or units process signals from input links and send activation levels to the next layer through output links, resembling the brain's data processing. Neural Networks have been used as an approximation tool in crashworthiness analyses since they are capable of reducing the computational costs significantly.

Hajela and Lee [24] highlighted that the Genetic Algorithm approach was computationally expensive, as each generation of evolution required a significant number of function evaluations, which was of the order of the population size. To make this approach effective, function approximation concepts were explored. Traditional Taylor series approximations would not be effective due to the absence of gradient evaluations. Instead, the multilayer perceptron (MPL) Neural Network was employed to construct a response surface of the DRI values based on training data making it akin to a special case of non-linear regression analysis.

The MPL network comprised input and output layers connected through hidden layers and, through training, learned to provide accurate estimates for input vectors in the training domain. A representation is depicted in Figure 2.18. The input layer, receives external inputs and transmits signals to the hidden layers. These signals undergo modifications, are weighted, and are redistributed throughout the hidden layers until they reach the output layer.

The output layer is designed in such a way that each neuron corresponds to an output component. A drawback of MLPs though is that, in general, they are not capable of extrapolating results beyond the design domain. Once trained, the network was used for generalization, producing approximate outputs for input patterns not part of the training set.

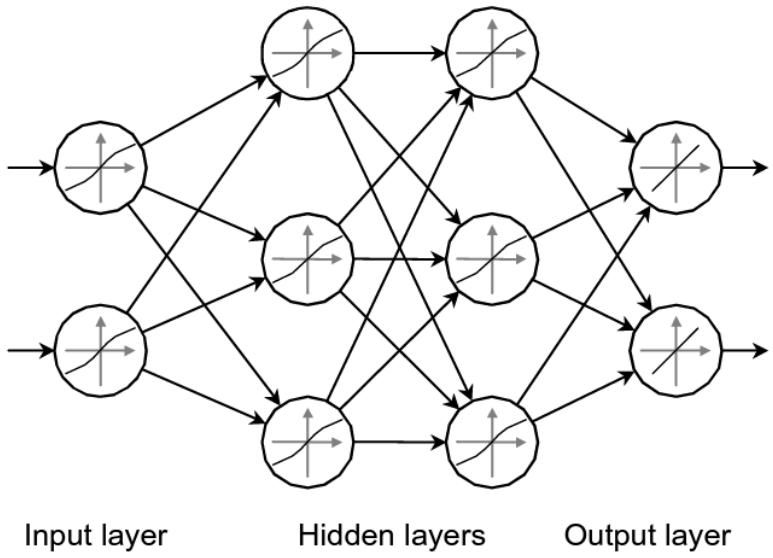


Figure 2.18: Architecture of a multilayer perceptron (MPL) [57]

This generalization operation was swift and significantly reduced the computational costs of repetitive analysis. To successfully implement the MPL network for function approximation, no important factors should be neglected from the input space, the number and distribution of training samples should be sufficient, and the training samples should be pre-processed to achieve computational efficiency.

Clustering techniques, which divided the training patterns into distinct clusters, were used to reduce the computational effort required in network training. Instead of a single network generated for the entire training set, multiple networks were trained with data belonging to specific clusters, allowing for more efficient generalization of new input patterns based on their cluster affiliation.

Bisagni et al. [19, 33] used multilayer perceptrons (MPL) Neural Networks to produce mean crash force, maximum crash force and load-time curves. The architecture of the adopted Neural Network in this study can be seen in Figure 2.19. The system involves four interconnected Neural Network subsystems: Box_1 receives normalized design variables and employs two Neural Networks (net1_a and net1_b) to return normalized maximum force. Box_2 processes input to return the normalized mean force value and uses three Neural Networks (net2_a, net2_b, and net2_c). Box_3 takes two normalized column vectors and handles the load-time curve using a Neural Network (Net3). Box_4 combines the results from Box_1, Box_2, and Box_3.

The system accurately reproduces crushing behavior, with small errors in force values when compared to finite element analysis, making it highly effective in simulating crash phenomena.

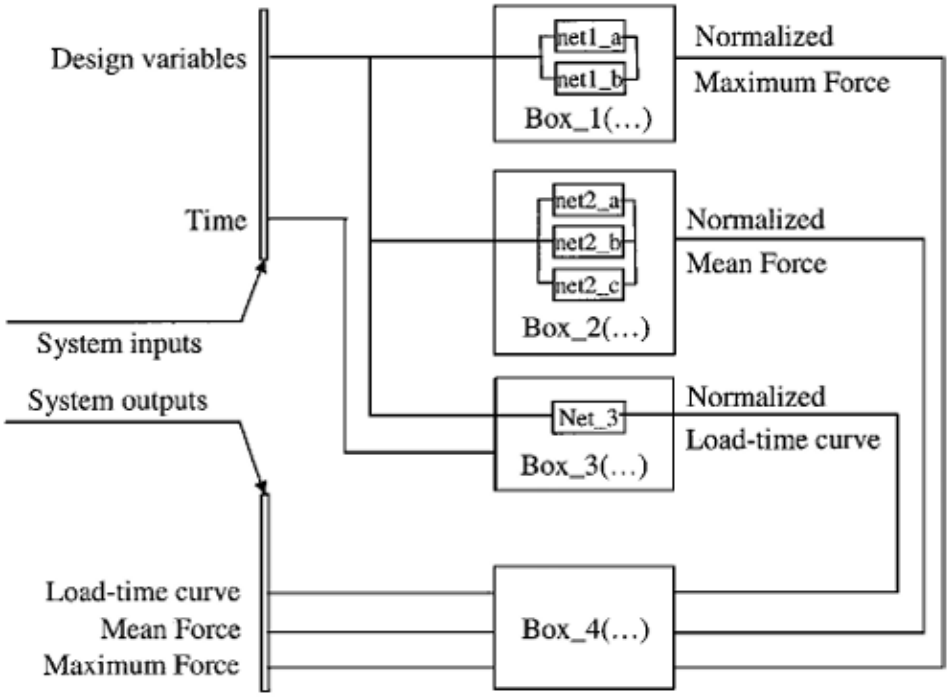


Figure 2.19: Adopted Neural Networks system in [33] to reproduce crash behavior

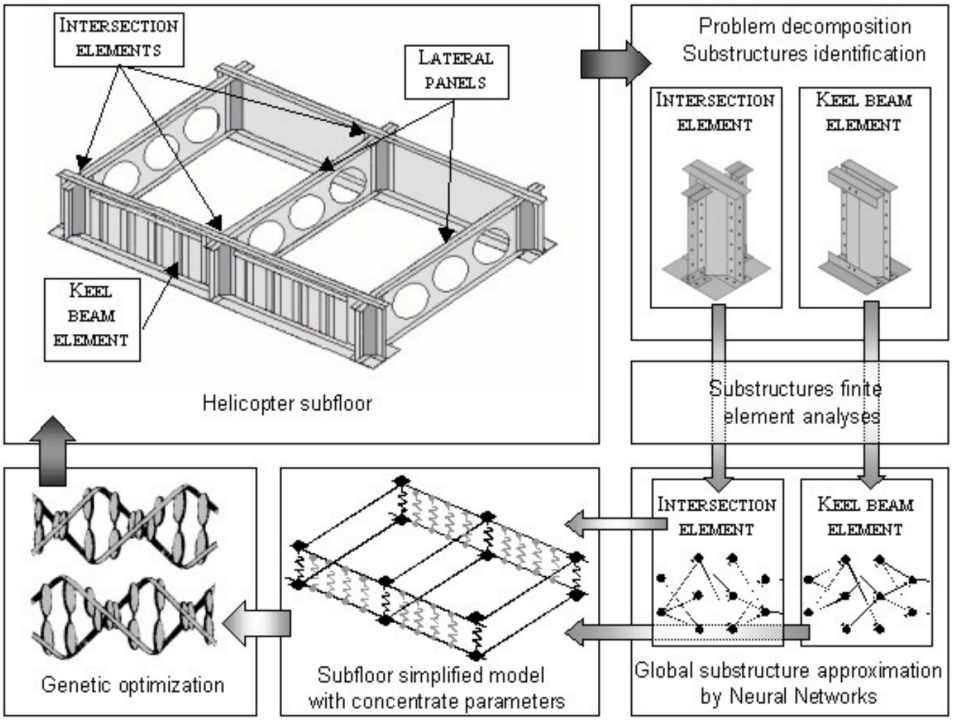


Figure 2.20: Optimization process implemented for a standard helicopter subfloor configuration [34]

Lanzi et al. [34, 57] used global approximation strategies based on Neural Networks able to provide indices regarding the structural behavior and complete load-time curves at crash events. The concept of parallel subsystems consisting of small Neural Networks was again adopted. A schematic overview of the optimization process implemented can be seen in Figure 2.20. This approach provides flexibility in adding, modifying, or removing individual outputs without the need for a complete system overhaul and retraining. Within MLP Neural Networks, the learning phase involves the adjustment of neuron connection weights, with the aim of minimizing the root mean square error between the known outputs in the training set and the returned outputs. Multi-objective optimization algorithms based on maximum distance concepts were employed taking into account equality and inequality constraints directly on the design variables. Explicit finite element analyses was used to obtain design points for the training process. The developed procedure was applied to riveted tubes, honeycomb structures, longitudinal keel beam and intersection elements of helicopter subfloors made of aluminium alloy.

2.3.2. Radial Basis Functions

Radial Basis Functions (RBF) method is an interpolating scheme used to describe the behavior of non-linear functions. It was first developed by Hardy [58] to address the interpolation of scattered multivariate data by employing a set of symmetric basis functions, each centered at individual sampling points. The formulation of radial basis functions is provided in Equation 2.3.1:

$$\hat{y}(\mathbf{x}) = \sum_{j=1}^{m} c_j p_j(\mathbf{x}) + \sum_{i=1}^{n_s} \lambda_i \phi(r(\mathbf{x}, \mathbf{x}_i)) \tag{2.3.1}$$

where m denotes the number of polynomial terms, c_j denotes the coefficient of the polynomial basis functions $p_j(\mathbf{x})$, n_s denotes the number of the sample points, λ_i denotes the weighted of the term of the i-th variable, $r(\mathbf{x}, \mathbf{x}_i)$ denotes the Euclidean distance, and $\phi(r)$ is the radial basis function.

Lanzi et al. [35] in order to overcome the computational challenges and evaluate the crash-load curves, replaced the finite element analyses with a system of response surfaces obtained using RBF. In this paper, linear, cubic and Gaussian RBF were used to produce reproduce load-deflection curves using bi-linear laws. To have a homogeneous allocation within the normalized domain the initial position of the sample points are modified using a hoc algorithm. Finite element analyses was used to obtain 30 sample points of which 20 were used to define the response surfaces and 10 to verify their accuracy level. The approximations obtained remained below the maximum error of 7 %. As a final step the 10 remaining sampling points were added to rebuild the response surfaces to incrase accuracy.

The same researchers in a subsequent study [36] estimated the crushing behavior with different response surfaces. They built three response surfaces using the RBF method, one for the first peak force, one for the ultimate force, and one for the absorbed energy. The error was estimated with Average Percentage Error (APE) and the Maximum Absolute Error (MAE) indices.

2.3.3. Polynomial Response Surfaces

Polynomial response surfaces have been extensively used in crashworthiness problems [48, 59–65] as approximation techniques. It is another simple yet effective way of creating surrogate modes. For instance, a quadratic Polynomial Response Surface (PRS) model, as described by Montgomery [66], can be represented as:

$$\hat{y}(\mathbf{x}) = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j>1}^{n} b_{ij} x_i x_j \tag{2.3.2}$$

where b_0, b_i, b_{ii} and b_{ij} denote unknown coefficients, n denotes the number of design variables, x_i denotes the design variable vector, $\hat{y}(\mathbf{x})$ denotes the approximation, and $y(\mathbf{x})$ denotes the value obtained from FEA. This meta-modeling technique exhibits limitations when applied to high-dimensional data or data characterized by oscillations. Improving accuracy is possible by increasing the order of the polynomials. However, in the case of high-order approximations, instability and inaccurate predictions may arise due to Runge's phenomenon [67].

2.3.4. Kriging Model

The Kriging model (KRG), was initially intended for use in mining and geostatistical applications that involve data exhibiting spatial and temporal correlations [68]. This model is comprised of two components: a global model denoted as $f(\mathbf{x})$ representing

the overall trend of the function of interest, and a local departure represented as $Z(\mathbf{x})$, which models the spatial correlation among data points through a stochastic process with a zero mean and variance σ^2. This can be mathematically expressed as:

$$y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}) \tag{2.3.3}$$

In Equation 2.3.3, $y(\mathbf{x})$ represents the unknown function of interest, $f(\mathbf{x})$ represents the global trend, and $Z(\mathbf{x})$ represents the correlation between data points, governed by a stochastic process with a mean of zero and variance σ^2. $Z(\mathbf{x})$ provides local variations, and the covariance between different points is expressed as:

$$\text{Cov}(Z(\mathbf{x}_i)), Z(\mathbf{x}_j))) = \sigma^2 \mathbf{R}[R(\mathbf{x}_i, \mathbf{x}_j)] \tag{2.3.4}$$

The correlation function $R(\mathbf{x}_i, \mathbf{x}_j)$ defines the correlation matrix $\mathbf{R}$ as follows:

$$R(\mathbf{x}_i, \mathbf{x}_j) = \exp\left[-\sum_{k=1}^{n} \theta_k |x_i^k - x_j^k|^2\right] \tag{2.3.5}$$

where θ_k denotes the correlation fitting parameter, and x_i^k and x_j^k denote the k-th components of the $\mathbf{x}_i$ and $\mathbf{x}_j$ sample points respectively.

Gao et al. [69] developed a time-space Kriging-based sequential meta-modeling approach that was proposed for Multi-objective Crashworthiness Optimization (MOCO). The approach utilized novel time-space design criteria to construct meta-models for the optimization objectives, which included mechanical responses with respect to both structural space domain and crash time domain. Adaptive addition of new samples was employed to improve the approximation accuracy during optimization, with the guidance of an adaptive weighted sum method. The effectiveness of the proposed method was demonstrated through investigation of a multi-cell thin-walled crashworthiness design problem. Kriging methods were used by Yang et al. [70] in multidisciplinary optimization of crashworthy vehicle structures while other examples of studies that used Kriging as an approximation method in computationally intensive problems for crashworthy designs are [62, 71].

Besides, the Anisotropic Kriging surrogate model utilizes statistical theory to construct a surrogate model. It is particularly applicable in the numerical simulations where random geometric errors are introduced to enhance the accuracy in representing crashworthiness. Since the geometric variables have different levels of importance, Anisotropic Kriging is able to effectively handle this variability. The suitability of this surrogate model does not depend on the presence of random errors. Its underlying technology enables control over the significance of variables, making it particularly suitable for crashworthiness scenarios [72].

2.4. Optimization for Structural Crashworthiness

Non-linear structural optimization can be very challenging especially for problems that require explicit solvers like crashworthiness problems. Many commercial codes (NASTRAN, LS-DYNA etc) are capable of performing size, shape, and topology optimization in acceptable amount of time while the computational cost can be further decreased when the (semi-) analytical sensitivity analysis allows the deployment of gradient-based optimizers. However, gradient information is not always available making the optimization task more intricate.

The dominant factors contributing to non-linearities in crash problems are contact forces, as well as material and geometric non-linearities. Contact forces represent a non-linearity in boundary conditions that can vary in magnitude and location from one time step to another, posing a significant challenge for applying the sensitivity analysis mentioned above. To overcome the complexities associated with sensitivity analysis in crash problems, meta-model-based techniques described in the previous section are frequently employed. Structural optimization techniques for crashworthiness are more complex due to the presence of multiple local optima in the design space, making the quest for the global optimum challenging. Especially deterministic algorithms, even though they have proven to be efficient, can easily be trapped in local minima. Non-linear optimization techniques can be divided into two subcategories: non-linear local and non-linear global optimization techniques.

The first category tend to converge to local optima, rendering them suitable for refining solutions in the proximity of initial guesses. The later category is designed to search for the global optimum across a broader problem space. A classification of the optimization techniques is presented in Figure 2.21.

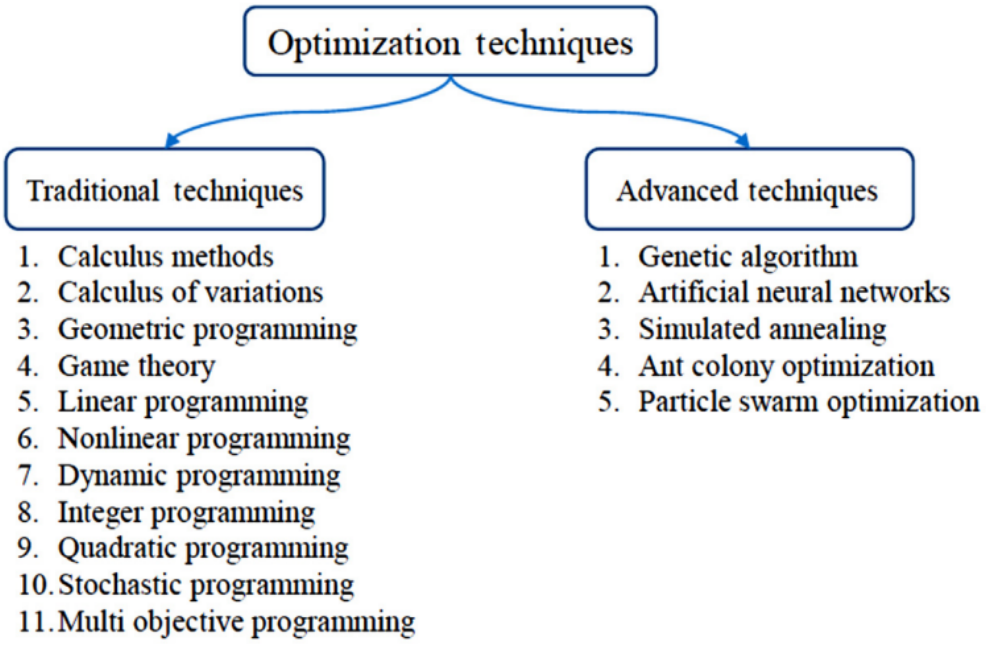


Figure 2.21: Classification of optimization techniques [73]

2.4.1. Genetic Algorithms

Genetic Algorithms can be aptly categorized as stochastic search methods based on Darwin's theory of survival of the fittest. The genetic search optimization does not require the use of gradients from objective or constraint functions, rendering it suitable for scenarios where the design space could exhibit non-convexity or discontinuity common in crashworthiness problems.

Hajela and Lee [24] used Genetic Algorithms together with a crash response analysis code. Their approach involved both sizing and topology optimization while two main concerns were addressed. Firstly, the design space comprises both discrete and continuous variables making conventional gradient-based optimization techniques computationally expensive. Secondly, the problem's design space was shown to be non-convex, requiring the use of a global search strategy to find the global optimum. The weighting coefficients of a linear combination of load-deflection curves were designated as design variables along with the linear stiffness and the linear displacement. Their aim was to minimize the accelerations of the occupants focusing on the design of the subfloor structure. This was mathematically stated as:

$$\begin{aligned} &\text{Minimize} && \bar{F} \\ &\text{subject to} && \bar{X}^L \leq \bar{X} \leq \bar{X}^U \end{aligned} \tag{2.4.1}$$

where $\bar{F}$ represents a scalar vector of the dynamic response index (DRI) values, indicative of the likelihood of severe spinal injury to the occupants; $\bar{F}$ is a function of the design variable vector represented by $\bar{X}$; $\bar{X}^L$ and $\bar{X}^U$ are the lower and upper limits respectively. The variable vector included topological variables indicating the presence or absence of energy-absorbing elements and parameters characterizing load-deflection curves. The topological variables were binary (0 or 1), and the load-deflection parameters involved linear stiffness, displacement, and six weighting coefficients used for the linear superposition of the basis functions. Typical values for the linear stiffness and displacement were determined with numerical experimentation conservatively assuming that the subfloor structure absorbs all kinetic energy from a vehicle's vertical impact. These values yielded an equivalent energy absorption on average and were used as reference points to create feasible and infeasible designs by varying parameters above and below the mean.

Bisagni et al. [19] addressed the optimization of subfloor configurations for crashworthiness, considering 23 design variables such as positions of intersection elements, thicknesses, angle of elements, number of rivets and widths. They treated panel thicknesses as continuous variables, optimizing them and then rounding to the nearest commercially available thickness. In contrast, the number of vertical rivets and longitudinal keel beam elements was treated as discrete variables. Acceleration constraints were considered in order to remain within the human tolerance limits while the the mass specific energy absorbed by the subfloor was chosen as objective function. The use of penalty functions helped in transforming the optimization problem into a unconstrained maximization

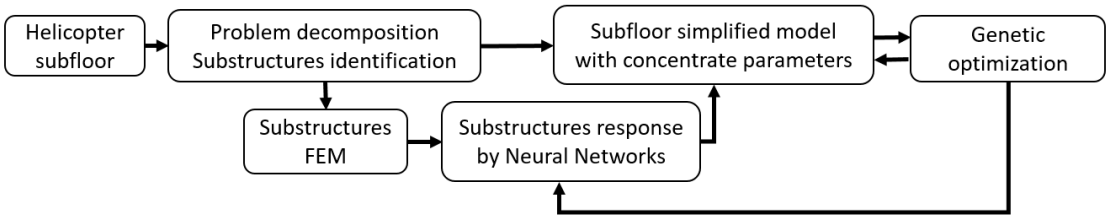


Figure 2.22: Optimization scheme of helicopter subfloor (adapted) [19]

problem. The optimization procedure followed is illustrated in Figure 2.22. It begins with problem decomposition and the identification of substructures. Size and global variables are separated and a simplified subfloor model defines and validates the substructure interactions. Subsequently, response surfaces for each substructure are generated using Neural Network systems trained with a minimal set of finite element analyses. These response surfaces were able to reproduce the values of the constraints and the objective functions in every optimization iteration. Finally, optimization runs were executed using Genetic Algorithms to obtain an optimal solution.

Bisagni et al. [33] utilized Genetic Algorithms within MATLAB, employing an objective function aimed at describing both crushing force efficiency and increased mass-specific absorbed energy. This implementation involved basic parameters such as selection, crossover, and mutation to optimize the process. Despite its simplicity, the algorithm's performance appeared to be suitable for addressing the defined crashworthiness problem. They incorporated penalty functions into the objective function which was considered as the sum of the crushing force efficiency and the specific absorbed energy. The expression defining the fitness function can be seen in Equation 2.4.2. As in their previous study, the design variables considered were the thickness of the webs, the thickness and the position of the angle elements, and the number of of vertical rivets (see Figure 2.2).

$$\boldsymbol{f}(\boldsymbol{x}) = \upsilon_1\upsilon_2\upsilon_3 \left\{ \left(\frac{F_{\text{mean}}(\boldsymbol{x})}{F_{\text{max}}(\boldsymbol{x})} \right) + K \left(\frac{F_{\text{mean}}(\boldsymbol{x})}{\text{mass}(\boldsymbol{x})} \right) \right\} \tag{2.4.2}$$

$$F_{mean} \geq \bar{F}_m \quad F_{max} \leq \bar{F}_M \tag{2.4.3}$$

where $\boldsymbol{f}(\boldsymbol{x})$ represents the objective function, $\boldsymbol{x}$ represents the vector of the normalized design variable, F_{mean} represents the mean force, F_{max} represents the maximum force, $\bar{F}_M$ represents upper constraint on the maximum force, and $\bar{F}_m$ represents lower constraint on the mean force. The maximum force is described by υ_1, the mean force is described by υ_2, and the vertical rivet number is described by υ_3. K is a weighting parameter which Bisagni et al. variated along with the constraint values.

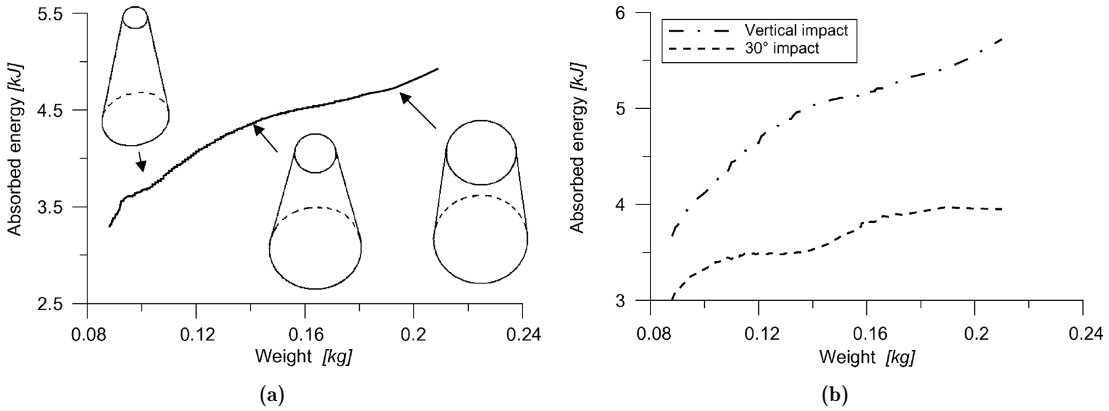


Figure 2.23: Pareto curves of energy absorption VS weight during vertical a 30° impact conditions [35]

Lanzi et al. [35] suggested an multi-objective optimization of a composite energy absorber under crashworthiness requirements. It was suggested that the simplest approach of the optimization problem would be a single objective formulation to minimize the weight while ensuring crashworthiness performance through the incorporation of nonlinear inequality constraints on absorbed energy. Other constraints such as limitations on the maximum load or the acceleration peaks could be also imposed. However, this approach would result in a single optimal solution for each set of constraints and is not recommended for preliminary design. The optimization problem can be generalized by using different objective functions, such as minimizing weight and maximizing absorbed energy under various impact conditions.

A multi-objective Genetic Algorithm is used to carry out the searching for the Pareto set and a depiction of some of the Pareto curves obtained in this study can be seen in Figure 2.23. Three independent design variables were specified for the energy absorber: the major axes of the lower edge, the eccentricity of the lower edge, and the taper ratio, which is defined as the ratio between the upper edge radius and the lower edge minor axis. Decision criteria were employed not only to find the Pareto set but also selecting a single feasible solution. Weighted exponential penalty functions were used to include constraints, while minimization problems investigated are formulated by using the opposite of the original objective function as the fitness function. Furthermore, a ranking selection method based on non-dominated points is integrated into the initial single-objective Genetic Algorithm (GA).

Lanzi et al. [34, 57] used a sequential multi-objective optimization method based on the "goal attainment" approach. It was applied to a typical helicopter subfloor through the process seen in Figure 2.20. The selected objective function focuses on the energy absorbed per unit mass by the subfloor, while constraints are implemented regarding acceleration. The design variables encompass both the position of individual elements within the subfloor and their respective local characteristics. The aim was to obtain a set of points where the distances between each point and all other points exceed a specified minimum goal distance, denoted as d_k. If a solution exists, the value of d_k increases. At each $k-$th iteration, the goal attainment method is applied to identify an appropriate arrangement of points starting from the final positions in the prior iteration, according

to the following formulation:

$$\underset{\boldsymbol{\lambda},x}{\text{minimize}}\ \boldsymbol{\lambda} \tag{2.4.4}$$

$$\text{subjected to} \begin{cases} d_1 + \boldsymbol{\lambda} \geq d_k \\ \vdots \qquad\qquad i = 1, \ldots, m \\ d_i + \boldsymbol{\lambda} \geq d_k \end{cases} \tag{2.4.5}$$

where x denotes the design variable vector; n denotes the total number of points to arrange; N denotes the dimension of the design domain; d_i denotes the distances between the points; and x is a vector that contains the positions of each point and comprises a total of $n \times N$ elements. To expedite the optimization process, gradients for distances and objective functions are computed analytically.

Astori and Impari [74] investigated the interaction of a two-stage energy-absorbing system, made of seat and subfloor crushing elements to minimize occupant lumbar spine load during crash landings. The variable parameters chosen for each energy-absorbing stage, encompass the activation load and the plastic stiffness associated with it. A fully multi-body model was developed as depicted in Figure 2.24.

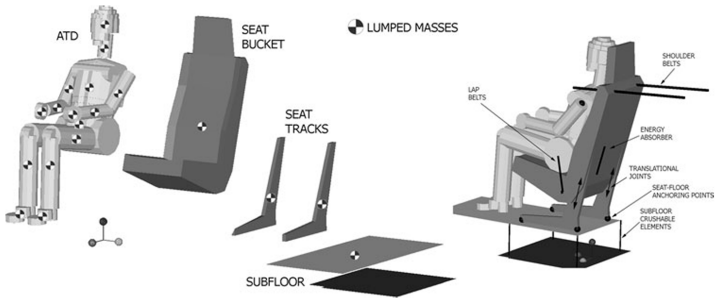
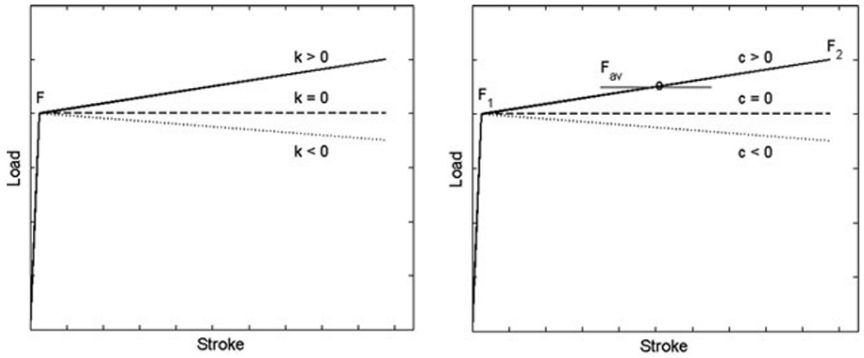


Figure 2.24: Multi-body model accounting for the seat, occupant and subfloor [74]

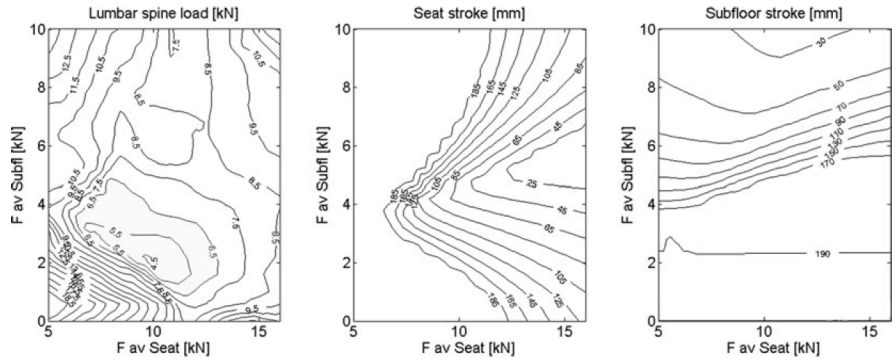
They performed single objective optimizations with iChrome Nexus^{TM} Genetic Algorithm where two cases were considered. The first optimization attempt evaluated ideal elastic-plastic laws for both the seat energy absorber and the subfloor elements. General elastic-plastic laws of deformation used can be seen in Figure 2.25 where F represents the activation load, k is the stiffness of the plastic region, and F_{av} is the average load in the plastic region. In this approach, the coefficients c_{seat} and c_{subfl} were set to zero. The second optimization attempt considered more general elastic-plastic laws characterized by linearly variable plastic loads. This approach led to the determination of specific values for the mechanical parameters of the seat and subfloor, optimized to minimize the maximum lumbar spine load. A sensitivity study generated response surfaces of the lumbar spine load as a function of loads applied to the seat and subfloor. These diagrams can be seen in Figure 2.26 where the considered safe area with spine load

less than 6.5 kN is shadowed. The multi-body approach used allowed to complete the optimization in under 3 hours on a dual-CPU personal computer. On the other hand, a full finite element approach, although more refined, was computationally expensive and posed challenges in system parameterization. Therefore, it was best suited for final verification or optimization purposes. A hybrid approach, combining both finite element and multi-body analysis, achieves a balance between accuracy and CPU cost.

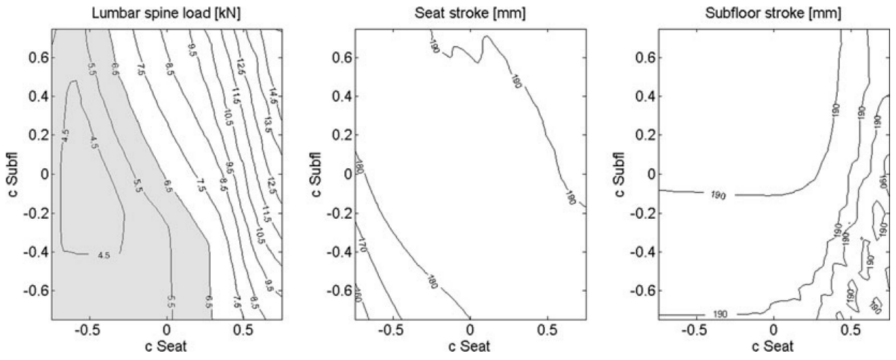


(a) **(b)**

Figure 2.25: General elastic-plastic laws of deformation used for subfloor elements as a function of plastic stiffness k (a) and coefficient c (b) [74]



(a) as functions of average seat energy absorber activation load ($F_{av\ Seat}$) and average subfloor crushing load ($F_{av\ Subfl}$)



(b) as functions of c_{Seat} and c_{subfl}

Figure 2.26: Response surfaces of the max lumbar spine load, seat stroke and subfloor deflection [74]

2.4.2. Sequential Quadratic Programming Methods

Bisagni et al. [33] stated the optimization problem as in Equation 2.4.2. They approached the problem by decomposing it into simpler sub-problems that could be solved iteratively. The idea was to formulate a quadratic sub-problem by means of a quadratic approximation of the Lagrangian function. The non-linear constraints were linearized and bound constraints were expressed as inequality constraints.

The implementation of the Sequential Quadratic Programming (SQP) algorithm comprised three steps. Firstly, the Hessian matrix of the Lagrangian function was updated through the use of a positive definite quasi-Newton approximation, such as the Broyden-Fletcher-Goldfarb-and-Shanno (BFGS) methods. Subsequently, the quadratic programming problem was solved, with the sub-problem obtained by linearizing non-linear constraints and employing an active set strategy. This solution procedure encompassed two phases, involving the determination of a feasible point and the generation of a convergent sequence of points. The third step of the past implementation involved conducting a line search and calculating the merit function.

The SQP algorithm was implemented in MATLAB while it was penalized when integer design variables were incorporated and when dealing with irregular or opposing behavior in mean and maximum forces. This renders the SQP method susceptible to finding local minima instead of the global minimum. As with many other gradient-based optimization methods, the optimal design might be influenced by the initial design.

SQP was also used by Lanzi et al. [75]. They formulated the optimization problem as:

$$minimize \ \mathbf{max}(w_1 \cdot S_{Seat}(t) + w_2 \cdot S_{Subfloor}(t)) \tag{2.4.6}$$

$$\text{subjected to} \begin{cases} \max(F_{Lumbar}(t)) \leq \bar{F}_{Lumbar} \\ \max(S_{Seat}(t)) \leq 0.95 \cdot S_{Seat}^{Max} \\ \max(S_{Subfloor}(t)) \leq 0.95 \cdot S_{Subfloor}^{Max} \end{cases} \tag{2.4.7}$$

where S_{Seat} represent the seat stroke, $S_{Subfloor}$ represents the subfloor stroke, F_{Lumbar} represents the actual maximum lumbar load carried by the dummy model, $\bar{F}_{Lumbar}$ represents the maximum allowable value, $S_{Subfloor}$ represents the maximum stroke of the absorption elements on the subfloor, $S_{Subfloor}^{Max}$ represents the maximum stroke of the absorption elements on the seats, and t represents the time. The weighting factors are represented by w_1 and w_2, while the factor of 0.95 was used so that the optimization algorithm controls and avoids bottoming.

Lanzi et al. [36] used standard Quasi-Newton method to conduct the optimization explorations. The optimization domain depended on four different design variables that define the shape of the intersection element of the helicopter subfloor. The parameteriza-

tion of the shape and the meticulous selection of sample points within the optimization domain (Design of Experiment) led to a highly effective problem formulation.

2.4.3. Interior-point Method

The Interior-point method was used by Al-Fatlawi et al. [42] to solve a single objective optimization subjected to nine constraints using MATLAB. Grippo and Sciandrone [76] provide a detailed documentation of this method. Considering a mathematical programming problem of the form:

$$\begin{aligned} &\min f(x) \\ &x \in H, \quad g(x) \leq 0 \end{aligned} \tag{2.4.8}$$

where $H \subseteq \mathbb{R}^n$ is a closed set, and is assumed that the functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are at least continuous. The feasible set is denoted by $\mathcal{F} = \{x \in H : g(x) \leq 0\}$, and the set where the inequality constraints $g(x) \leq 0$ are satisfied as strict inequalities is denoted by $\mathcal{D} = \{x \in \mathbb{R}^n : g(x) < 0\}$.

In general, we refer to interior point methods with reference to approaches such that a solution of the constrained problem Equation 2.4.8 is approximated either with a sequence of points in $\mathcal{D}$ (infeasible methods) or a sequence of points in $H \cap \mathcal{D}$ (feasible methods). In both instances, the principle an interior point method is to remain in $\mathcal{D}$, strictly satisfying the inequality constraints $g(x) \leq 0$. Feasible methods for inequality-constrained problems are considered methods where $H = \mathbb{R}^n$, $\mathcal{D}$ is non-empty, and a point $x_0 \in \mathcal{D}$ is available. In this case, the motivation for interior point methods is essentially to employ solution techniques based on unconstrained minimization algorithms directly, without introducing penalty terms on the constraints.

2.4.4. Generalized Reduced Gradient Method

In 1967, Wolfe introduced the reduced gradient method, which was based on a straightforward variable elimination approach designed for equality-constrained problems [77]. The Generalized Reduced Gradient (GRG) method extends the principles of the reduced gradient method to handle non-linear inequality constraints. In this approach, a search direction is determined to ensure that the currently active constraints remain precisely active even with small adjustments. If, due to the non-linearity of constraint functions, some active constraints are not precisely met, the method resorts to the Newton–Raphson technique to return to the constraint boundary. As a result, the GRG method bears certain similarities to the gradient projection method. This method was used by Al-Fatlawi et al. [42] along with the Interior-point method to perform the single objective optimization of the helicopter subfloor.

2.4.5. Bayesian Optimization

Bayesian optimization has been efficiently employed to address crashworthiness optimization problems in the automotive industry [78–82]. This optimization technique usually initiates with the formulation of the design problem, the development of a baseline model, and parameterization. Data-driven prior knowledge derived from critical components in-

forms the design problem. Subsequently, an initial design dataset is generated by the design of experiments (DoE) and assessed via Finite Element (FE) simulations to train the surrogate model. The optimal acquisition process determines the best solution, with the surrogate model training and optimal acquisition conducted sequentially due to their computational efficiency. In parallel, The new optimized response is evaluated through FE simulations in parallel to reduce the computational cost and is then incorporated into the design dataset for updating the surrogate model in the subsequent iteration of optimal acquisition. This iterative process continues until the termination criterion is satisfied.

2.5. Research Questions

The aforementioned studies primarily focused on two main objectives: optimizing specific components of helicopter subfloors for energy absorption and weight reduction, and validating existing numerical models through experimental testing.

To evaluate the crashworthiness performance of the investigated concepts, load-deflection curves and energy absorption-deformation curves were predominantly employed. Specifically, they provide valuable insights into how the components responded under various loading conditions and offer a means to assess the effectiveness of structures in absorbing energy during crashes.

While Machine Learning techniques offer a promising avenue for addressing these challenges, there exists a notable dearth of comprehensive studies that focus on accurately predicting load-displacement curves and energy absorption, particularly in scenarios where numerical or experimental data are scarce.

To bridge this gap, several key research questions emerge. Firstly, **how can Machine Learning techniques be effectively utilized to confront the complexities inherent in crashworthiness assessment of helicopter subfloor structures and components?** Understanding the comparative performance of different predictive approaches and algorithms constitutes a critical task for this inquiry.

Moreover, delving into the core challenges and mechanisms involved in predicting the structural behavior of tubular metallic structures under crushing conditions becomes imperative as the methodologies developed will be implemented in a crushing cylinder structure. Machine Learning can offer valuable understanding into these intricacies and potentially surmount the obstacles posed by non-linear responses, material failures, and other non-linear phenomena.

In the context of crashworthiness evaluation, a comprehensive examination of the energy-absorbing subfloor structure at the component level is essential. The construction of an accurate surrogate model, representative of the structure, demands careful consideration and thorough analysis to ensure efficiency with minimal training. Additional research questions arise here: **What advantages does a surrogate model gain by predicting the load-displacement curve as a function compared to directly predicting energy absorption as a scalar value? How could the surrogate leverage more in terms of accuracy**

and efficiency?

The study aims to explore methodologies capable of augmenting the prediction performance of energy absorption for tubular metallic structures, particularly in scenarios characterized by limited numerical or experimental data availability. A potential approach would be to introduce low-fidelity models along with the high-fidelity models. Another research question arises here. **How does the introduction of low-fidelity data alongside high-fidelity data impact prediction accuracy, and what implications does this have for predicting energy absorption in more intricate structures like helicopter subfloors?** Additional questions related to this is **how accurate the low-fidelity model will be, how will this data be incorporated in the methodology, and how will the low-fidelity data influence the overall accuracy of the predictions?**

By addressing these research questions, the study endeavors to contribute substantially to the existing body of literature, offering insights into the application of Machine Learning techniques for crashworthiness assessment and proposing innovative methodologies to improve prediction accuracy, even in data-scarce environments.

3 Thin-walled Metallic Tubular Model

For the development of the methodology, the entire helicopter subfloor is replaced by a tubular model to render it computationally feasible. The numerical model of thin-walled tubular structures that is considered as high-fidelity model in the present thesis, is presented in Section 3.1. It encompasses four cross-sectional shapes and two materials for studying the crushing behavior. The cross-sectional shapes consist of square, circular, hexagonal, and octagonal configurations. Crushing modes include either extensional or in-extensional, as depicted in Figure 3.1. The materials involved are AA6060-T4 and AISI-316. An analytical model is presented in Section 3.2 where a closed form relationship for the mean crushing force is provided. This closed-form relationship constitutes an additional font of information which, if efficiently processed, might augment the accuracy of predicting the energy absorption.

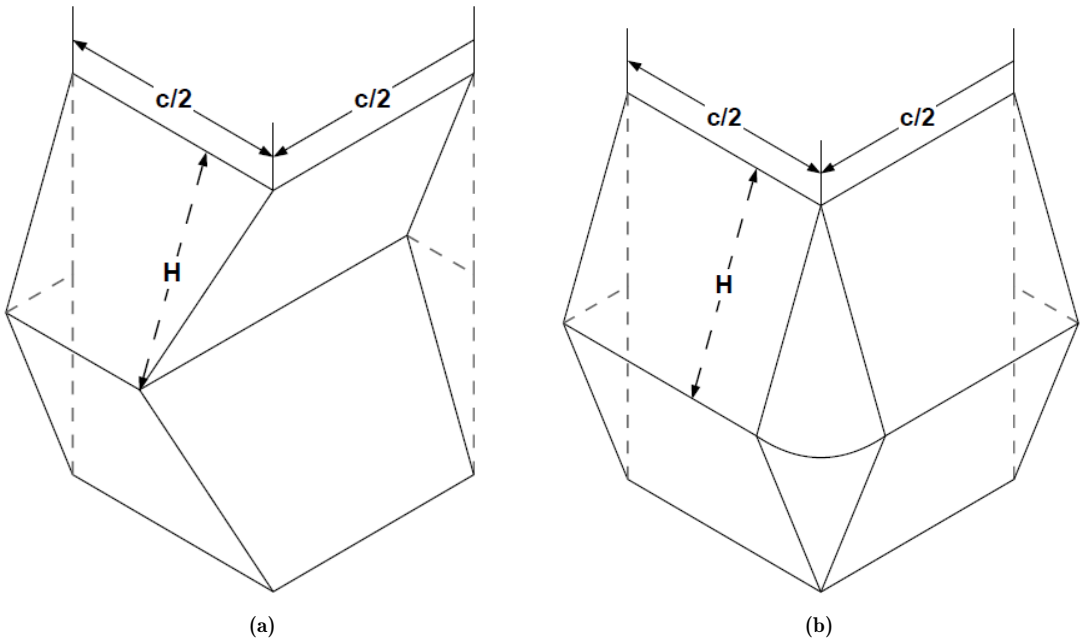


Figure 3.1: Extensional crushing mode **(a)** and inextensional crushing mode **(b)** for metallic tubular structures [83]

Each sample obtained by the crushing tube consist of a load-displacement curve and is associated with geometrical parameters of the respective tube sample such as material, cross-sectional shape, side length or radius, wall thickness, and tube length. It is shown that there is a good agreement with the analytical, the numerical, and experimental data [83].

3.1. Description of the Numerical Model

Shell-based crushing models are commonly used in simulating the crushing behavior of thin-walled metallic structures, and their accuracy has been well-documented in the

literature [84–86]. The data used were generated by a shell-based finite element model developed in Abaqus, employing quadrilateral shell elements with reduced integration (S4R) [83]. All models were comprised of a tubular structure positioned between two rigid planes. A vertical displacement is applied to the top rigid plane, while the bottom plane is encastred, and the nodes at the bottom of the tubular structure were connected to the bottom rigid plane.

In the inextensional crushing mode, a type-I trigger is introduced on alternate faces, while in the extensional crushing mode, a type-II trigger is introduced on all faces. A trigger mechanism is employed to initiate a progressive folding sequence within the structure by creating local weakness. This ensures the formation of a stable crushing zone and prevents other failure modes such as global bending. The use of trigger mechanisms also leads to a decrease in peak crushing force which is beneficial for crash applications due to the susceptibility of the human body to severe injury when exposed to high deceleration magnitudes [87].

The load-displacement curves, illustrated in Figure 3.4, are obtained by positioning a reference node at the center of the top rigid plane where both the reaction force and the displacement are measured. All the samples used in the present dissertation are documented in Appendix B. A representation of the finite element shell-based model used for hexagonal tubular structures is depicted in Figure 3.2.

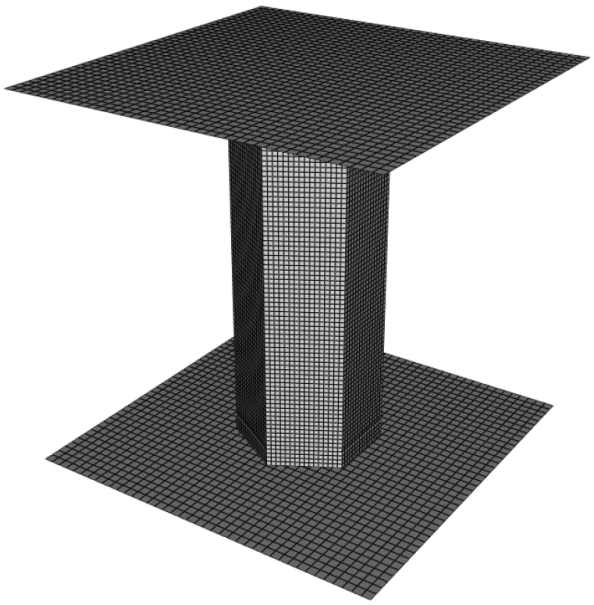


Figure 3.2: Tubular structure shell-based numerical model in Abaqus [83]

The numerical model underwent validation by comparing simulated data with experimental results. Material data from published stress-strain curves were used to ensure consistency across datasets. The close alignment between the numerical simulations and experimental data points, with a coefficient of correlation R^2 of 0.97, demonstrates the model's robustness in simulating the axial crushing behavior of metallic tubular structures [83].

An example of the type of curves aimed to be predicted in the present study is shown in Figure 3.3. It depicts a finite element result for a square cross-section with material AA6060-T4. The side length is 30 mm, thickness is 1.75 mm, and tube length is 200 mm.

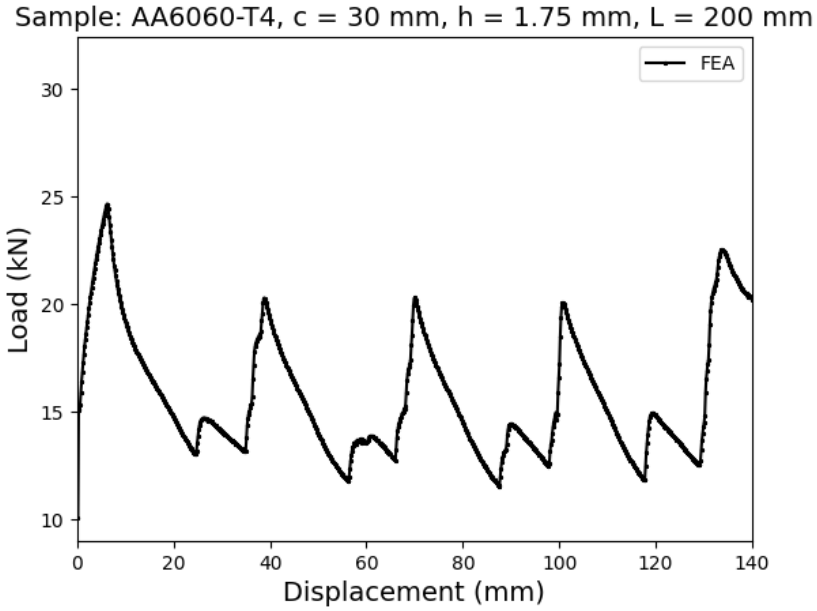


Figure 3.3: FEA load-displacement curve for square cross-section sample [83]

3.2. Closed-form Expression of Mean Crushing Load

The purpose of this section is to present the equations that lead to a generalized mean crushing load expression applicable for axial crushing of metallic tubular structures. In this section, M_0 represents the fully plastic bending moment per unit length, σ_0 the flow stress, ε the strain, ε_u the ultimate strain, N_c the number of corners for a polygonal cross-section [85], P_{mean} the mean crushing load, h the metallic structure wall thickness, κ the effective crushing length, c the edge length of the cross-section in polygonal tubular cases, $2H$ the initial distance between plastic hinges at top and bottom of a basic folding element [88] and R^2 the coefficient of determination. An ideal load-displacement curve (continuous line) and the mean value (dash line) for a tabular structure is illustrated in Figure 3.4 .

The following models in the literature exhibit satisfactory correlation with the experimental and numerical mean crushing force data. First Tabacu et al. [85] with an equation valid for circular structures:

$$\frac{P_m}{M_0} \cdot \kappa = 33.85 \cdot N_c^0 \cdot \left(\frac{R}{h}\right)^{0.5} \tag{3.2.1}$$

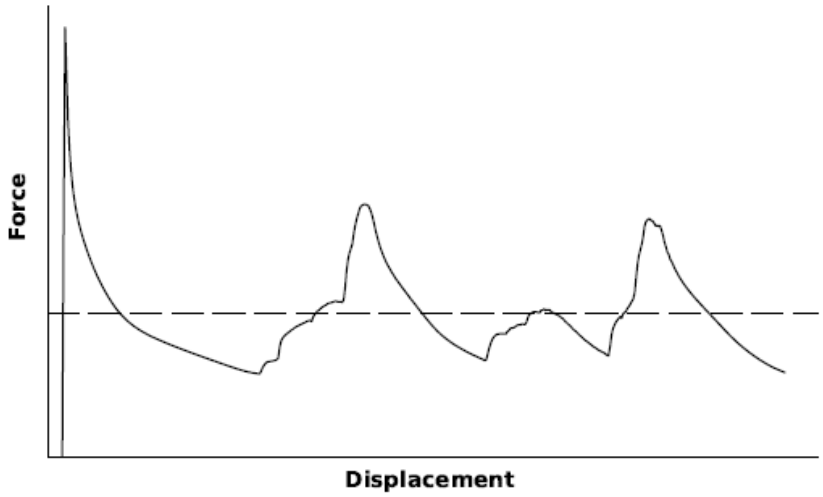


Figure 3.4: Ideal load-displacement curve for a crushing tube and mean value [83] (adapted)

The N_c variable is kept for similarity purposes in the previous equation. Tabacu et al. proposed also a closed-form relationship applicable to the extensional crushing of polygonal tubular structures:

$$\frac{P_m}{M_0} \cdot \kappa = 12.56 \cdot N_c^{0.5} \cdot \left(\frac{c}{h}\right)^{0.5} \tag{3.2.2}$$

The flow stress denotes the critical stress level required for initiating plastic deformation within the material. The determination of flow stress directly impacts the computation of the mean crushing force exerted on these metallic structures. A formula [85] to calculate the flow stress is :

$$\sigma_0 = \frac{1}{\varepsilon_u} \int_0^{\varepsilon_u} \sigma \cdot d\varepsilon \tag{3.2.3}$$

The flow stress and the fully plastic bending moment connect with the following relationship [89] for circular tubular structures:

$$M_0 = \frac{\sigma_0 h^2}{4} \tag{3.2.4}$$

The model proposed by Zhang and Zhang [90] for the inextensional crushing of polygonal tubular structures also maintains a consistent level of agreement between the experimental and numerical data:

$$\frac{P_m}{M_0} \cdot \kappa = 15.91 \cdot N_c^{1.03} \cdot \left(\frac{c}{h}\right)^{0.15} \tag{3.2.5}$$

Shreyas et al. [83] combined Equation 3.2.1, Equation 3.2.2 and Equation 3.2.5 into one general equation for mean crushing force, applicable to metallic tubular structures:

$$\frac{P_m}{M_0} \cdot \kappa = X \cdot N_c^Y \cdot \left(\frac{c}{h}\right)^Z \tag{3.2.6}$$

where c/h is replaced with R/h for circular tubular structures. The values of X, Y and Z vary for cross-sectional shape and crushing mode (Ext stands for extensional and Inext stands for inextensional) and can be found in Table 3.1. Square cross-section tubes undergo inextensile crushing, indicating they fall under the Polygonal (Inext) case.

Table 3.1: Values for exponents of Equation 3.2.6 for each configuration

#	Cross-section	X	Y	Z
1	Circular	33.85	0	0.5
2	Polygonal (Ext)	12.56	0.5	0.5
3	Polygonal (Inext)	15.91	1.03	0.15

A good correlation between the analytical average crushing load and the mean crushing dataset is achieved encompassing all cross-sections and both modes of crushing ($R^2 = 0.97$), as illustrated in Figure 3.5.

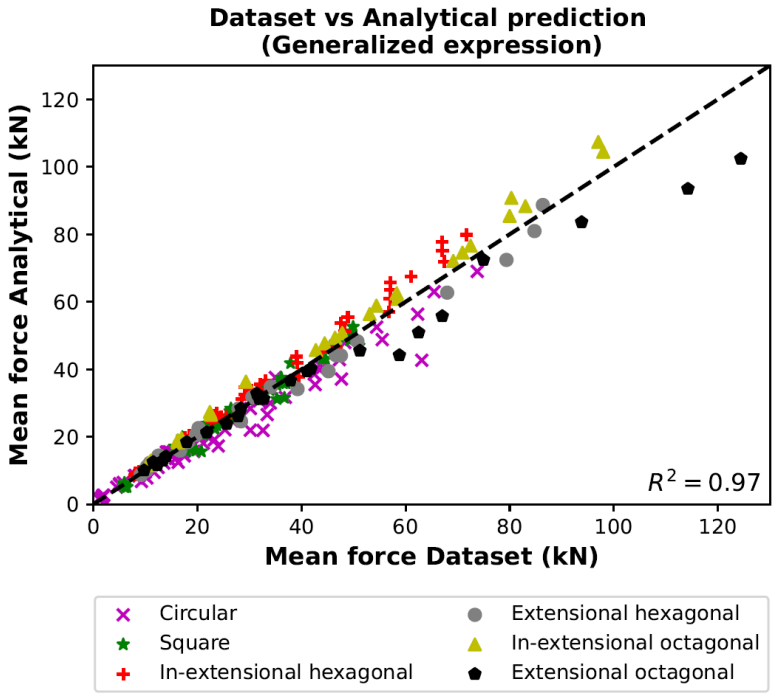


Figure 3.5: Comparison between generalized expression for mean crushing force and mean crushing force dataset for different cases and configurations [83]

In Chapter 3, a numerical model experimentally validated is described. Results generated from this model (see Figure 3.3) will be considered as the expensive and accurate data that are referred as high-fidelity in the following chapters. Moreover, a recent study [83] showed that the mean crushing force can be estimated with a closed-form relationship which provides an additional font of information for the crashworthiness evaluation. In Chapter 4, an investigation takes place on how the aforementioned data sources can be processed in order to predict the energy absorption of a crushing tubular structure and obtain its load-displacement curve as accurate and computationally efficiently as possible using Machine Learning techniques.

4 Methodology

The present chapter discusses the Machine Learning techniques employed in the study, the core of the algorithms that are utilized, the approaches followed to predict the energy absorption, and the methodologies developed to increase the predictive accuracy. The multi-output regressors utilized are determined to be the most effective among a large list of regressors documented in literature libraries.

4.1. Multi-output Regression in Machine Learning

In the field of Machine Learning, regression analysis is frequently employed to perform predictive tasks involving the estimation of a continuous target variable. However, numerous real-world applications necessitate the prediction of multiple variables simultaneously, leading to the utilization of multi-output regression algorithms. A classification of Machine Learning techniques can be seen in Figure 4.1. Regression is a specific type that belongs to Supervised Learning.

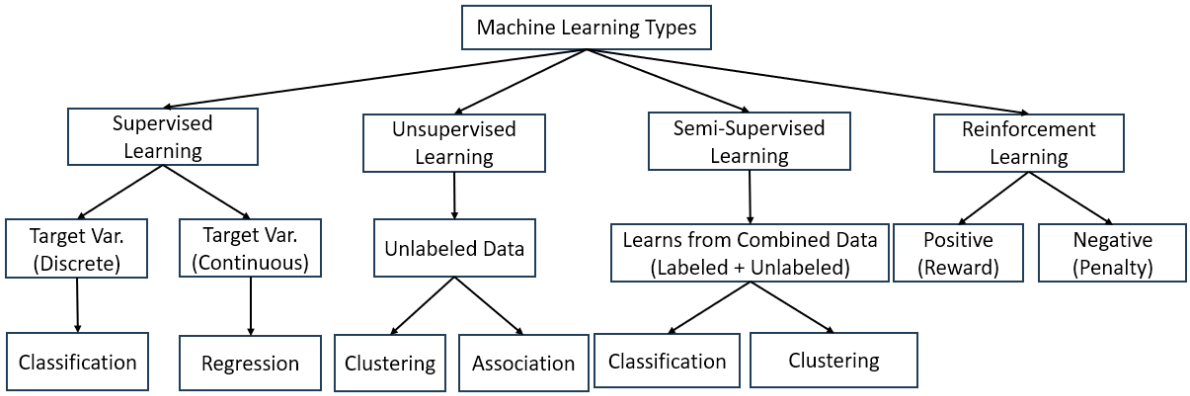


Figure 4.1: Machine Learning techniques classification [91]

Multi-output regression encompasses methodologies aimed at modeling relationships between input features and multiple target variables, thereby enabling the prediction of several outcomes concurrently. This framework is particularly pertinent in crashworthiness analysis to address challenges regarding the structural behavior during crushing, where identifying triggering mechanisms and failure modes is required.

Non-linear responses can emerge from various factors such as high peak loads, plastic deformations from inertial forces, alterations in boundary conditions, material and joint failures, high rates of deformation, contact forces, large deformations, and rivet failures. Predicting the load-displacement curves can give insight on the evaluation and assessment of the aforementioned non-linear behavior. The schematic of multi-output regression for predicting load-displacement curves is illustrated in Figure 4.2.

Figure 4.2: The schematic of multi-output regression when predicting functions

In the present study, the focus is on regressors where geometric features serve as input and load-displacement curves are the output. This approach facilitates understanding of the relationship between geometric features and the non-linear structural behavior in crashworthiness scenarios.

There are some models in Machine Learning that inherently support performing multi-output regression. These models encompass a range of regression techniques. The Linear Regressor, for instance, assumes a linear relationship between input features and targets, employing a straight line to predict the target variable by minimizing the difference between predicted and actual values.

The Random Forest Regressor utilizes ensemble learning, combining multiple Decision Trees to predict regression tasks, with the final prediction being an average of all individual tree predictions. The Extra Trees Regressor employs ensemble learning but introduces additional randomness by sub-sampling the input data, potentially reducing overfitting. Extra Trees use the original sample entirely, enhancing robustness by selecting random splits instead of optimal ones. Another algorithm that inherently supports performing multi-output regression is the K-Neighbors Regressor. K-Neighbors Regressor estimates target values by averaging the target values of its k nearest neighbors in the feature space, making it adaptable and effective, particularly when local patterns among features are significant.

In case the Machine Learning algorithm does not support performing multi-output regression a separate regressor could be trained for each target variable. This augments the functionality of regressors originally intended for single-output regression tasks in the context of multi-target regression endeavors and could be achieved for example through the utilization of a Support Vector Regressor (SVR). By integrating SVR into the multi-output regressor framework, which serves as a wrapper over SVR, the methodology effectively extends SVR's applicability to handle multi-output regression tasks.

Another approach is chained Multi-output Regression, also known as Regression Chain. This entails organizing individual regression models into a sequential chain where each model within the chain predicts a target label by considering all available input features alongside the predictions made by preceding models in the sequence. This sequential chaining strategy benefits from both feature information and insights gained from earlier model predictions, thereby enhancing the accuracy of multi-output predictions.

4.2. Description of the Algorithms Employed

For the selection of the optimal regressor, a considerable challenge is faced due to the plethora of regressors available in Python toolboxes online [92, 93]. The selection is based on tested performance rather than a detailed analysis of each regressor's mathematical background. Following this rigorous evaluation process, six regressors that consistently demonstrated superior performance across various metrics (such as Mean Absolute Percentage Error) are identified. An overview of their baseline is documented in the present section. The Table 4.1 shows which toolbox was used for each algorithm.

Table 4.1: Algorithms and relevant python toolboxes used in the study

Prediction Method	Scikitlearn [92]	SMT [93]
Transform Target Regressor	✓	
Adaptive Boosting Regressor	✓	
Poisson Regressor	✓	
Decision Tree Regressor	✓	
Radius Neighbor Regressor	✓	
Gaussian Processes		✓

4.2.1. Transformed Target Regressor

The Transformed Target regressor [92] serves as a meta-estimator utilized for regression on a transformed target. This approach proves beneficial for applying non-linear transformations to the target in regression scenarios. The regressor performs transformations on the targets before fitting a regression model, and subsequently maps the predictions back to the original space via an inverse transform. Its parameters include the regressor utilized for prediction and the transformer applied to the target variable. In cases of simple transformations, instead of a transformer object, a function can be provided, defining the transformation and its inverse mapping.

Common examples include the quantile transformation which is a method that transforms the features to follow a uniform or a normal distribution or the logarithmic transformation and its reverse operation. During each fitting process, these provided functions undergo validation to ensure their mutual inverses. Nonetheless, it remains possible to bypass this validation step. Figure 4.3 depicts the probability density functions of the target both before and after the application of logarithmic functions.

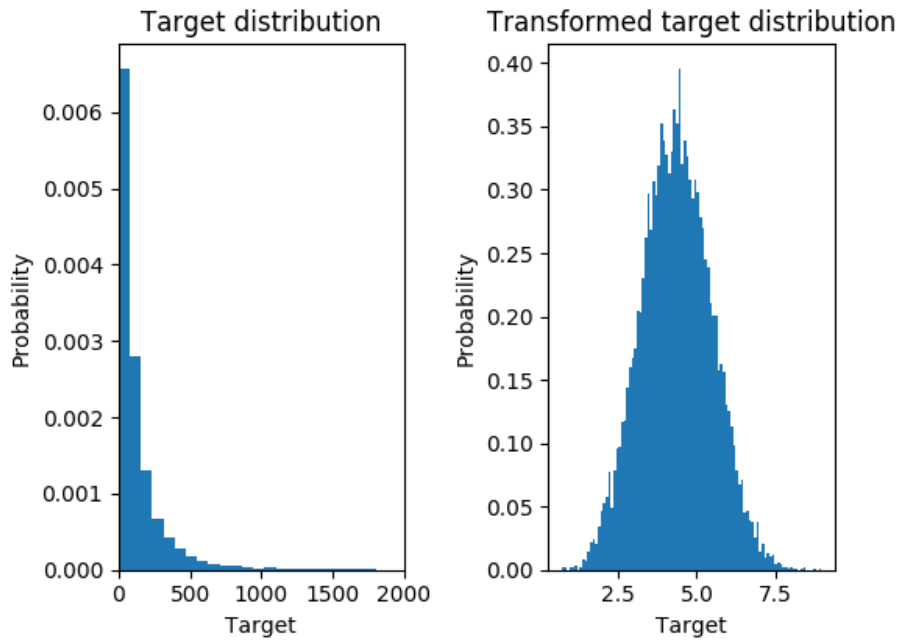


Figure 4.3: Target distribution before and after the Target transformation [92]

4.2.2. Adaptive Boosting Regressor

The Adaptive Boosting algorithm (AdaBoost) belongs to the category of Boosting techniques utilized as an Ensemble method in Machine Learning [92, 94–97]. It is used for weak learners and exhibits adaptability by focusing on instances misclassified by previous classifiers. AdaBoost's properties are sensitive to noisy data and outliers while the issue of over-fitting is less pronounced compared to other learning algorithms. Although individual learners are weak, as long as each one's performance slightly surpasses random guessing, the final model will converge to a strong learner. AdaBoost specifically denotes a method of training a boosted classifier. A boosted classifier is expressed by the following equation:

$$F(x) = \sum_{T=1}^{T} f_t(x) \tag{4.2.1}$$

In equation 4.2.1, each $f_t(x)$ is defined as a weak learner, which accepts x as input for an object and produces a value indicating the object's class. For instance, in a two-class problem, the sign of the weak learner's output signifies the predicted class of the object, while the absolute value provides confidence in that classification. The T_{th} classifier is considered positive if the sample belongs to the positive class or indicates negativity otherwise. In the initial stage of the Adaptive Boosting algorithm, a training subset is randomly chosen.

Subsequently, the AdaBoost Machine Learning model undergoes iterative training, wherein the selection of the training set is based on the accurate predictions from the previous training rounds. Higher weights are assigned to incorrectly classified observations, thereby increasing their likelihood of being classified correctly in subsequent iterations.

Furthermore, weights are allocated to the trained classifiers in each iteration based on their accuracy; classifiers with higher accuracy receive greater weight. This iterative process continues until either the entire training dataset fits without error or until the specified maximum number of estimators is reached. For classification, a "vote" is conducted across all the learning algorithms that have been constructed. A representation of these steps can be seen in Figure 4.4.

4.2.3. Poisson Regressor

Poisson Regressor belongs to generalized linear models with a Poisson distribution [92, 98–100]. It is considered as a simple count regression model, involving coefficients that are exponentiated due to the requirement that counts must be non-negative (0 or greater). Poisson regression operates under the assumption of a Poisson distribution, typically featuring a significant positive skew where the majority of cases cluster at the lower end of the dependent variable's distribution. Additionally, it assumes a variance equal to the mean. Given that count data distributions, such as visit counts, frequently conform to a Poisson distribution, Poisson regression typically provides a superior fit for such data compared to linear regression, which assumes a normal distribution.

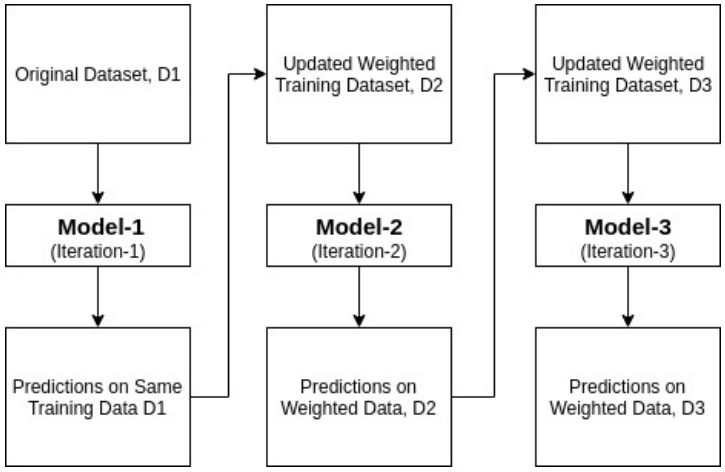


Figure 4.4: Adaptive Boost algorithm [101]

Consequently, it enables the examination of predictive relationships with a dependent variable, such as visit counts, similar to ordinary linear regression. This is achieved without encountering issues stemming from non-normal distributions and heteroscedasticity commonly associated with visit counts and costs. However, Poisson regression has a significant limitation. It necessitates that the variance of the count variable does not exceed its mean. When this assumption is violated, resulting in "overdispersion" standard errors become deflated, leading to Type I errors, rendering Poisson regression inappropriate. Various tests within software programs can evaluate the presence of significant overdispersion.

4.2.4. Nearest Neighbor Regressor

The Nearest Neighbor Search (NNS) problem is defined as follows: within a multidimensional space X containing a set P of n points and characterized by a distance function D, the task is to implement an algorithm that, when presented with a query point $q \in X$, identifies the point p from set P that minimizes the distance $D(q, p)$. When regression relies on neighbors within a fixed radius, the algorithm used is called Radius Neighbors Regressor. It essentially predicts the target by locally interpolating the targets associated with the nearest neighbors in the training set [102–104]. k-Nearest Neighbor (kNN) algorithm is a commonly used classification method for unknown data. In the training process, the training dataset is loaded and stored.

Subsequently, in the testing phase, k nearest neighbors are searched from the training dataset, where k is the hyperparameter chosen initially. The voting method is generally applied in the classification task, wherein the label of x_0 is determined by selecting the most frequent label among its k nearest neighbors. Similarly, in the regression task, the label of x_0 is determined by setting it as the mean value of the labels of its k nearest neighbors. An ilustration is depicted in Figure 4.5.

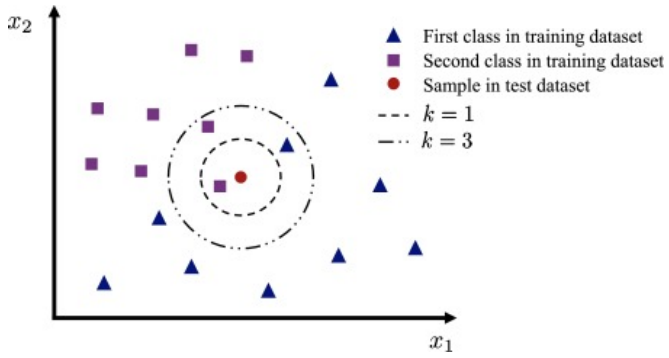


Figure 4.5: k-Nearest Neighbor algorithm for k=1 and k=3 [105]

The algorithms provided by [92] toolbox encompasses both unsupervised and supervised neighbors-based learning methods. Unsupervised nearest neighbors form the basis of various other learning techniques for learning and spectral clustering. Supervised neighbors-based learning manifests in two forms: classification for data with discrete labels, and regression for data with continuous labels. The essence of nearest neighbor methods lies in identify a predefined number of training samples closest in distance to the new point and predicting the label based on these neighbors. The user can specify the number of samples either as a constant (k-nearest neighbor learning) or allow it to vary based on the local density of points (radius-based neighbor learning). The distance metric can be any measure, with the standard Euclidean distance being a common choice. Neighbors-based methods are considered non-generalizing Machine Learning techniques, as they simply retain all their training data potentially transformed into a fast indexing structure such as a Ball Tree or K-dimensional tree. Nearest neighbors algorithm has proven successful in numerous classification and regression tasks, such as recognizing handwritten digits and classifying satellite image scenes. Being a non-parametric method, it often performs well in classification scenarios where the decision boundary is highly irregular. Other learning algorithms such as kernel density estimation, rely on

nearest neighbors as their core component [106].

4.2.5. Decision Tree Regressor

Decision Tree learning is a method that employs a Decision Tree (DT) structure to predict target values at the leaves based on observations in the branches [92, 107, 108]. Arranged in a straightforward tree structure, Decision Trees feature terminal nodes that display decision outcomes, while non-terminal nodes represent tests on one or multiple attributes. In this approach, the quality of a split at each node of the Decision Trees is measured by its ability to reduce the Mean Squared Error (MSE). However, alternative criteria such as Mean Absolute Error (MAE) can be considered, which aims to minimize the L1 loss by utilizing the median of each terminal node. A typical structure of a Decision Tree can be seen in Figure 4.6

Decision Trees offer the advantage of facilitating multi-target joint feature selection and prediction by taking into account inter-correlations between the targets. Furthermore, Decision Trees are characterized by their ease of understanding and high interpretability while they do not require data scaling, and they can effectively capture nonlinear relationships within the data.

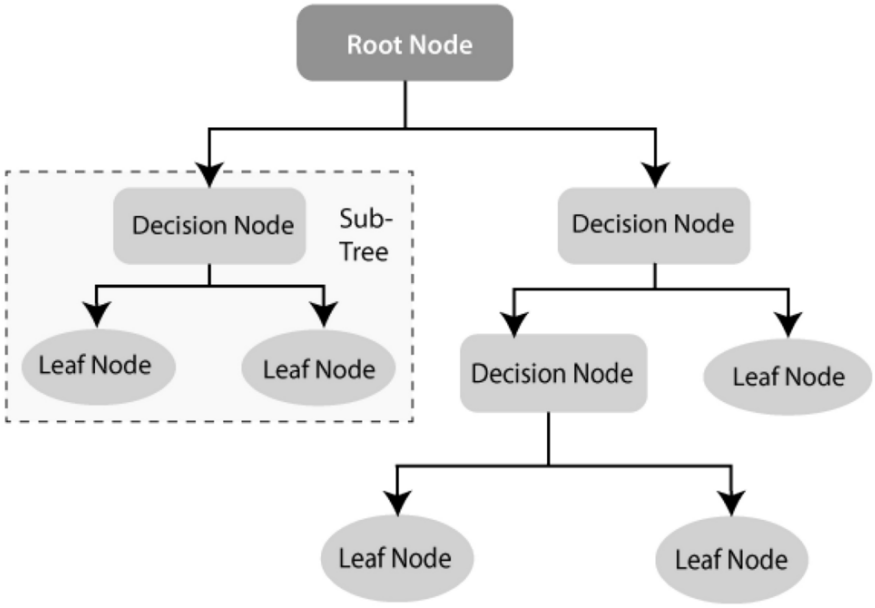


Figure 4.6: Decision Tree structure [91]

However, Decision Trees are prone to overfitting, where the model excessively tailors itself to the training data, compromising its ability to generalize and predict future observations reliably. To mitigate overfitting, it's common practice to limit the depth of the trees. Scikit-learn library [92] and specifically the Decision Tree Regressor is also one of the algorithms employed.

4.2.6. Gaussian Processes

Gaussian Process, often mentioned in literature as Kriging, is a statistical surrogate model that enables the approximation of any unknown mapping black-box function $f(\cdot)$ within its input space by considering it as a realization of a Gaussian Process [93, 109–111] . Essentially, a Gaussian Process (GP) defines a distribution across a set of functions, corresponding an ensemble of infinite random variables, each finite subset of which follows a joint Gaussian distribution.

The Gaussian Process is characterized by the covariance functions and mean while its framework entails a supervised learning problem, wherein it is trained on a dataset or Design of Experiment (DoE) comprising M samples, $\mathcal{X}_M = \left\{\mathbf{x}^1, \ldots, \mathbf{x}^M\right\}$ being the input data set $\left(\mathbf{x} \in \mathbb{R}^d\right)$ and the corresponding unknown function responses noted as $\mathcal{Y}^M = \left\{y^1 = f\left(\mathbf{x}^1\right), \ldots, \quad y^M = f\left(\mathbf{x}^M\right)\right\}$.

The exact function response $f(\cdot)$ can be predicted by the meta-model at a new unmapped location without evaluating it. The reduced computational evaluation cost is considered as the main advantage of this approach since instead of evaluating the expensive black-box function $f(\cdot)$, the surrogate model is used for evaluation. In the GP regression, a prior covariance function $k^{\mathbf{\Theta}}\left(\mathbf{x}, \mathbf{x}'\right)$ is used to place a GP prior on the unobserved function $f(\cdot)$. This covariance function depends on hyper-parameters $\mathbf{\Theta}$ and a mean function $m(\cdot)$. A commonly used kernel referred as p-exponential kernel is given in Equation 4.2.2.

$$k^{\Theta}\left(\mathbf{x}, \mathbf{x}'\right) = \Theta_\sigma \exp\left(-\sum_{i=0}^{d} \Theta_{\theta_i} \left|\mathbf{x}^{(i)} - \mathbf{x}'^{(i)}\right|^{\Theta_{p_i}}\right) \tag{4.2.2}$$

A constant mean function, denoted by μ, is commonly assumed, leading to what's referred to as ordinary Kriging in Gaussian Process (GP) modeling when the trend of the response is uncertain beforehand. Different types of mean functions such as linear, quadratic, or general basis functions can be considered based on available prior knowledge. Under the assumption of a constant mean function, the GP can be represented as $\hat{f}\left(\mathbf{x}'\right) \sim \mathcal{N}\left(m, k^{\mathbf{\Theta}}\left(\mathbf{x}, \mathbf{x}'\right)\right)$, where the function has a multivariate normal distribution on any finite subset of variables, notably on the Design of Experiment (DoE) $\mathcal{X}^M$ (denoted as $\mathbf{f}^M\Big), \mathbf{f}^M \mid \mathcal{X}^M \sim \mathcal{N}\left(\mathbf{1}\mu, \mathbf{K}_{MM}^{\Theta}\right)$. Here, $\mathbf{K}_{MM}^{\Theta}$ represents the covariance matrix derived from the parameterized covariance function $k^{\mathbf{\Theta}}(\cdot)$ on $\mathcal{X}^M$.

The selection of the covariance function dictates the prior assumptions regarding the modeled function. Introducing a Gaussian noise variance, often referred to as the nugget effect, allows for the incorporation of uncertainty in the relationship between the latent function values $f(\mathcal{X}^M)$ and the observed responses $\mathcal{Y}^M$. This relationship can be expressed as $p(\mathbf{y} \mid \mathbf{f}^M) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}^M, \sigma^2\mathbf{I})$. The marginal likelihood is derived by integrating out the latent function $f(\cdot)$:

$$p\left(\mathbf{y} \mid \mathcal{X}^M, \mathbf{\Theta}\right) = \mathcal{N}\left(\mathbf{y} \mid m, \mathbf{K}_{MM} + \sigma^2\mathbf{I}\right) \tag{4.2.3}$$

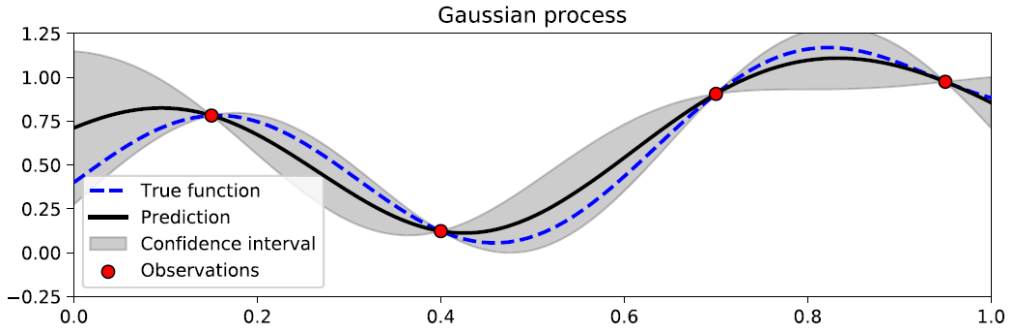


Figure 4.7: Gaussian process prediction and its confidence interval [109]

The optimal values of the hyper-parameters $\boldsymbol{\Theta}, m$ and σ can be determined by maximizing the log marginal likelihood in order to train the Gaussian Process. The negative log marginal likelihood and its derivative are defined in Equation 4.2.4 and Equation 4.2.5 respectively. The kernel matrices implicitly depend on the hyperparameters $\boldsymbol{\Theta}$ along the previously two mentioned quantities.

$$L\left(\boldsymbol{\Theta} \mid \mathcal{X}^M, \mathcal{Y}^M\right) = \log\left(p\left(\mathbf{y} \mid \mathcal{X}^M, \mathcal{Y}^M, \boldsymbol{\Theta}\right)\right) \propto \log\left(\left|\hat{\mathbf{K}}_{MM}\right|\right) - \mathbf{y}^T \hat{\mathbf{K}}_{MM}^{-1} \mathbf{y} \tag{4.2.4}$$

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\Theta}} = \mathbf{y}^T \hat{\mathbf{K}}_{MM}^{-1} \frac{\mathrm{d}\hat{\mathbf{K}}_{MM}}{\mathrm{d}\boldsymbol{\Theta}} \hat{\mathbf{K}}_{MM}^{-1} \mathbf{y} + \mathrm{Tr}\left(\hat{\mathbf{K}}_{MM}^{-1} \frac{\mathrm{d}\hat{\mathbf{K}}_{MM}}{\mathrm{d}\boldsymbol{\Theta}}\right) \tag{4.2.5}$$

with $\hat{\mathbf{K}}_{MM} = \mathbf{K}_{MM} + \sigma^2 \mathbf{I}$.

Optimization algorithms can be employed to minimize the negative log marginal likelihood for determining the hyperparameter values of the trained Gaussian Process (GP). Additionally a closed form of the constant mean function can be obtained rather than being determined through an optimization algorithm, as indicated in references. Following the training phase, predictions at a new point $\mathbf{x}^* \in \mathbb{R}^d$ are made by utilizing the conditional properties of a multivariate normal distribution expressed in Equation 4.2.6 and illustrated in Figure 4.7. The mean prediction $\hat{y}^*$ is given in Equation 4.2.6 and the associated variance $\hat{s}^{*2}$ is given in Equation 4.2.7.

$$p\left(y^* \mid \mathbf{x}^*, \mathcal{X}^M, \mathcal{Y}^M, \boldsymbol{\Theta}\right) = \mathcal{N}\left(y^* \mid \hat{y}^*, \hat{s}^{*2}\right) \tag{4.2.6}$$

$$\hat{y}^* = m + \mathbf{k}_{\mathbf{x}^*}^T \left(\mathbf{K}_{MM} + \sigma^2 \mathbf{I}\right)^{-1} (\mathbf{y} - \mathbf{1}\mu) \tag{4.2.7}$$

$$\hat{s}^{*2} = k_{\mathbf{x}^*,\mathbf{x}^*} - \mathbf{k}_{\mathbf{x}^*}^T \left(\mathbf{K}_{MM} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{k}_{\mathbf{x}^*} \tag{4.2.8}$$

where $k_{\mathbf{x}^*,\mathbf{x}^*} = k\left(\mathbf{x}^*, \mathbf{x}^*\right)$ and $\mathbf{k}_{\mathbf{x}^*} = \left[k\left(\mathbf{x}_{(i)}, \mathbf{x}^*\right)\right]_{i=1,\ldots,M}$
During the training phase of Gaussian Processes the computational burden is primarily carried by operations such as computing the linear solve $\hat{\mathbf{K}}_{MM}^{-1}\mathbf{y}$, evaluating the log determinant $\log\left(\left|\hat{\mathbf{K}}_{MM}\right|\right)$, and determining the trace term $\mathrm{Tr}\left(\hat{\mathbf{K}}_{MM}^{-1}\frac{\mathrm{d}\hat{\mathbf{K}}_{MM}}{\mathrm{d}\boldsymbol{\Theta}}\right)$. These tasks rely on the Cholesky decomposition of $\hat{\mathbf{K}}_{MM}$, which is computationally demanding and necessitates $\mathcal{O}\left(M^3\right)$ operations. Sparse GP can be used in order to mitigate these computational costs.

A 1D function and the corresponding Gaussian Process (GP) constructed from four observations is illustrated in Figure 4.7. The confidence interval, determined by the GP variance ($\pm 3\hat{s}^{*2}$), shows no variance at observation points without a nugget effect and increases with distance from existing data. GPs are valuable in complex system design, offering both model predictions and uncertainty estimates to designers.

4.3. Predictive Approaches for Energy Absorption Evaluation

The evaluation of the energy absorption capability of a structural element requires to access data related either to the load-displacement curve or to the area below the load-displacement curve directly which represents the energy absorbed by the system. The development of meta-models able to perform these predictions is essential due to the significant computational resources needed for explicit finite element simulations and the challenges associated with experiments. The use of regression models that they are either multi-output for the case of the load-displacement curve prediction or single-output for the case of the energy absorption prediction is investigated. These approaches are described in the following two subsections.

4.3.1. Prediction of Energy Absorption as Scalar Value

One approach to asses the crashworthiness performance is to employ one of the Machine Learning algorithms described in Section 4.2 to predict a single output, or target, which is the energy absorption. The schematic of the single-output regressor used to estimate the energy absorption is illustrated in Figure 4.8. The geometrical features herein considered are: material, the thickness, the side length and the tube length. In other words the same surrogate model can have as input load-displacement curves derived from different materials, thicknesses, side lengths and tube lengths.

Figure 4.8: The schematic of single-output regression

The drawback of this approach is that the only information that becomes available after training the meta-models is the energy absorption as scalar value. Thus, no information is estimated regarding the load-displacement curve.

4.3.2. Prediction of Load-displacement Curve

Another approach to asses the crashworthiness performance is to employ the Machine Learning algorithms described in Section 4.2 by considering the values of the curve as the targets to be predicted. The motivation for predicting the load-displacement curve of each structural element individually lies in the fact that the superposition of these curves enables the estimation of the crashing behavior of the subfloor.

The schematic of the multi-output regressor used to estimate the load-displacement curve is illustrated in Figure 4.9. The material, the thickness, the side length and the tube length are considered as "Geometrical Features". The value of the load at displacement equal to d_i for $i = 1, ..., m$, is represented by $f(d_i)$. The curves, which are generated as results of the finite element analysis using Abaqus (see Figure 3.3), consist of 1000 points or m equals 1000. These curves are the high-fidelity data mentioned in Chapter 3, which are emulating what would be available from a physical test.

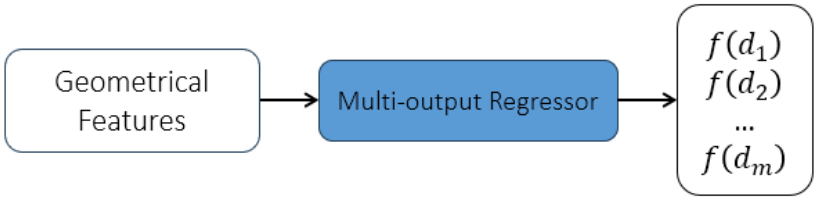


Figure 4.9: The schematic of multi-output regression

The main goal is to assess the crashworthiness performance meaning that after estimating the load-displacement curve with the multi-output regressor another step is needed, the derivation of the integral below the predicted curve, f_{pred}, as illustrated in Figure 4.10. This area corresponds to the energy absorbed by the structure and in the present analysis this task is performed with Simpson's rule [112]. When the units of the loading is in kN and the units of the displacement is in mm, the calculated energy absorption (or area) is in J.

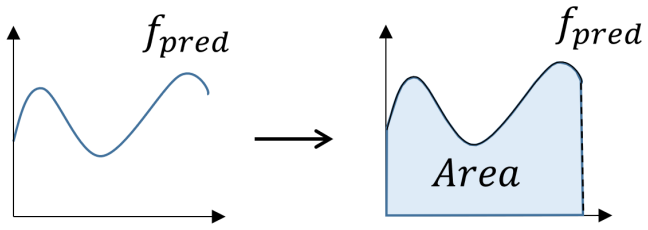


Figure 4.10: Energy absorption calculation after the load-displacement curve estimation

To evaluate how well the load-displacement curve is estimated along with how well the energy absorption is estimated, the Mean Absolute Percentage Error (MAPE) metric is used as formulated in Equation 4.3.1. It is a regression loss designed to evaluate the

accuracy of regression models. It quantifies the average absolute percentage difference between predicted and actual values, making it useful for understanding relative errors across different scales of target variables. MAPE is used within the results in the format of percentage (%).

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{4.3.1}$$

where y_i represents the true value, $\hat{y}_i$ represents the predicted value, and n is the number of samples. The predictive approach that forecasts the load-displacement curve is adept at estimating the entire curve, despite the higher computational demand of predicting 1000 targets compared to a single one.

4.4. Methodologies and Techniques to Improve Training Performance

After defining the potential regression models, a process to evaluate its performance and prediction accuracy is required. Two methodologies are described in the present section, both employing multi-output regressors since the ability to predict the load-displacement curve is deemed necessary. In the first methodology the surrogate model is solely trained with numerical data derived from a high-fidelity (HF) model. Data from high-fidelity models are usually expensive and accurate that come from experiments or computationally intense simulations.

In the second methodology a low-fidelity (LF) model is introduced along with the high-fidelity model to improve the accuracy of the predictions. The low-fidelity model is a simplified model that relies on the numerical data and the analytical equations presented in Chapter 3.

After the creation of the low-fidelity model samples can be generated without computationally demanding processes for any geometrical characteristics (thickness, side length and tube length) and material input. This low-fidelity samples in combination with the available numerical samples from the high-fidelity model become now the input of the Machine Learning models for the second methodology. The ultimate goal is to improve the prediction performance of the energy absorption and compare the results between the two methodologies.

4.4.1. Regression Solely Based on HF Samples

The initial methodology aims to predict the energy absorption, when training is solely based on high-fidelity (HF) samples, as illustrated in Figure 4.11. The precision of energy absorption prediction directly correlates with the efficacy of Regressor A (see Figure 4.11) in forecasting load-displacement curves.

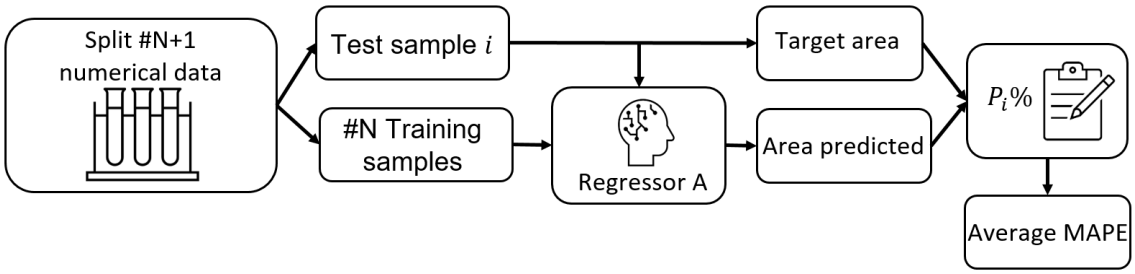


Figure 4.11: Methodology initially employed to predict the energy absorption

Firstly, the available $N+1$ numerical samples (or FEA samples or high-fidelity samples) are separated in to N training samples and 1 test sample i. The purpose of this is to train the Regressor A with the N training samples and then evaluate its performance by comparing with the test sample i, considering it as the ground truth. Given the scarcity of data, the test sample is selected to be only one so as to train the regressor with the highest possible amount of samples.

After training the Regressor A with the N training samples, the features (material, thickness, side length and tube length) of the test sample i serve as input to the Regressor and a load-displacement curve is estimated as output of it from which the area below the curve is calculated. Thus, a comparison with the target area which is computed from the test sample's i load-displacement curve, can be made. Comparing these two scalar values the Mean Absolute Percentage Error P_i % is calculated.

To get a better understanding of how the Regressor A performs the process is repeated so that all the available samples become once the test sample i. The outcome is a $(N+1) \times 1$ array including all the MAPEs, the average of which is calculated in the final step. This average represents the performance of the Regressor A, since it can give an overall insight on how accurate the energy absorption predictions are.

4.4.2. Regression Based on HF and LF Samples

The goal is to improve the prediction performance of the high-fidelity model, described in Subsection 4.4.1, by introducing low-fidelity information as illustrated in Figure 4.12. The blue loop depicts the difference between the two methodologies where low-fidelity samples are incorporated in the training process. This approach employs two completely different regressors.

Regressor A is a 1000-target regressor used to estimate the load-displacement curves after being trained with $N+M$ samples. Regressor B is a dual-target regressor used to estimate the P_{max} and the corresponding displacement, $x_{P_{max}}$ after being trained with N training samples in order to construct the low-fidelity model in the next step.

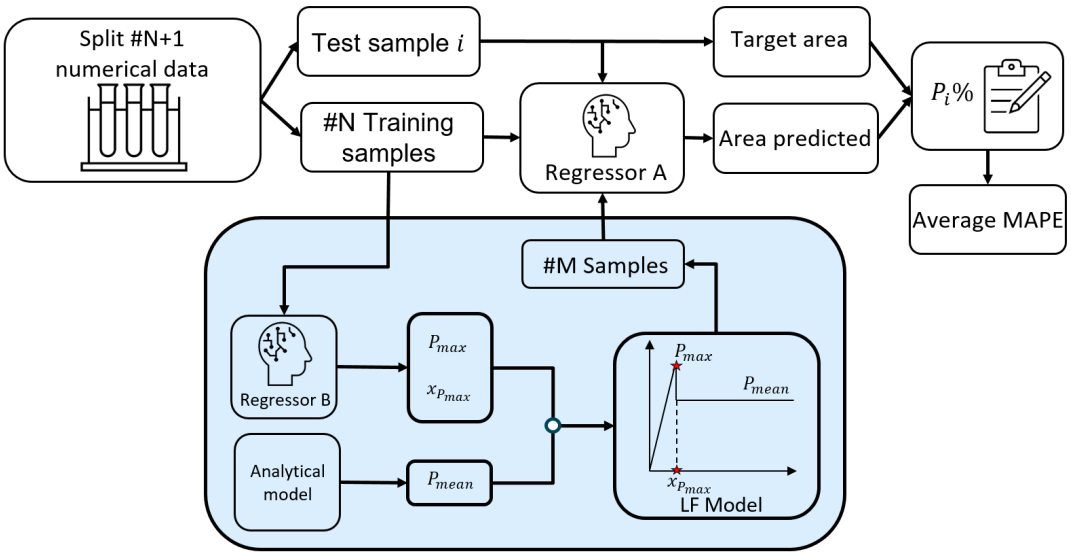


Figure 4.12: Methodology employed for improved prediction capability of the energy absorption

Similarly to the high-fidelity model the available $N+1$ numerical data are separated in to N training samples and 1 test sample i. Regressor A is trained with the N training samples and M samples generated by the low-fidelity model. Then its performance is evaluated by comparing with the test sample i, considering it again as the ground truth.

The M samples, referred as low-fidelity samples, are generated as follows. The N training samples (which consist the high-fidelity data) serve as input to the dual-target Regressor B so as to create a model able to predict the P_{max} and the corresponding displacement, $x_{P_{max}}$. This in combination with the the analytical estimation of the P_{mean}, given in Equation 3.2.6, can be used to construct a low-fidelity curve as shown in Figure 4.13 for any desired input in terms of material, thickness, side length and tube length [83]. The load-displacement curve of that simplified model is assumed to consist of a linear part, a peak value and a plateau similarly working as Astori et al. [74] (see Figure 2.25).

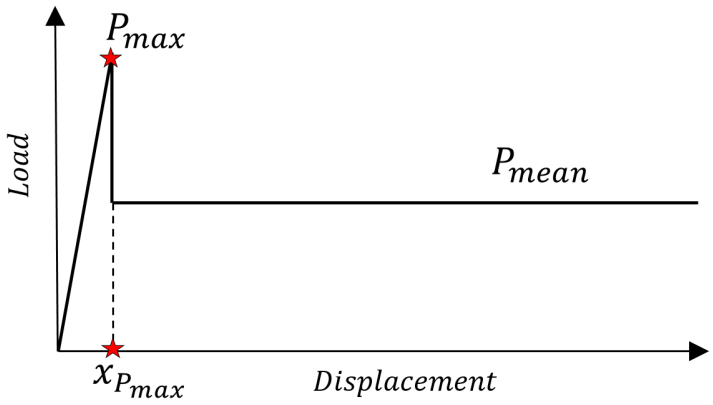


Figure 4.13: Low-fidelity model to represent a load-displacement curve

Thus, P_{mean} is analytically estimated while the P_{max} and the displacement $x_{P_{max}}$ are estimated with a dual-target regression model, appearing as Regressor B in Figure 4.12.

After constructing the low-fidelity model, M low-fidelity samples are generated. The number of samples generated by the low-fidelity model depends on the user. The low-fidelity model is based on the input taken by the Regressor B and the Analytical model which are two completely separate models. On the one hand, Regressor B is a dual-target regressor trained with high-fidelity samples and on the other hand the Analytical model is a closed-form relationship, given in Equation 3.2.6, that estimated the mean crushing load.

After Regressor A is trained with the $N + M$ samples, the features (material, thickness, side length and tube length) of the test sample i serve as input to the Regressor A and a load-displacement curve is estimated as output of it, from which the area below the curve is derived. Thus, a comparison with the target area can be made computed from the test sample's i load-displacement curve. Comparing these two scalar values the Mean Absolute Percentage Error P_i % is calculated.

The process is repeated so that all the available numerical samples become once the test sample i. The outcome is a $(N + 1) \times 1$ array including all the MAPEs, the average of which is calculated in the final step. This average represents the performance of the Regressor A since it can give an overall insight on how accurate the predictions of the energy absorption are.

Both methodologies described in Section 4.4 are evaluated for the following multi-output regressors (referring to regressor A): Transform Target Regressor, Poisson Regressor, Decision Tree regressor, and Radius Neighbor Regressor. Running the Adaptive Boosting Regressor and Gaussian Process models proved to be computationally infeasible to reproduce the graphs shown in Section 5.2. The Regressor B is selected to be the Transform Target Regressor to predict the P_{max} and the $x_{P_{max}}$. Other regressors are also utilized; however, they exhibited comparable satisfactory performance. Consequently, no comparative analysis is conducted within this study among Regressor B's alternatives. The Transform Target Regressor is deemed adequate for comparing the approaches outlined in 4.4.

4.4.3. Evaluation of the LF Model

Before employing the low-fidelity model, shown in Figure 4.13 as main part of the methodology depicted in Figure 4.12, its imperative to evaluate how well it can represent the respective high-fidelity samples. Utilizing a low-fidelity model devoid of satisfactory predictive performance would inherently lack the capacity to enhance the accuracy of energy absorption prediction. For this purpose the evaluation process described in this section is performed. For the square cross-section 40 high-fidelity samples (FEA) are available. The low-fidelity model is relying on 39 of these high-fidelity samples and one is used as target sample. This can be illustrated in Figure 4.14 by selecting one target sample with Cross-section: Square, Material: AA6060-T4, Side length: 30 mm, Thickness: 2.25 mm, and Tube length: 200 mm

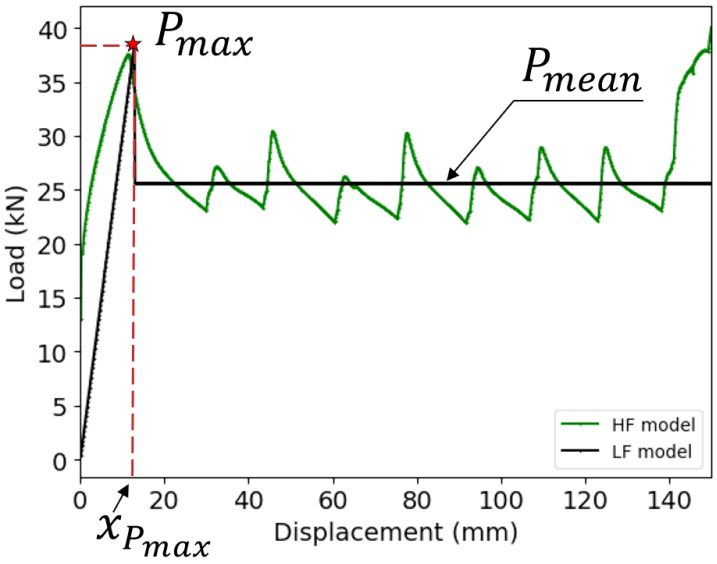


Figure 4.14: Comparison of LF-Model and HF-Model load-displacement curve

To evaluate the accuracy of the LF-Model for this specific case illustrated in Figure 4.14 the following indicators are computed: MAPE (Mean Absolute Percentage Error) of P_{max} is equal to 2.7%, MAPE of $x_{P_{max}}$ is equal to 13.2%, MAPE of P_{mean} is equal to 4.6%, and MAPE of Area (or energy absorption) is equal to 4.6%. The same process can be repeated for all the 40 available high-fidelity samples. Then the aforementioned MAPEs can be computed for each of these 40 samples and the average values can be derived. The average MAPEs of these 40 samples are shown in Table 4.2.

Table 4.2: Average MAPEs for the square cs samples with reference to the HF Model

Estimated Variable	Average MAPE %
P_{max}	9.0
$x_{P_{max}}$	21.6
P_{mean}	3.4
Energy Absorption	3.5

The low-fidelity model has as a primary objective to increase the accuracy of the energy absorption predictions. The energy absorption is derived as the area below the load-displacement curve which is mainly relying on the P_{mean} compared to the other estimated variables. This means that a good prediction of the P_{mean} will be of outmost importance for a good energy absorption prediction.

This argumentation is confirmed also from the close values of average MAPEs P_{mean} and Energy Absorption have in Table 4.2. Achieving average MAPE below 5% for the square cross-section configuration is a promising outcome that employing a low-fidelity model can increase the predictions of energy absorption.

5 Results

5.1. Evaluation of the Regression Algorithms Performance

Before employing the methodologies described in Chapter 4, the machine learning algorithms are evaluated on how well they perform in predicting the load-displacement curves. Specifically the process illustrated in Figure 4.2 is employed for the four cross-sections (cs). The first set of samples is provided in Table B.1 in Appendix B. It includes 40 samples in total, from which 39 are used for training of the multi-output regressor and 1 is used as target. The target sample is randomly selected. Figure 5.1 illustrates a prediction of the load-displacement curve and evaluation of the Mean Absolute Percentage Error (MAPE) with reference to the load-displacement curve of the target sample. The specimen selected as test specimen (target) is material: AISI-316, side length: 30 mm, wall thickness: 1.25 mm, and tube length: 200 mm.

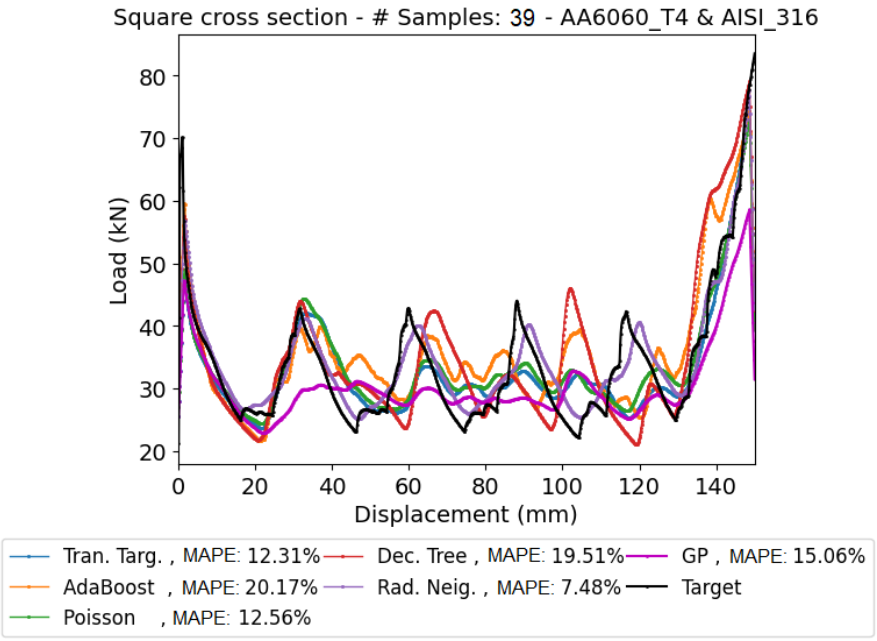


Figure 5.1: Prediction of the load-displacement with 6 different multi-output regressors for the square cs

All the algorithms showed very good correlation for the first 20 mm of crushing length while all except for the GP showed sufficient correlation for the first 45 mm of crushing length. The overall trend of the target curve is captured with the Radius Neighbor Regressor with a MAPE of 7.48%, then with the Transform Target Regressor with a MAPE of 12.31% and the third best performed algorithm is the Poisson Regressor with a MAPE of 12.56%. All six Regressors showed MAPE below 21% compared to the target sample.

After estimating the load-displacement curves with the six multi-output regressors (MO Regr.), the next step is to compare with the target sample the energy absorption by

calculating the areas below the curves. Here another comparison can be made. As mentioned in Chapter 4 the energy absorption can be estimated with single-output regressors (SO Regr.) directly as a scalar value. Thus, apart from evaluating the performance of the multi-output regressors on estimating the energy absorption, their predictions can be compared with the ones of the single-output regressors the schematic of which is illustrated in Figure 4.8.

The evaluation of the MAPE of the areas below the load-displacement curves for the function prediction approach (MO Regr.) and for the scalar value prediction approach (SO Regr.) is presented in Table 5.1. The areas below the curve represent the absorbed energy (J).

Table 5.1: Prediction of the energy absorption (J) with the SO Regr. and for the MO Regr. for the square cs

Prediction Method	Target area	SO Regr.	MAPE %	MO Regr.	MAPE %
Transform T.	5089.44	5050.19	0.77	4944.26	2.85
AdaBoost	5089.44	5329.80	4.72	5334.10	4.81
Poisson	5089.44	5113.06	0.46	5086.53	0.06
Dec. Tree	5089.44	5285.59	3.85	5236.88	2.90
Rad. Neig.	5089.44	5228.56	2.73	5183.97	1.86
GP	5089.44	4947.53	2.79	4615.67	9.31

Transform T. stands for the Transformed Target regressor, AdaBoost stands for the Adaptive Boosting Regressor, Poisson stands for the Poisson Regressor, Dec. Tree stands for the Decision Tree Regressor, Rad. Neig. stands for the Radius Neighbor Regressor and GP stands for the Gaussian Processes. All the results except for the GP that have a MAPE 9.31% are below 5% for the multi-output Regressors while all the single-output Regressors showed MAPE below 5%.

The same process is repeated for the circular cross-sections. The second set of samples is provided in Table B.2 in Appendix B. It includes 32 samples in total, from which 31 are used for training of the multi-output regressor and 1 is used as target sample. Figure 5.2 illustrates a prediction of the load-displacement curve and evaluation of the Mean Absolute Percentage Error with reference to the target sample. The specimen selected as test specimen (target) is material: AA6060-T4, radius: 20 mm, wall thickness: 1.75 mm, and tube length: 200 mm.

The algorithms for the circular cross-section didn't show as good correlation as the square cross-section, even for the first 20 mm of crushing length. The only algorithm that is following the trend of the target sample is the Adaptive Boost Regressor for a crushing length approximately until 100 mm. The overall trend of the target curve is captured with the Adaptive Boosting Regressor with a MAPE of 21.92%, then with the Poisson Regressor with a MAPE of 22.59% and the third best performed algorithm is the Transform Target Regressor with a MAPE of 22.92%. All six Regressors showed

MAPE below 27% compared to the target.

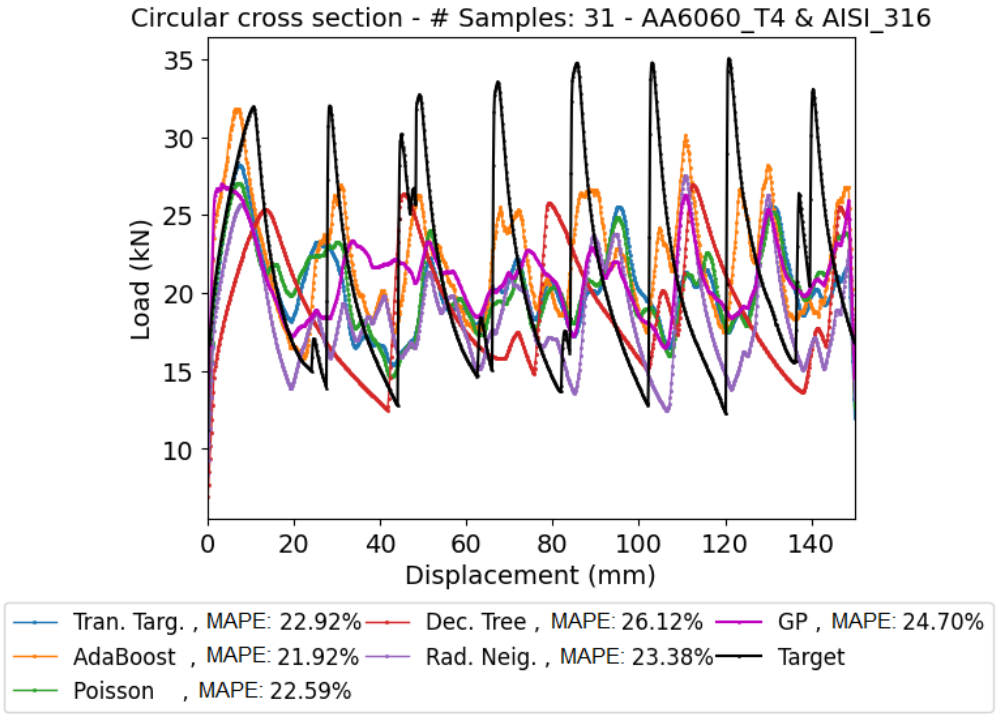


Figure 5.2: Prediction of the load-displacement with 6 different multi-output regressors for the circular cs

After estimating the load-displacement curves with the six multi-output regressors (MO Regr.), the next step is to compare with the target sample the energy absorption by calculating the areas below the curves. Again apart from evaluating the performance of the multi-output regressors on estimating the energy absorption, their predictions are compared again with the ones of the single-output regressors.

The evaluation of the MAPE of the areas below the load-displacement curves for the function prediction approach (MO Regr.) and for the scalar value prediction approach (SO Regr.) is presented in Table 5.2. The areas below the curve represent the absorbed energy (J).

Table 5.2: Prediction of the energy absorption (J) with the SO Regr. and for the MO Regr. for the circular cs

Prediction Method	Target area	SO Regr.	MAPE %	MO Regr.	MAPE %
Transform T.	3192.48	3125.68	2.09	3031.01	5.06
AdaBoost	3192.48	3184.22	0.26	3316.63	3.89
Poisson	3192.48	3058.90	4.18	3050.34	4.45
Dec. Tree	3192.48	3058.90	4.18	2863.74	10.30
Rad. Neig.	3192.48	2780.07	12.92	2766.89	13.33
GP	3192.48	3210.44	0.56	3172.04	0.64

All the MO Regr. results except for the Transform Target Regressor, the Decision Tree Regressor and the Radius Neighbor Regressor are below 5%. The best multi-output regressor appears to be the GP with below 1% prediction. For the single-output regressors all the algorithms except for the Radius Neighbor Regressor showed MAPE below 5%. The best result is shown for the Adaptive Boosting Regressor where the MAPE is 0.26% and the second well performed algorithm is the GP with a MAPE of 0.56%.

The third set of samples refers to the hexagonal cross-section and is provided in Table B.3 in Appendix B. It includes 40 samples in total, from which 39 are used for training of the multi-output regressor and 1 is used as target sample. Figure 5.3 illustrates a prediction of the load-displacement curve and evaluation of the Mean Absolute Percentage Error with reference to the target sample. The specimen selected as test specimen (target) is material: AA6060-T4, side length: 50 mm, wall thickness: 1.75 mm, and tube length: 200 mm.

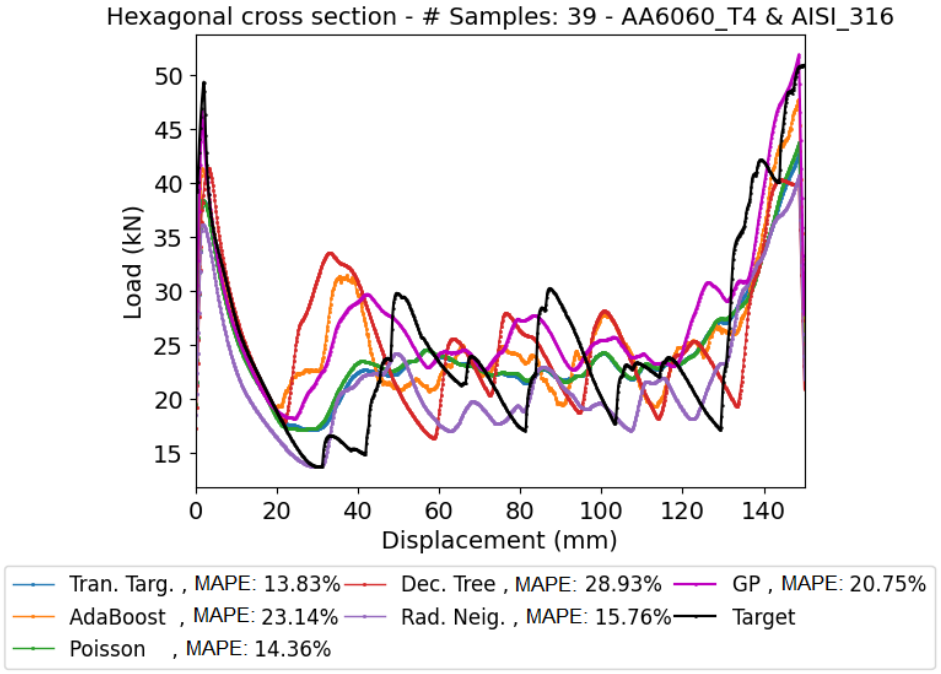


Figure 5.3: Prediction of the load-displacement with 6 different multi-output regressors for the hexagonal cs

The algorithms for the hexagonal cross-section showed better correlation compared to the circular cross-section but not compared to the square cross-section. The Regressors seem to follow the trend with an offset in the horizontal axis. The best performed regressor is the Transform Target Regressor with a MAPE of 13.83%, the second best performed regressor is the Poisson Regressor with MAPE of 14.36%, and the third best performed regressor is the Radius Neighbor Regressor with MAPE of 15.76%. All six regressors showed a MAPE below 29% compared to the target sample.

Again apart from evaluating the performance of the multi-output regressors on estimating the energy absorption compared to the target sample, their predictions are compared

with the ones of the single-output regressors.

The evaluation of the MAPE of the areas below the load-displacement curves for the function prediction approach (MO Regr.) and for the scalar value prediction approach (SO Regr.) is presented in Table 5.3. The areas below the curve represent the absorbed energy (J).

Table 5.3: Prediction of the energy absorption (J) with the SO Regr. and for the MO Regr. for the hexagonal cs

Prediction Method	Target area	SO Regr.	MAPE %	MO Regr.	MAPE %
Transform T.	3748.42	3741.01	0.20	3660.14	2.36
AdaBoost	3748.42	3809.28	1.62	3868.74	3.21
Poisson	3748.42	3714.17	0.91	3678.05	1.88
Dec. Tree	3748.42	3874.38	3.36	3851.07	2.74
Rad. Neig.	3748.42	3304.95	11.83	3279.92	12.50
GP	3748.42	2957.23	21.11	4123.13	10.00

All the MO Regr. results except for the Radius Neighbor Regressor and the Decision Tree Regressor are below 3.3%. The best multi-output regressor appears to be the Poison Regressor with a MAPE of 1.88%. For the single-output regressors all the algorithms except for the Radius Neighbor Regressor and the Decision Tree Regressor are below 3.5%. The best result is shown for the Transformed Target Regressor where the MAPE is 0.20% and the second well performed algorithm is the Poisson Regressor with a MAPE of 0.91%.

The fourth and final set of samples is the octagonal cross-section and is provided in Table B.4 in Appendix B. It includes 26 samples in total, from which 25 are used for training of the multi-output regressor and 1 is used as target sample. Figure 5.3 illustrates a prediction of the load-displacement curve and evaluation of the Mean Absolute Percentage Error with reference to the target sample. The specimen selected as test specimen (target) is material: AA6060-T4, radius: 60 mm, wall thickness: 1.25 mm, and tube length: 200 mm.

The algorithms for the octagonal cross-section showed good correlation, comparable to the one showed with the square cross-section. The regressors seem to follow the trend and magnitude until the 60 mm crushing length. The best performed regressor is the Transform Target Regressor with a MAPE of 12.19%, the second best performed regressor is the Poisson Regressor with a MAPE of 12.89%, and the third best performed regressor is the Decision Tree Regressor with a MAPE of 16.79%. All six regressors showed a MAPE below 31% compared to the target sample.

Again apart from evaluating the performance of the multi-output regressors with the target sample on estimating the energy absorption, their predictions are compared with the ones of the single-output regressors.

The evaluation of the MAPE of the areas below the load-displacement curves for the function prediction approach (MO Regr.) and for the scalar value prediction approach (SO Regr.) is presented in Table 5.4. The areas below the curve represent the absorbed energy (J).

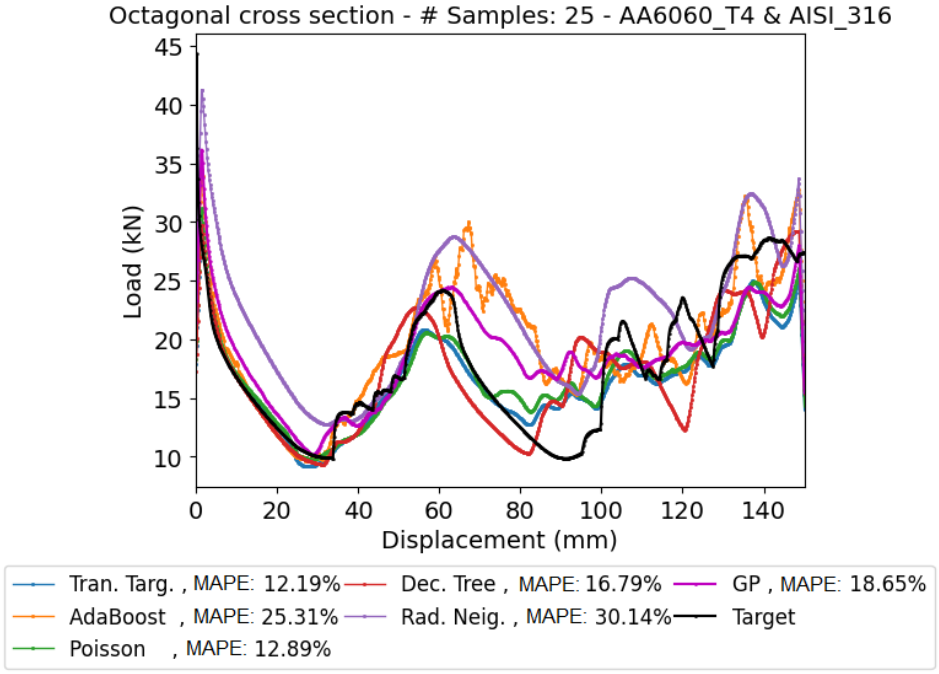


Figure 5.4: Prediction of the load-displacement with 6 different multi-output regressors for the octagonal cs

All the MO Regr. results except for the Adaptive Boosting Regressor have a MAPE below 11%. The best multi-output regressor appears to be the Decision Tree Regressor with a MAPE of 4.27%. The second well performed algorithm is the Poisson Regressor with a MAPE of 4.71% and the third is the GP with a MAPE of 6.31%

Table 5.4: Prediction of the energy absorption (J) with the SO Regr. and for the MO Regr. for the octagonal cs

Prediction Method	Target area	SO Regr.	MAPE %	MO Regr.	MAPE %
Transform T.	2675.32	2542.89	4.95	2486.39	7.06
AdaBoost	2675.32	2581.93	3.49	2962.38	10.73
Poisson	2675.32	2580.80	3.53	2549.31	4.71
Dec. Tree	2675.32	2581.93	3.49	2561.09	4.27
Rad. Neig.	2675.32	3320.57	24.12	3293.83	23.12
GP	2675.32	3253.15	21.60	2844.16	6.31

For the single-output regressors all the algorithms except for the Radius Neighbor Regressor and the GP are below 5%. The best result is shown for the Adaptive Boosting

Regressor and the Decision Tree Regressor where the MAPE is 3.49% while the third best performed algorithm is the Poisson Regressor with a MAPE of 3.53%.

Considering studies such as [35], a maximum error of 7% was achieved for similar approximations. Thus, for the purpose of comparing the performance of the above algorithms, a threshold of 10% is deemed sufficient to predict the energy absorption. The Table 5.5 depicts which of these did show satisfactory performance and managed to predict the energy absorption with MAPE below 10% and for what cross-section. The average run time refers to the time the multi-output regressor takes in order to be trained and estimate the load-displacement curve. This time is calculated for each regressor and each cross-section and the average result is documented in Table 5.5.

Table 5.5: Multi-output regressors that exhibited Mean Absolute Percentage Error below 10% in predicting energy absorption and average run time

Prediction Method	Square cs	Circular cs	Hexagonal cs	Octagonal cs	Average run (sec)
Transform T.	✓	✓	✓	✓	1.6
AdaBoost	✓	✓	✓		58
Poisson	✓	✓	✓	✓	7.6
Dec. Tree	✓		✓	✓	1.5
Rad. Neig.	✓				1.4
GP	✓	✓		✓	840

The device that was used to run the Python codes and achieve the running times appearing in Table 5.5 has the following specifications. Device: LAPTOP, Processor: Intel(R) Core(TM) i9-10980HK CPU 2.40GHz 3.1 GHz, Installed RAM: 32.0 GB, System type: 64-bit operating system, x64-based processor.

The Transform Target Regressor and the Poisson Regressor showed for all four cross-sections (cs) MAPE below 10%. Then the Adaptive Boosting Regressor, the Decision Tree Regressor and the GP showed for three of the four cross-sections MAPE below 10% while the Radius Neighbor Regressor achieved to predict within this threshold only for one cross-section. However, the Decision Tree Regressor outperformed the Adaptive Boosting Regressor and the GP in computational efficiency as illustrated in the last column of the Table 5.5. One of the two reasons that the GP is highly computationally intense is that unlike the other regressors a distribution is predicted. Thus, the evaluation phase requires sampling of over that distribution in order to obtain the load-displacement curve. The second reason is that is the only of the evaluated regressors that takes into account the correlation of the outputs. Upon initial evaluation, the three most efficient algorithms considered overall appear to be the Transform Target Regressor, the Poisson Regressor, and the Decision Tree Regressor.

The square cross-section HF samples exhibit less variability compared to samples of other cross-sections. Thus, the task of comparing the regression solely based on high-fidelity samples with the regression based on high-fidelity and low-fidelity samples will

be performed on the square cross-section configuration. The regressors employed to evaluate the influence of low-fidelity samples in the accuracy of the energy absorption prediction, implementing the method illustrated in Figure 4.12, are the Transform Target Regressor, the Poisson Regressor, the Decision Tree regressor, and the Radius Neighbor Regressor. Executing the Adaptive Boosting Regressor and Gaussian Processes posed computational challenges, hindering their respective evaluation in Section 5.2. The relevant results of all the other regressors are documented in the following section.

5.2. Regression Solely Based on HF Samples vs. Based on HF and LF Samples

This phase of the analysis utilizes data from the square cross-section samples, as they demonstrate a more stable behavior compared to other cross-section shapes. This stability makes it easier to discern the impact of low-fidelity information during the training phase. The multi-output regressors employed include the Transform Target Regressor, the Poisson Regressor, the Decision Tree Regressor, and the Radius Neighbor Regressor. However, both the Adaptive Boosting Regressor and the Gaussian Process (GP) were deemed computationally infeasible to execute. Their computational intensity is evident from the comparison presented in Table 5.5.

Specifically the Transform Target Regressor and the Poisson Regressor, both showed efficiency as multi-output regressors not only for the square cross-section configuration but for all the four cross-sections so it is expected to be suitable for the implementation of this phase. The methodologies described in Chapter 4.4 are implemented in order to identify whether including low-fidelity along with high-fidelity samples can influence the accuracy of the energy absorption prediction.

The low-fidelity samples are generated as follows. The first low-fidelity samples generated correspond to samples identical with the high-fidelity samples, which in this case-study is 39 excluding the test sample that is used as reference. After that number, the rest of the low-fidelity samples are created randomly. The limits of the side length, the thickness and the tube length are bounded based on the variety of the existing high-fidelity data. Particularly, the side length is bounded from 30 mm to 60 mm and the thickness is bounded from 1 mm to 2.5 mm. The tube length remained as 200 mm since all the available high-fidelity samples for the squared cross-section have length 200 mm while the material can be either AA6060-T4 or AISI-316.

After the potential combinations are defined the samples increase in a random way based on the user input. For instance, when the user requests the generation of 100 samples then a 100×4 matrix is created which includes 39 low-fidelity samples identical to the high-fidelity samples and 61 randomly selected low-fidelity samples based on the aforementioned defined bounds. The dimension 4 of the matrix refers to the 4 features that can be defined for each sample: material, side length, thickness and tube length.

The regressors utilized do not take into account the different fidelities between the samples. In other words, the the high-fidelity samples and the low-fidelity samples are of

equal importance for the Regressor A (see Figure 4.12). For this reason, increasing excessively the percentage (%) of the low-fidelity samples would result in a low-fidelity model instead of a combination of a high-fidelity / low-fidelity model. An investigation in the range of 0-95% in the added low-fidelity samples is pursued, depending on when convergence behaviour is reached. The graphs are presented as follows: the y-axis represents the average MAPE of energy absorption (EA) in percentage (%) and the x-axis represents the percentage (%) of the low-fidelity samples added to the total number of samples (high-fidelity and low-fidelity together).

The trend of the average MAPE of the energy absorption for the two methodologies described in Chapter 4.4 when the Poisson Regressor is employed is illustrated in Figure 5.5. The red dash line, representing the high-fidelity model (HF Model), shows the average MAPE when the training is based solely on high-fidelity (HF) samples, implementing the methodology illustrated in Figure 4.11.

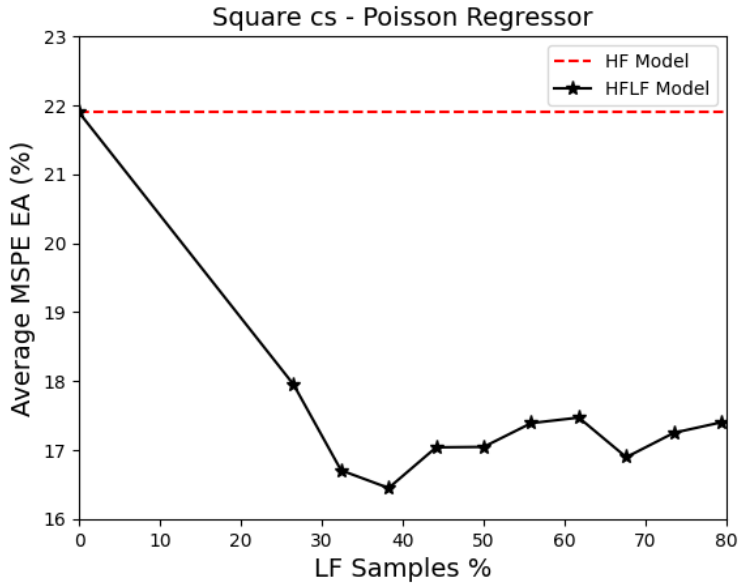


Figure 5.5: Regression Solely Based on HF samples vs. Regression Based on HF and LF samples with the Poisson MO Regr. - Square cs - HF Samples = 40

The average MAPE when the training is only relying to high-fidelity samples is 21.9%. The black curve on the other hand, representing the adjusted high-fidelity model (HFLF Model), shows the average MAPE when the training is based on high-fidelity (HF) samples augmented with low-fidelity (LF) samples, implementing the methodology illustrated in Figure 4.12.

The HF Model and the HFLF Model have common initial point as the two methods are identical when no LF samples are introduced in the training. When adding 26% of LF samples, a decrease of 18% is observed in the average MSPE of energy absorption (EA) compared to the HF Model. The maximum decrease is equal to 25% at 40% of LF added samples while after 45% of LF added samples the decrease is stabilized at approximately 21%. The HFLF Model when employing the Poisson Regressor produced

improved performance in predicting the energy absorption compared to the HF Model while it showed a maximum accuracy at 40% of LF added samples.

The behavior of the Transformed Target Regressor when is trained solely with high-fidelity samples and when is trained with high-fidelity and low-fidelity samples is illustrated in Figure 5.6. Again the HF Model and the HFLF Model have common initial point as the two methods are identical when no LF samples are utilized. When adding 25% of LF samples, a decrease of 1.6% is observed in the average MSPE of energy absorption (EA) compared to the HF Model. The maximum decrease is equal to 9.1% at 40% of LF added samples while after 46% of LF added samples the decrease is stabilized at approximately 8.1%. The HFLF Model when employing the Transformed Target Regressor produced improved performance in predicting the energy absorption compared to the HF Model while it showed a maximum accuracy at 45% of LF added samples.

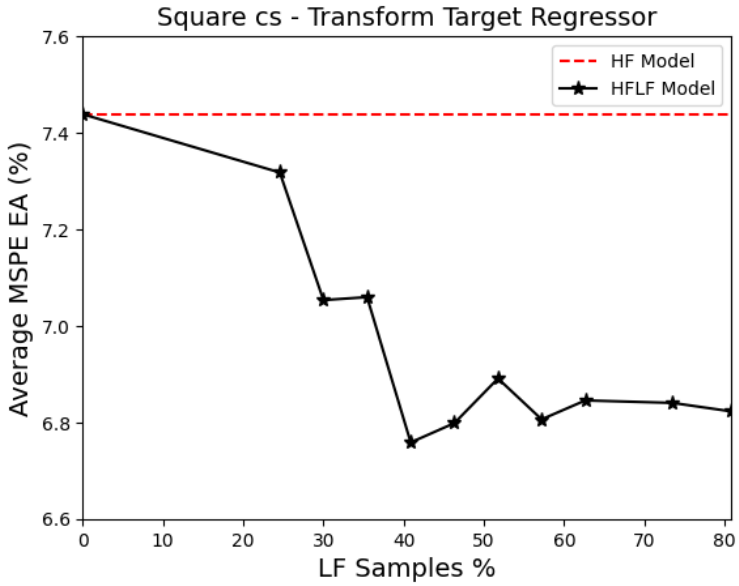


Figure 5.6: Regression Solely Based on HF samples vs. Regression Based on HF and LF samples with the Transformed Target MO Regr. - Square cs - HF Samples = 40

The same curves when the Radius Neighbor Regressor is employed are shown in Figure 5.7. An almost linear decrease is observed in the average MSPE EA (%) from 0 to 60% LF Samples. When no LF Samples are added the average MSPE is at 27.8% while the highest decrease is at 60% of LF added Samples where the average MSPE becomes 23.6% or there is a 14.4% decrease. Then the average MSPE reaches a steady state after 80% of LF Samples with a 13.3% decrease compared to the HF Model.

The Radius Neighbor Regressor efficiently shows that the presence of low-fidelity data can improve the predictions of the energy absorption result also shown with the Transformed Target Regressor and the Poisson Regressor. However this is not the case for the Decision Tree regressor which was deemed unsuitable for part of the current study. Detailed results for the Decision Tree regressor can be found in the Appendix C.

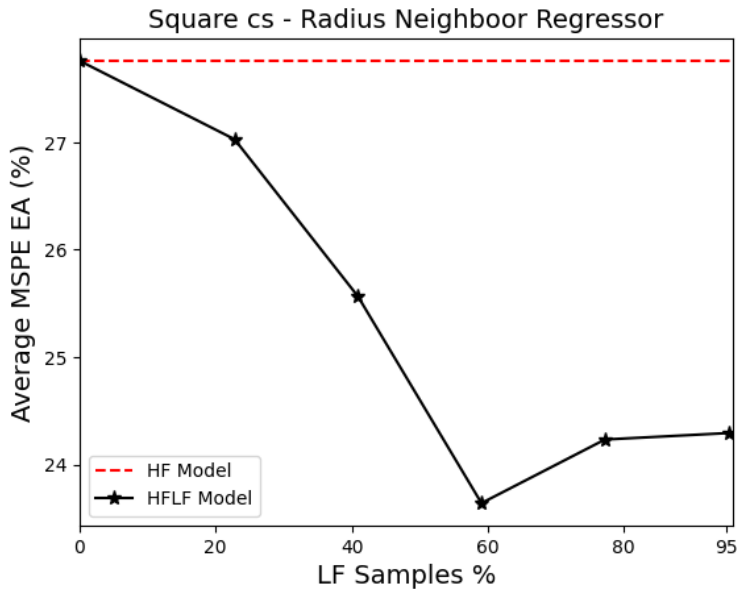


Figure 5.7: Regression Solely Based on HF samples vs. Regression Based on HF and LF samples with the Radius Neighbor MO Regr. - Square cs - HF Samples = 40

Observing the behavior of the Poisson Regressor, the Transformed Target Regressor and the Radius Neighbor Regressor in Figure 5.5, in Figure 5.6, and in Figure 5.7 respectively, several comments can be made. The maximum decrease compared to the HF Model is observed for the Poisson Regressor and the Transformed Target Regressor at approximately 40% of LF samples while for the Radius Neighbor Regressor the maximum decrease compared to the HF Model is achieved at 60% LF Samples.

This means that if the multi-output regressor had as input 100 samples in total, the maximum accuracy would be observed when 60 of them are of high-fidelity and 40 of them are of low-fidelity, referring to the Poisson Regressor and the Transformed Target Regressor. Regarding all the three regressors (see Figure 5.5, Figure 5.6, and Figure 5.7), by increasing the percentage (%) of LF Samples a decreasing trend of the HFLF Model compared to the HF Model is shown, suggesting the positive influence of the low-fidelity samples in predicting the energy absorption.

5.3. Discussion on the Results

The Transformed Target Regressor is able to predict the energy absorption more accurately between the evaluated regressors, independently on what training approach is utilized, the one presented in Figure 4.11 or Figure 4.12. This can be seen by comparing Figure 5.6, Figure 5.5 and Figure 5.7; where the Transformed Target Regressor achieves average MSPE of energy absorption in the range of 6.8-7.4%, the Poisson Regressor in the range of 16.4-21.9% and the Radius Neighbor Regressor in the range of 24.3-27.8%. This indicates that the logarithmic target transformation inherent in the Transformed Target Regressor is more suitable compared to the assumption of a Poisson distribution that the Poisson Regressor relies on and the radius neighbor theory that the Radius Neighbor Regressor relies on, for the present case-study focused on square cross-section configurations.

The ability of regressors to represent the structural behavior of tubular structures with load-displacement curves was evaluated across four different cross-sections for randomly picked samples. The Transform Target Regressor and the Poisson Regressor consistently exhibited MAPE below 10% for all four cross-sections. The motivation behind this threshold was based on studies as [34] that achieved error of 7% for similar predictions. Following closely, the Adaptive Boosting Regressor, the Decision Tree Regressor, and the Gaussian Process (GP) achieved MAPE below 10% for three out of the four cross-sections. However, the Radius Neighbor Regressor performed within this threshold only for one cross-section. The Decision Tree Regressor was the third best performed regressor since it outperformed the Adaptive Boosting Regressor and the GP in computational efficiency. The training samples range was 25 to 39 depending on the case under evaluation. This suggests that the surrogate should be able after being given a data-based input of the aforementioned range to predict efficiently the structural behavior of an unknown sample, which is a rather complex objective.

Two methodologies were discussed aiming to improve the prediction accuracy of energy absorption and were applied to tubular metallic structures with square cross-sectional configuration. In the first methodology, a surrogate model is trained solely with numerical data obtained from a high-fidelity model. In the second methodology, a low-fidelity model is introduced alongside the high-fidelity model to enhance prediction accuracy. Once created, the LF model enables generating results without the need for computationally demanding processes, considering various geometrical characteristics and material inputs. The combination of LF data and numerical data from the HF model serves as input for machine learning models in the second methodology. The primary objective is to enhance the prediction performance of energy absorption and compare the outcomes between the two methodologies. The task of comparing regressions based solely on high-fidelity samples with those incorporating both high-fidelity and low-fidelity samples was successfully executed using the Transform Target Regressor, the Poisson Regressor, and the Radius Neighbor Regressor for square cross-section samples.

Overall, as documented in Section 5.1 the Transformed Target Regressor showed capability of predicting the load-displacement curves with a minimum MAPE of 12.2% for the octagonal cross-section and a maximum MAPE 22.9% for the circular cross-section as shown in Figure 5.4 and Figure 5.2 respectively. It predicted the energy absorption, when employed as multi-output regressor, with a minimum MAPE of 2.9% for the square cross-section and with a maximum MAPE of 7.1% for the octagonal cross-section compared to the target sample as shown in Figure 5.1 and Figure 5.4 respectively. The Transformed Target Regressor and the Poisson Regressor were the only among the evaluated regressors that succeeded to predict the energy absorption for all cross-sections with a MAPE of EA below 10%, making them very competitive candidates to implement the methodologies presented in Section 4.4.

Indeed, when the Transformed Target Regressor was trained with high-fidelity samples an average MAPE of 7.4% was achieved. Introducing LF Samples decreased this value by 9.1% achieving an average MAPE EA of 6.8% at 40% LF added Samples. On the

other hand, when the Poisson Regressor was trained with high-fidelity samples an average MAPE of 21.9% was achieved. Introducing LF Samples decreased this value by 25% achieving an average MAPE EA of 16.4% again at 40% LF added Samples. Reduction in the average MAPE was also shown with the Radius Neighbor Regressor with a 14.4% decrease compared to the HF Model at 60% LF added Samples.

6 Conclusions

The development of meta-models is essential due to the significant computational resources needed for explicit finite element simulations and the challenges associated with experimental drop-tests. Accurately predicting the structural behavior enables also the potential of using surrogates in a crashworthiness optimization framework. In the current study, the aim is to tackle the inherent difficulties in crashworthiness assessment for helicopter structures and components when represented by surrogate models. These models do not solve any type of analysis such as FEM. Instead they learn from the data-based input given by the user.

A comparative analysis is conducted to assess the performance of two predictive approaches in determining the energy absorption of tubular metallic structures of four cross-sections. The first approach involves predicting the energy absorption directly as a scalar value, while the second approach predicts the load-displacement curve as a function, from which the energy absorption is computed as the integral of the curve. Both approaches utilized Machine Learning techniques and specifically regression models. The evaluation of the entire load-displacement curve was proved to be a valuable input for the surrogate models and they were imperative in representing the structural behavior of the structure. This would not be possible if the only information utilized was the energy absorption of each sample, proving the first hypothesis documented in Chapter 1.

The challenges of assessing crashworthiness performance include the limited information about the structural behavior when crushing so the need of introducing a low-fidelity model emerges. Three evaluated regressors proved that introducing low-fidelity samples in the training along with high-fidelity samples can increase the accuracy in predicting the energy absorption proving the second hypothesis of the thesis. However, incorporating low-fidelity samples did not increase the prediction accuracy of the load-displacement curve since the low-fidelity samples smooth the high-fidelity curves rather than trying to capture the sharp drops and peaks in their behavior. From that perspective other approach should be utilized, potentially employing completely different surrogate.

Another important conclusion is that the selection of the regressor is case-study related. A regressor that can provide good predictions for one configuration might not perform well with another configuration. Since the input of the methodologies are libraries of structural data and no solving is performed during the surrogate construction it is imperative to identify a suitable regressor to the available data.

These findings are promising in implementing the methodology in more complex assemblies such as helicopter subfloors. Machine Learning techniques appeared to be efficient in representing the structural behavior in crushing conditions while the accuracy of the predictions can be further improved by incorporating low-fidelity models along with high-fidelity models during the training process when there is scarcity of numerical or experimental data.

7 Recommendations

In future research initiatives, it is imperative to take into account more complex models as low-fidelity models if the intention is to incorporate low-fidelity samples in the training phase along with high-fidelity samples. This could be either an analytical derivation or samples that were derived from experiments of similar structural components to the ones under evaluation. This might contribute in increasing the accuracy of the prediction of the load-displacement curve instead of only the energy absorption of the crushing element.

The developed approaches primarily employ data-based models, meaning that all results are derived from a data library serving as input. This methodology enables the prediction of structural behaviors without necessitating substantial computational resources since structural solvers like FEM solvers are not utilized. Incorporating a simplified configuration solved with FEM could serve as an additional information source for the surrogate model. Although this addition may incur some computational expense, it would be mitigated by the simplification of the potential model. Thus, employing FEM can enhance the predictive capabilities of the structural behavior analysis.

Another future initiative is to implement the methodology to composite tubular structures or cruciforms which are more complicated structural elements, and confirm that indeed the regressors can provide accurate results in predicting the energy absorption. The next step is to implement the approaches of this project to even more complex structures, as that was the motivation of the study. However, conducting experiments such as helicopter subfloor drop-tests requires meticulous and long-lasting endeavors.

Additionally, incorporating a factor into the meta-models that distinguishes the importance between high-fidelity and low-fidelity samples could potentially yield even more improved results. In this scenario, the regressor would be able to prioritize the high-fidelity data over the low-fidelity data.

The Transform Target Regressor was chosen as the dual-target regressor for estimating both $P_{\max}$ and $x_{P_{\max}}$, as depicted in Figure 4.13. Even though this regressor outperformed in the present study, alternative estimators could be considered to assess potential enhancements in the accuracy of the overall methodology.

Supervised learning and specifically regressors were utilized for the surrogate models construction. A comparative analysis with other types of Machine Learning techniques or meta-models in general could give a better understanding whether there are margins in further improving the results and extending the findings of this research. Given the capability of regression algorithms in representing tubular structures accurately a design optimization can be also part of the future work. Employing the developed meta-models within optimization algorithms can lead to lightweight and crashworthy designs.

References

[1] D. F. Shanahan. *Basic Principles of Helicopter Crashworthiness*. Tech. rep. A. F. Rucker: U.S. Army Aeromedical Research Lab.

[2] A. H. Rao and K. Marais. "High risk occurrence chains in helicopter accidents". In: *Reliability Engineering & System Safety* 170 (2018), pp. 83–98. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2017.10.014.

[3] J. S. Churchwell, K. S. Zhang, and J. H. Saleh. "Epidemiology of helicopter accidents: Trends, rates, and covariates". In: *Reliability Engineering & System Safety* 180 (2018), pp. 373–384. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2018.08.007.

[4] Airbus. *A Statistical Analysis of Commercial Aviation Accidents 2023*. PDF document. 2023. URL: https://www.airbus.com/sites/g/files/jlcbta136/files/2023-03/A-Statistical-Analysis-of-Commercial-Aviation-Accidents-2023.pdf.

[5] D. Good. "U. S. Army contributions in crashworthiness". In: *SAFE Journal* 27.2 (1997), p. 138.

[6] FAA. *14 CFR Airworthiness Standards: Part 23 Normal, Utility, Acrobatic and Commuter Category Airplanes, Part 25 Transport Category Airplanes, Part 27 Normal Category Rotorcraft, Part 29 Transport Category Rotorcraft*. Federal Aviation Administration. URL: www.faa.gov.

[7] EASA. *Certification Specifications: CS 23 Normal, Utility, Aerobatic and Commuter Aeroplanes, CS 25 Large Aeroplanes, CS 27 Small Rotorcraft, CS 29 Large Rotorcraft*. European Aviation Safety Agency. URL: www.easa.europa.eu.

[8] X. Yang et al. "Crashworthy design and energy absorption mechanisms for helicopter structures: A systematic literature review". In: *Progress in Aerospace Sciences* 114 (Apr. 2020). ISSN: 03760421. DOI: 10.1016/j.paerosci.2020.100618.

[9] FAA Technical Center. *Rotorcraft Crashworthy Airframe and Fuel System Technology Development Program*. Tech. rep. U.S. Department of Transportation Federal Aviation Administration, Oct. 1994.

[10] A. Johnson et al. "Crash resistant composite subfloor structures for helicopters". In: *AGARD FVP-Symposium: Advances in Rotorcraft Technology, Ontario, Canada*. 1996.

[11] C. Bisagni. "Crashworthiness of helicopter subfloor structural components". In: *Aircraft Engineering and Aerospace Technology* 71.1 (1999), pp. 6–11. DOI: 10.1108/00022669910252088.

[12] H. Z. Zhou and Z. J. Wang. "Application of Foldcore Sandwich Structures in Helicopter Subfloor Energy Absorption Structure". In: *IOP Conference Series: Materials Science and Engineering* 248.1 (Oct. 2017), p. 012033. DOI: 10.1088/1757-899X/248/1/012033.

[13] G. L. Farley, R. L. Boitnott, and H. D. Carden. “Helicopter crashworthiness research program”. In: (1988).

[14] C. Kindervater, D. Kohlgruber, and A. Johnson. “Composite vehicle structural crashworthiness - A status of design methodology and numerical simulation techniques”. In: *International Journal of Crashworthiness* 4.2 (1999), pp. 213–230. DOI: 10.1533/cras.1999.0101.

[15] A. F. Johnson et al. “13 - Design and testing of crashworthy aerospace composite components”. In: *Polymer Composites in the Aerospace Industry (Second Edition)*. Ed. by P. Irving and C. Soutis. Second Edition. Woodhead Publishing Series in Composites Science and Engineering. Woodhead Publishing, 2020, pp. 371–414. ISBN: 978-0-08-102679-3. DOI: https://doi.org/10.1016/B978-0-08-102679-3.00013-7.

[16] C. Kindervater and H. Georgi. “Composite Strength and Energy Absorption as an Aspect of Structural Crash Resistance”. In: (1993).

[17] H. Kuhn. “Mechanical Testing and Evaluation”. In: *ASM Handbook*. Vol. 8. Sept. 2000, pp. 943–983. URL: http://www.asminternational.org.

[18] E. L. Fasanella, K. E. Jackson, and K. H. Lyle. “Finite Element Simulation of a Full-Scale Crash Test of a Composite Helicopter”. In: *Journal of The American Helicopter Society* (2002). DOI: 10.4050/jahs.47.156.

[19] C. Bisagni, L. Lanzi, and S. Ricci. “Size and topological optimization for crashworthiness design of helicopter subfloor”. In: (2002). DOI: 10.2514/6.2002-5484.

[20] T. Kermanidis et al. “Numerical Simulation Of Composite Structures Under Impact”. In: *WIT Transactions on the Built Environment* (1970). DOI: 10.2495/su980531.

[21] J. Frese and D. Nitschke. “Crushing Behaviour of Helicopter Subfloor Structures”. In: (1988).

[22] R. L. Boitnott and C. Kindervater. “Crashworthy design of helicopter composite airframe structures”. In: (1989).

[23] K. Hughes, R. Vignjevic, and J. Campbell. “Experimental observations of an 8 m/s drop test of a metallic helicopter underfloor structure onto a hard surface: Part 1”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 221 (2007), pp. 661–678. DOI: 10.1243/09544100JAERO214.

[24] P. Hajela and E. Lee. “Topological optimization of rotorcraft subfloor structures for crashworthiness considerations”. In: *Computers & Structures* 64.1 (1997). Computational Structures Technology, pp. 65–76. ISSN: 0045-7949. DOI: https://doi.org/10.1016/S0045-7949(96)00143-5.

[25] Marco Desiderio et al. “Crashworthiness of the Flying-V Aircraft Concept with Vertical Drop Test Simulations”. In: *ASIDIC* (2023). DOI: 10.31224/3034.

[26] K. E. Jackson et al. “Occupant Responses in a Full-Scale Crash Test of the Sikorsky ACAP Helicopter”. In: *Journal of The American Helicopter Society* 49 (2004), pp. 127–139.

[27] M. Votaw and J. Sen. “A system approach for designing a crashworthy helicopter using program KRASH”. In: (1984). DOI: 10.2514/6.1984-2448.

[28] D. Kohlgrüber and A. Kamoulakos. “Validation of Numerical Simulation of Composite Helicopter Sub-floor Structures Under Crash Loading”. In: (1998).

[29] C. Kindervater and E. Deletombe. *Composite Helicopter Structural Crashworthiness: Progress in Design and Crash Simulation Approaches*. 2000.

[30] C. Bisagni. “Energy absorption of riveted structures”. In: (1998). DOI: 10.1533/cras.1999.0100.

[31] C. Bisagni. “Crashworthiness of helicopter subfloor structures”. In: *International Journal of Impact Engineering* 27.10 (2002), pp. 1067–1082. ISSN: 0734-743X. DOI: https://doi.org/10.1016/S0734-743X(02)00015-5.

[32] M. A. Mccarthy and J. F. M. Wiggenraad. “Numerical investigation of a crash test of a composite helicopter subfloor structure”. In: (2001).

[33] C. Bisagni, L. Lanzi, and S. Ricci. “Optimization of Helicopter Subfloor Components Under Crashworthiness Requirements Using Neural Networks”. In: *Journal of Aircraft* (2002). DOI: 10.2514/2.2927.

[34] L. Lanzi, C. Bisagni, and S. Ricci. “Crashworthiness optimization of helicopter subfloor based on decomposition and global approximation”. In: *Structural and Multidisciplinary Optimization* 27 (July 2004), pp. 401–410. DOI: 10.1007/s00158-004-0394-z.

[35] L. Lanzi, L. M. L. Castelletti, and M. Anghileri. “Multi-objective optimisation of composite absorber shape under crashworthiness requirements”. In: *Composite Structures* 65.3 (2004), pp. 433–441. ISSN: 0263-8223. DOI: https://doi.org/10.1016/j.compstruct.2003.12.005.

[36] L. Lanzi, C. Mirandola, and L. M. Castelletti. “Crashworthiness Shape Optimization of Helicopter Subfloor Intersection Elements”. In: (2006).

[37] M. W. Joosten et al. “Improved Design Methods for Crashworthy Composite Helicopter Structures”. In: (2012).

[38] B. Prusty et al. “Experimental development and analysis of an active crushing element using pressurized composite tubes”. In: (2012).

[39] M. Anghileri. “Experimental and numerical assessment of helicopter subfloor crashworthiness”. In: (2012).

[40] R. Subbaramaiah et al. “A feasibility study for multi-material retrofittable energy absorbing structure for aged helicopter subfloor”. In: (2012), pp. 23–28.

[41] P. Astori, M. Zanella, and M. Bernardini. “Validation of Numerical Models of a Rotorcraft Crashworthy Seat and Subfloor”. In: *Aerospace* 7 (Dec. 2020), p. 174. DOI: 10.3390/aerospace7120174.

[42] A. Al-Fatlawi, K. Jármai, and G. Kovács. “Optimization of a Totally Fiber-Reinforced Plastic Composite Sandwich Construction of Helicopter Floor for Weight Saving, Fuel Saving and Higher Safety”. In: *Polymers* (2021). DOI: 10.3390/polym13162735.

[43] J. Fang et al. “On design optimization for structural crashworthiness and its state of the art”. In: *Structural and Multidisciplinary Optimization* 55 (Mar. 2017), pp. 1091–1119. DOI: 10.1007/s00158-016-1579-y.

[44] R. R. Mayer, Noboru Kikuchi, and R. A. Scott. “Application of topological optimization techniques to structural crashworthiness”. In: *International Journal for Numerical Methods in Engineering* (1996). DOI: 10.1002/(sici)1097-0207(19960430)39:8<1383::aid-nme909>3.0.co;2-3.

[45] Q. Shi and I. Hagiwara. “Optimal Design Method to Automobile Problems Using Holographic Neural Network’s Approximation”. In: *Japan Journal of Industrial and Applied Mathematics* (2000). DOI: 10.1007/bf03167370.

[46] A. G. Hanssen, M. Langseth, and O. S. Hopperstad. “Optimum design for energy absorption of square aluminium columns with aluminium foam filler”. In: *International Journal of Mechanical Sciences* (2001). DOI: 10.1016/s0020-7403(99)00108-3.

[47] H. Kim. “Analysis of crash response of aluminium foam-filled front side rail of a passenger car”. In: *International Journal of Crashworthiness* (2001). DOI: 10.1533/cras.2001.0172.

[48] H. Fang et al. “A comparative study of metamodeling methods for multiobjective crashworthiness optimization”. In: *Computers & Structures* 83.25 (2005), pp. 2121–2136. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2005.02.025.

[49] S. Hou et al. “Crashworthiness design for foam filled thin-wall structures”. In: *Materials & Design* (2009). DOI: 10.1016/j.matdes.2008.08.044.

[50] E. Acar and K. Solanki. “Improving the accuracy of vehicle crashworthiness response predictions using an ensemble of metamodels”. In: *International Journal of Crashworthiness* (2009). DOI: 10.1080/13588260802462419.

[51] H. R. Zarei and M. Kröger. “Crashworthiness optimization of empty and filled aluminum crash boxes”. In: *International Journal of Crashworthiness* (2007). DOI: 10.1080/13588260701441159.

[52] G. Sun et al. “Crashworthiness design for functionally graded foam-filled thin-walled structures”. In: *Materials Science and Engineering A-structural Materials Properties Microstructure and Processing* (2010). DOI: 10.1016/j.msea.2009.11.022.

[53] H. Yin et al. “Multiobjective crashworthiness optimization of functionally lateral graded foam-filled tubes”. In: *Materials & Design* (2013). DOI: 10.1016/j.matdes.2012.08.033.

[54] M. Costas et al. “A multi-objective surrogate-based optimization of the crashworthiness of a hybrid impact absorber”. In: *International Journal of Mechanical Sciences* (2014). DOI: 10.1016/j.ijmecsci.2014.07.002.

[55] J. Fang et al. “Crashworthiness design of foam-filled bitubal structures with uncertainty”. In: *International Journal of Non-linear Mechanics* (2014). DOI: 10.1016/j.ijnonlinmec.2014.08.005.

[56] A. Maia et al. “3 - Numerical optimization strategies for springback compensation in sheet metal forming”. In: *Computational Methods and Production Engineering*. Ed. by J. P. Davim. Woodhead Publishing Reviews: Mechanical Engineering Series. Woodhead Publishing, 2017, pp. 51–82. ISBN: 978-0-85709-481-0. DOI: https://doi.org/10.1016/B978-0-85709-481-0.00003-3.

[57] L. Lanzi, C. Bisagni, and S. Ricci. “Neural network systems to reproduce crash behavior of structural components”. In: *Computers & Structures* (2004). DOI: 10.1016/j.compstruc.2003.06.001.

[58] R. L. Hardy. “Multiquadric equations of topography and other irregular surfaces”. In: *Journal of Geophysical Research* 76 (1971), pp. 1905–1915. URL: https://api.semanticscholar.org/CorpusID:129508657.

[59] J. Forsberg and L. Nilsson. “Evaluation of response surface methodologies used in crashworthiness optimization”. In: *International Journal of Impact Engineering* 32.5 (2006). International Symposium on the Crashworthiness of Light-weight Automotive Structures, pp. 759–777. ISSN: 0734-743X. DOI: https://doi.org/10.1016/j.ijimpeng.2005.01.007.

[60] H. Yin et al. “Multiobjective crashworthiness optimization design of functionally graded foam-filled tapered tube based on dynamic ensemble metamodel”. In: *Materials & Design* 55 (2014), pp. 747–757. ISSN: 0261-3069. DOI: https://doi.org/10.1016/j.matdes.2013.10.054.

[61] N. Qiu et al. “Crashworthiness optimization with uncertainty from surrogate model and numerical error”. In: *Thin-Walled Structures* 129 (2018), pp. 457–472. ISSN: 0263-8231. DOI: https://doi.org/10.1016/j.tws.2018.05.002.

[62] P. Zhu, Y. Zhang, and G. Chen. “Metamodeling development for reliability-based design optimization of automotive body structure”. In: *Computers in Industry* 62.7 (2011), pp. 729–741. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2011.05.008.

[63] S. Xie et al. “Crashworthiness analysis of multi-cell square tubes under axial loads”. In: *International Journal of Mechanical Sciences* 121 (2017), pp. 106–118. ISSN: 0020-7403. DOI: https://doi.org/10.1016/j.ijmecsci.2016.12.005.

[64] M. Rocas et al. “Nonintrusive uncertainty quantification for automotive crash problems with VPS/Pamcrash”. In: *Finite Elements in Analysis and Design* 193 (2021), p. 103556. ISSN: 0168-874X. DOI: https://doi.org/10.1016/j.finel.2021.103556.

[65] H. Yin et al. “Crushing behavior and optimization of sheet-based 3D periodic cellular structures”. In: *Composites Part B: Engineering* 182 (2020), p. 107565. ISSN: 1359-8368. DOI: https://doi.org/10.1016/j.compositesb.2019.107565.

[66] D. C. Montgomery. *Design and Analysis of Experiments*. 1st. USA: Wiley, 1996.

[67] J. P. Boyd and F. Xu. “Divergence (Runge Phenomenon) for least-squares polynomial approximation on an equispaced grid and Mock–Chebyshev subset interpolation”. In: *Applied Mathematics and Computation* 210.1 (2009), pp. 158–168. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2008.12.087.

[68] J. Sacks et al. “Design and Analysis of Computer Experiments”. In: *Statistical Science* 4 (1989), pp. 409–423.

[69] F. L. Gao et al. “A time-space Kriging-based sequential metamodeling approach for multi-objective crashworthiness optimization”. In: *Applied Mathematical Modelling* 69 (2019), pp. 378–404. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2018.12.011.

[70] R. Yang, C. Tho, and L. Gu. “Recent Development in Multidisciplinary Design Optimization of Vehicle Structures”. In: (Sept. 2002). DOI: 10.2514/6.2002-5606.

[71] J. Gu. “An efficient multiple meta-model-based global optimization method for computationally intensive problems”. In: *Advances in Engineering Software* 152 (2021), p. 102958. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2020.102958.

[72] Z. HuaZhi and W. ZhiJin. “Study and Optimization of Helicopter Subfloor Energy Absorption Structure with Foldcore Sandwich Structures”. In: *IOP Conference Series: Materials Science and Engineering* 269 (Nov. 2017), p. 012022. DOI: 10.1088/1757-899X/269/1/012022.

[73] S. Nayak. “Chapter one - Introduction to optimization”. In: *Fundamentals of Optimization Techniques with Algorithms*. Ed. by Sukanta Nayak. Academic Press, 2020, pp. 1–7. ISBN: 978-0-12-821126-7. DOI: https://doi.org/10.1016/B978-0-12-821126-7.00001-2.

[74] P. Astori and F. Impari. “Crash response optimisation of helicopter seat and subfloor”. In: *International Journal of Crashworthiness* 18 (Dec. 2013), pp. 570–578. DOI: 10.1080/13588265.2013.815602.

[75] L. Lanzi et al. “Optimisation of Energy Absorbing Subsystems for Helicopter Vertical Crashes”. In: (2006).

[76] L. Grippo and M. Sciandrone. *Introduction to Methods for Nonlinear Optimization*. 1st ed. UNITEXT. Springer Cham, 2023. DOI: 10.1007/978-3-031-26790-1.

[77] J. S. Arora. “Chapter 13 - More on Numerical Methods for Constrained Optimum Design”. In: *Introduction to Optimum Design (Fourth Edition)*. Ed. by J. S. Arora. Fourth Edition. Boston: Academic Press, 2017, pp. 555–599. ISBN: 978-0-12-800806-5. DOI: https://doi.org/10.1016/B978-0-12-800806-5.00013-5.

[78] A. Farhang-Mehr et al. “Bayesian Approximation-Assisted Optimization Applied to Crashworthiness Design of a Pickup Truck”. In: *Design Automation Conference* (2003). DOI: 10.1115/detc2003/dac-48755.

[79] L. Shi et al. “Adaptive sampling-based RBDO method for vehicle crashworthiness design using Bayesian metric and stochastic sensitivity analysis with independent random variables”. In: *International Journal of Crashworthiness* 18.4 (2013), pp. 331–342. DOI: 10.1080/13588265.2013.793262.

[80] X. Wang and L. Shi. "A new metamodel method using Gaussian process based bias function for vehicle crashworthiness design". In: *International Journal of Crashworthiness* 19.3 (2014), pp. 311–321. DOI: 10.1080/13588265.2014.898932.

[81] X. Du et al. "A radial-basis function mesh morphing and Bayesian optimization framework for vehicle crashworthiness design". In: *Structural And Multidisciplinary Optimization* (2023). DOI: 10.1007/s00158-023-03496-x.

[82] K. Liu et al. "Design for Crashworthiness of Categorical Multimaterial Structures Using Cluster Analysis and Bayesian Optimization". In: *Journal of Mechanical Design* 141 (Sept. 2019), pp. 1–44. DOI: 10.1115/1.4044838.

[83] S. Anand, R. Alderliesten, and S. G. P. Castro. "Crashworthiness in preliminary design: Mean crushing force prediction for closed-section thin-walled metallic structures". In: *International Journal of Impact Engineering* (2024), p. 104946. ISSN: 0734-743X. DOI: https://doi.org/10.1016/j.ijimpeng.2024.104946.

[84] A. G. Mamalis et al. "Finite element simulation of the axial collapse of metallic thin-walled tubes with octagonal cross-section". In: *Thin-Walled Structures* 41.10 (2003), pp. 891–900. ISSN: 0263-8231. DOI: https://doi.org/10.1016/S0263-8231(03)00046-6.

[85] S. Tabacu and C. Ducu. "An analytical solution for the estimate of the mean crushing force of structures with polygonal and star-shaped cross-sections subjected to axial load". In: *International Journal of Mechanical Sciences* 161-162 (2019), p. 105010. ISSN: 0020-7403. DOI: https://doi.org/10.1016/j.ijmecsci.2019.105010.

[86] Y. Liu and M. L. Day. "Development of simplified finite element models for straight thin-walled tubes with octagonal cross section". In: *International Journal of Crashworthiness* 12.5 (2007), pp. 503–508. DOI: 10.1080/13588260701483557.

[87] D. F. Shanahan. *Human tolerance and crash survivability*. Tech. rep. RTO-EN-HFM-113, 2004. URL: https://www.sto.nato.int/publications/STO%20Educational%20Notes/RTO-EN-HFM-113/EN-HFM-113-06.pdf.

[88] W. Abramowicz and N. Jones. "Dynamic axial crushing of square tubes". In: *International Journal of Impact Engineering* 2.2 (1984), pp. 179–208. ISSN: 0734-743X. DOI: https://doi.org/10.1016/0734-743X(84)90005-8.

[89] A. A. Singace, H. Elsobky, and T. Y. Reddy. "On the eccentricity factor in the progressive crushing of tubes". In: *International Journal of Solids and Structures* 32.24 (1995), pp. 3589–3602. ISSN: 0020-7683. DOI: https://doi.org/10.1016/0020-7683(95)00020-B.

[90] X. Zhang and H. Zhang. "Theoretical and numerical investigation on the crush resistance of rhombic and kagome honeycombs". In: *Composite Structures* 96 (2013), pp. 143–152. ISSN: 0263-8223. DOI: https://doi.org/10.1016/j.compstruct.2012.09.028.

[91] I. H. Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions". In: *SN Computer Science* 2 (2021), p. 160. ISSN: 2661-8907. DOI: 10.1007/s42979-021-00592-x.

[92] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[93] P. Saves et al. “SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes”. In: *Advances in Engineering Sofware* 188 (2024), p. 103571. DOI: https://doi.org/10.1016/j.advengsoft.2023.103571.

[94] Q. Lv et al. “Modeling hydrogen solubility in water: Comparison of adaptive boosting support vector regression, gene expression programming, and cubic equations of state”. In: *International Journal of Hydrogen Energy* 57 (2024), pp. 637–650. ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2023.12.227.

[95] D. Chakraborty, A. Ghosh, and S. Saha. “Chapter 2 - A survey on Internet-of-Thing applications using electroencephalogram”. In: *Emergence of Pharmaceutical Industry Growth with Industrial IoT Approach*. Ed. by Valentina Emilia Balas, Vijender Kumar Solanki, and Raghvendra Kumar. Academic Press, 2020, pp. 21–47. ISBN: 978-0-12-819593-2. DOI: https://doi.org/10.1016/B978-0-12-819593-2.00002-9.

[96] G. D. Sad, L. D. Terissi, and J. C. Gómez. “Chapter 4 - Disambiguating Conflicting Classification Results in AVSR”. In: *Intelligent Speech Signal Processing*. Ed. by Nilanjan Dey. Academic Press, 2019, pp. 55–80. ISBN: 978-0-12-818130-0. DOI: https://doi.org/10.1016/B978-0-12-818130-0.00004-0.

[97] A. Subasi. “Chapter 3 - Machine learning techniques”. In: *Practical Machine Learning for Data Analysis Using Python*. Ed. by Abdulhamit Subasi. Academic Press, 2020, pp. 91–202. ISBN: 978-0-12-821379-7. DOI: https://doi.org/10.1016/B978-0-12-821379-7.00003-5.

[98] J. D. Elhai, P. S. Calhoun, and J. D. Ford. “Statistical procedures for analyzing mental health services data”. In: *Psychiatry Research* 160.2 (2008), pp. 129–136. ISSN: 0165-1781. DOI: https://doi.org/10.1016/j.psychres.2007.07.003.

[99] W. Wu and T. D. Little. “Quantitative Research Methods”. In: *Encyclopedia of Adolescence*. Ed. by B. Bradford Brown and Mitchell J. Prinstein. San Diego: Academic Press, 2011, pp. 287–297. ISBN: 978-0-12-373951-3. DOI: https://doi.org/10.1016/B978-0-12-373951-3.00034-X.

[100] Y. Xie. “Chapter 9 - Data Analytics for Safety Applications”. In: *Data Analytics for Intelligent Transportation Systems*. Ed. by Mashrur Chowdhury, Amy Apon, and Kakan Dey. Elsevier, 2017, pp. 215–239. ISBN: 978-0-12-809715-1. DOI: https://doi.org/10.1016/B978-0-12-809715-1.00009-2.

[101] DataCamp. *AdaBoost Classifier Tutorial*. Accessed: February 20, 2024. 2018. URL: https://www.datacamp.com/tutorial/adaboost-classifier-python.

[102] M. Chalela et al. “GriSPy: A Python package for fixed-radius nearest neighbors search”. In: *Astronomy and Computing* 34 (2021), p. 100443. ISSN: 2213-1337. DOI: https://doi.org/10.1016/j.ascom.2020.100443.

[103] Z. Wang et al. “Entropy and gravitation based dynamic radius nearest neighbor classification for imbalanced problem”. In: *Knowledge-Based Systems* 193 (2020), p. 105474. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2020.105474.

[104] Y. Zhu, Z. Wang, and D. Gao. “Gravitational fixed radius nearest neighbor for imbalanced problem”. In: *Knowledge-Based Systems* 90 (2015), pp. 224–238. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2015.09.015.

[105] Y. Shi et al. “Chapter Two - Primer on artificial intelligence”. In: *Mobile Edge Artificial Intelligence*. Ed. by Yuanming Shi et al. Academic Press, 2022, pp. 7–36. ISBN: 978-0-12-823817-2. DOI: https://doi.org/10.1016/B978-0-12-823817-2.00011-5.

[106] S. Weglarczyk. “Kernel density estimation and its application”. In: *ITM Web of Conferences* 23 (Jan. 2018), p. 00037. DOI: 10.1051/itmconf/20182300037.

[107] T. Hastie. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Jan. 2009. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7.

[108] Y. Yang, S. S. Farid, and N. F. Thornhill. “Prediction of biopharmaceutical facility fit issues using decision tree analysis”. In: ed. by Andrzej Kraslawski and Ilkka Turunen. Vol. 32. Computer Aided Chemical Engineering. Elsevier, 2013, pp. 61–66. DOI: https://doi.org/10.1016/B978-0-444-63234-0.50011-7.

[109] L. Brevault, M. Balesdent, and A. Hebbal. “Overview of Gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems”. In: *Aerospace Science and Technology* 107 (2020), p. 106339. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2020.106339.

[110] A. C. Ogren et al. “Gaussian process regression as a surrogate model for the computation of dispersion relations”. In: *Computer Methods in Applied Mechanics and Engineering* 420 (2024), p. 116661. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.116661.

[111] E. Bonilla, K. Chai, and C. Williams. “Multi-task Gaussian Process Prediction”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[112] R. G. Mortimer and S. M. Blinder. “Chapter 7 - Integral Calculus”. In: *Mathematics for Physical Chemistry (Fifth Edition)*. Ed. by Robert G. Mortimer and S. M. Blinder. Fifth Edition. Elsevier, 2024, pp. 71–85. ISBN: 978-0-443-18945-6. DOI: https://doi.org/10.1016/B978-0-44-318945-6.00012-0.

A Python Codes

```
"""
Code to generate results shown in section 5.2
"""
import numpy as np
import utils_design_vector as udv
import LowFidelityModel as lf
import utils_plot as up
import utils_data as ud
import warnings
import matplotlib.pyplot as plt
from scipy.integrate import simps
from sklearn.metrics import mean_absolute_percentage_error
warnings.filterwarnings("ignore", category=RuntimeWarning)
warnings.filterwarnings("ignore", category=DeprecationWarning)

method_list = ['Transform Target Regressor', 'Poisson Regressor', 'RadNeig
    .','DecTree','AdaBoost'] # , 'GP'] #
method = method_list[2]
predict =  'multi-output'
numel_samples = 40 # fea samples

evaluations = 800 - numel_samples + 1
total_samples = (evaluations + numel_samples - 1) + numel_samples - 1 #
    solve the eq above wrt the integer and add the training samples
step = 5
xj = np.linspace(0, evaluations, num=step+1)
evals = np.zeros_like(xj)

for j in range(len(evals)):
    mape_Pmax, mape_disp, mape_mean, mape_area = [np.zeros(numel_samples)
        for _ in range(4)]

    for index_sample in range(numel_samples):
        X_train_fea, y_train_fea, X_test_fea, y_test_fea, x_array = lf.
            get_fea_data (index_sample)
        X_train_dummy, y_train_dummy = lf.get_dummy_data_with_random_ds(
            int(xj[j]),    X_train_fea)
        if X_train_dummy==[]:
            X_train = X_train_fea
            y_train = y_train_fea
        else:
            X_train = np.vstack((X_train_fea, X_train_dummy))
            y_train = np.vstack((y_train_fea, y_train_dummy))

        y_pred = curves_train_dummy =udv.regression_model_advanced(method,
            X_test_fea, X_train, y_train,    predict)

        y_test_fea_max_hf = np.max(y_test_fea[0, :200])  # Pmax in kN
        index_hf = np.argmax(y_test_fea[0, :200])
        mean_hf = np.mean(y_test_fea)
        area_hf = simps(y_test_fea / 1000, x_array)
```

```
        y_test_fea_max_lf = np.max(curves_train_dummy[0, :200])  # Pmax in kN
        index_lf = np.argmax(curves_train_dummy[0, :200])
        mean_lf = np.mean(curves_train_dummy)
        area_lf = simps(curves_train_dummy / 1000, x_array)
        mape_area[index_sample] = mean_absolute_percentage_error(np.array([area_hf]), np.array([area_lf])) * 100

    evals[j] = np.mean(mape_area)

    if j==0:
        print(f"added samples: {int(xj[j]):2.0f}, average mape area: {np.mean(mape_area):2.2f} ")
    else:
        print(f"added samples: {int(xj[j])+numel_samples-1:2.0f}, average mape area: {np.mean(mape_area):2.2f} ")

xj[1:] += numel_samples - 1
xj = xj / (total_samples * 0.01)
plt.plot(xj, evals,'k-*', markersize=8, label='HFLF Model')
plt.axhline(y=evals[0], color='r', linestyle='--', label='HF Model')
plt.legend()
plt.xlabel('LF Samples %', fontsize=14)
plt.ylabel('Average MSPE EA(%)' , fontsize=14)
plt.xlim(min(xj), max(xj))
plt.title('Square cs - Multi-output regressor: {}'.format(method), fontsize=14)
plt.show()
"""
LowFidelityModel.py
"""
from matplotlib import pyplot as plt
import numpy as np
from sklearn.compose import TransformedTargetRegressor
from sklearn.linear_model import LinearRegression
import utils_data as ud
import utils_design_vector as udv
from scipy.integrate import simps
from sklearn.metrics import mean_absolute_percentage_error
import random

def get_fea_data(index_sample):
    predict =  'multi-output'
    Include_integral_in_the_training = False
    list_test, list_train, test_train_list = ud.select_test_specimen(index_sample)

    X_train, y_train, X_test, y_test, x_array, = udv.assign_arrays(list_test, list_train, test_train_list,
                                                   Include_integral_in_the_training, predict)
    y_train = y_train / 1000 # kN
    y_test = y_test / 1000 # kN
    return X_train, y_train, X_test, y_test, x_array

def get_dummy_data_with_random_ds(num_samples, X_train_fea):
    samples = []
```

```
94      if num_samples == 0:
95          X_train_dummy = [] #X_train_fea
96      else:
97          for _ in range(num_samples):
98              x1 = random.choice([1, 2])
99              x2 = random.uniform(30, 60)
100             x3 = random.uniform(1, 2.5)
101             x4 = 200
102             samples.append((x1, x2, x3, x4))
103
104         X_train_dummy = np.vstack((X_train_fea, samples))
105
106     # Identify how many fea samples we have in total
107     dummy_sample = 0
108     _, _, dummy_list = ud.select_test_specimen(dummy_sample)
109     #
110     # In the mspe_errors_array we will include the mspe of each comparison
            with the test (or each itteration) so as to compare in every itt
111     # with the specimen that is considered as test or in other words the
            one that was not included in the training process.
112     mspe_errors_array = np.zeros(len(dummy_list))
113     #for index_sample in range(len(dummy_list)):
114     index_sample = 0
115     # Select which of the 40 in total specimens will be excluded from the
            training. By setting this as variable
116     # index_sample we make sure that in each itteration the test specimen
            (the one that we will compare the prediction with)
117     # will always be out of the training process.
118     list_test, list_train, test_train_list = ud.select_test_specimen(
            index_sample)
119     X_train, Pmax_train, X_test, Pmax_test, x_array, curves_test,
            curves_train = udv.assign_Pmax_and_x_at_Pmax(list_test,list_train,
            test_train_list) # Pmax in kN
120
121     # Train excluding the index_sample
122     model = TransformedTargetRegressor(regressor=LinearRegression(), func=
            np.log, inverse_func=np.exp)
123     model.fit(X_train, Pmax_train)
124
125     # Create a matrix Pmean_Pmax_PmaxIndex_pred that contains 3 info:
            Pmean from Shreyas model Pmax from machine learning model and
            PmaxIndex
126     # also from from machine learning model. This array will be the input
            to construct the low fidelity curves.
127
128     curves_train_dummy = np.zeros((len(X_train_dummy),1000))
129     for ikat in range(len(X_train_dummy)):
130         Pmean = ud.analytical(X_train_dummy[ikat,:])  # Pmean in kN
131         beta  = model.predict([X_train_dummy[ikat,:] ]) # Pmax in kN and
                index of Pmax
132         Pmax  = beta[0][0]
133         index = int(round(beta[0][1]))
134         #
135         Pmax2  = beta[0][2]
136         index2 = int(round(beta[0][3]))
137         if index == 0 or index == 1:
```

```
            index = 2
        #array1 = np.linspace(2*Pmax, Pmax, index)
        #array2 = np.linspace(Pmean, Pmean, 1000-index-index2)
        #array3 = np.linspace(Pmean, Pmax2, index2)
        #
        array1 = np.linspace(Pmax/100    , Pmax, index)
        array2 = np.linspace(Pmean, Pmean, 1000-index)
        curves_train_dummy[ikat,:] = np.hstack((array1, array2))#, array3)
            )
    #print(curves_train_dummy[:,0:4])

    # Nothing important here. We choose whether we would like to compare
       the predictions with the train data (multiple sets of data) or
    # with the test data (1 set of data). We choose the test data so as to
        compare with the specimen that was not included in the
    # training process.
    X_train_or_X_test = X_test
    curves_train_or_curves_test = curves_test # this is in kN

    # Fea model.  Construct the array that will contain the areas below
       the fea curves
    areas_train = np.zeros((len(X_train_or_X_test)))
    for pkat in range(len(X_train_or_X_test)):
        areas_train[pkat] = simps(curves_train_or_curves_test[pkat, :],
            x_array)
    # Low fidelity model. Construct the array that will contain the areas
       below the low fidelity curves
    areas_train_dummy = np.zeros((len(X_train_dummy)))
    for hkat in range(len(X_train_dummy)):
        areas_train_dummy[hkat] = simps(curves_train_dummy[hkat,:],
            x_array)
    for hkat in range(len(X_train_dummy)):
        for pkat in range(len(X_train_or_X_test)):
            if np.array_equal(X_train_dummy[hkat, :], X_train_or_X_test[
                pkat,:]):
                mspe = mean_absolute_percentage_error(np.array([
                    areas_train[pkat]]), np.array([areas_train_dummy[hkat
                    ]]))
                mspe_errors_array[index_sample] = mspe * 100
                #print(f"FEA area: {areas_train[pkat]:8.2f} J, low
                    fidelity area with pred Pmax and x @ Pmax: {
                    areas_train_dummy[hkat]:8.2f} J, mspe error: {mspe
                    *100:4.2f} %")
                if False:
                    plt.plot(x_array, curves_train_or_curves_test[pkat,
                        :],'k-*', markersize=1, label='FEA') ###
                    plt.tick_params(axis='both', which='major', labelsize
                        =14)
                    plt.plot(x_array, curves_train_dummy[hkat,:], '-*',
                        markersize=1, color='green',label='LF model' )  #
                        Plot the total array ith #a blue curve
                    plt.legend()
                    plt.xlabel('Displacement (mm)', fontsize=14)
                    plt.ylabel('Load (kN)' , fontsize=14)
                    plt.title('Sample: {}'.format(X_train_or_X_test[pkat])
                        , fontsize=14) ###
```

```
176                     plt.xlim(0, max(x_array))
177                     # Adding text box with print information
178                     textstr = '\n'.join((
179                         f'FEA area: {areas_train[pkat]:8.2f} J',
180                         f'LF    area: {areas_train_dummy[hkat]:8.2f} J',
181                         f'mspe error: {mspe*100:4.2f} %'))
182                     plt.text(0.57, 0.05, textstr, transform=plt.gca().
                            transAxes,
183                              fontsize=14, verticalalignment='bottom', bbox=
                                 dict(boxstyle='round', facecolor='wheat',
                                 alpha=0.5))
184                     plt.show()
185     return X_train_dummy, curves_train_dummy
186     #print(f"Mean of Mean Squared Per. errors: {np.mean(mspe_errors_array)
            :2.2f} %")
187
188
189 def lfvalidation(index_sample, X_test_fea):
190     # Select which of the 40 in total specimens will be excluded from the
            training. By setting this as variable
191     # index_sample we make sure that in each itteration the test specimen
            (the one that we will compare the prediction with)
192     # will always be out of the training process.
193     list_test, list_train, test_train_list = ud.select_test_specimen(
            index_sample)
194     X_train, Pmax_train, X_test, Pmax_test, x_array, curves_test,
            curves_train = udv.assign_Pmax_and_x_at_Pmax(list_test,list_train,
            test_train_list) # Pmax in kN
195     # Train excluding the index_sample
196     model = TransformedTargetRegressor(regressor=LinearRegression(), func=
            np.log, inverse_func=np.exp)
197     model.fit(X_train, Pmax_train[:,0:2])
198
199     # Create a matrix Pmean_Pmax_PmaxIndex_pred that contains 3 info:
            Pmean from Shreyas model Pmax from machine learning model and
            PmaxIndex
200     # also from from machine learning model. This array will be the input
            to construct the low fidelity curves.
201
202     curves_train_dummy = np.zeros((1,1000))
203     Pmean = ud.analytical( X_test_fea[0,:] )  # Pmean in kN
204     beta  = model.predict([X_test_fea[0,:] ]) # Pmax in kN and index of
            Pmax
205     Pmax  =           beta[0][0]
206     index = int(round(beta[0][1]))
207
208     if index == 0 or index == 1:
209         print('oups')
210         index = 2
211
212     array1 = np.linspace(Pmax/100    , Pmax, index)
213     array2 = np.linspace(Pmean, Pmean, 1000-index)
214     curves_train_dummy[0,:] = np.hstack((array1, array2))
215
216     return curves_train_dummy
217
```

```
218 def get_dummy_data(step):
219     def create_X_train_dummy(range_start, range_end, step,
        second_column_values):
220         array = []
221         for first_column in [1, 2]:
222             for second_column in second_column_values:
223                 for third_column in np.arange(range_start, range_end +
                    step, step):
224                     array.append([first_column, second_column,
                        third_column, 200])
225         return np.array(array)
226 
227     # Define the range and step size for the side length in the
        X_train_dummy
228     second_column_range_start = 30
229     second_column_range_end = 60
230     second_column_step = 5
231     # Define the range and step size for the thickness in the
        X_train_dummy
232     third_column_range_start = 1
233     third_column_range_end = 2.5
234     third_column_step = step
235     # Generate the X_train_dummy
236     second_column_values = np.arange(second_column_range_start,
        second_column_range_end + second_column_step, second_column_step)
237     X_train_dummy = create_X_train_dummy(third_column_range_start,
        third_column_range_end, third_column_step, second_column_values)
238     # Identify how many fea samples we have in total
239     dummy_sample = 0
240     _, _, dummy_list = ud.select_test_specimen(dummy_sample)
241     #
242     # In the mspe_errors_array we will include the mspe of each comparison
         with the test (or each itteration) so as to compare in every itt
243     # with the specimen that is considered as test or in other words the
        one that was not included in the training process.
244     mspe_errors_array = np.zeros(len(dummy_list))
245     #for index_sample in range(len(dummy_list)):
246     index_sample = 0
247     # Select which of the 40 in total specimens will be excluded from the
        training. By setting this as variable
248     # index_sample we make sure that in each itteration the test specimen
        (the one that we will compare the prediction with)
249     # will always be out of the training process.
250     list_test, list_train, test_train_list = ud.select_test_specimen(
        index_sample)
251     X_train, Pmax_train, X_test, Pmax_test, x_array, curves_test,
        curves_train = udv.assign_Pmax_and_x_at_Pmax(list_test,list_train,
        test_train_list) # Pmax in kN
252     # Train excluding the index_sample
253     model = TransformedTargetRegressor(regressor=LinearRegression(), func=
        np.log, inverse_func=np.exp)
254     model.fit(X_train, Pmax_train)
255     # Create a matrix Pmean_Pmax_PmaxIndex_pred that contains 3 info:
        Pmean from Shreyas model Pmax from machine learning model and
        PmaxIndex
256     # also from from machine learning model. This array will be the input
```

```
        to construct the low fidelity curves.
257
258     curves_train_dummy = np.zeros((len(X_train_dummy),1000))
259     for ikat in range(len(X_train_dummy)):
260         Pmean = ud.analytical(X_train_dummy[ikat,:])  # Pmean in kN
261         beta  = model.predict([X_train_dummy[ikat,:] ]) # Pmax in kN and
                index of Pmax
262         Pmax  = beta[0][0]
263         index = int(round(beta[0][1]))
264         #
265         Pmax2  = beta[0][2]
266         index2 = int(round(beta[0][3]))
267         if index == 0 or index == 1:
268             index = 2
269         array1 = np.linspace(Pmax/100    , Pmax, index)
270         array2 = np.linspace(Pmean, Pmean, 1000-index)
271         curves_train_dummy[ikat,:] = np.hstack((array1, array2))#, array3)
                )
272     # We choose whether we would like to compare the predictions with the
        train data (multiple sets of data) or
273     # with the test data (1 set of data). We choose the test data so as to
         compare with the specimen that was not included in the
274     # training process.
275     X_train_or_X_test = X_test
276     curves_train_or_curves_test = curves_test # this is in kN
277     # Fea model.  Construct the array that will contain the areas below
        the fea curves
278     areas_train = np.zeros((len(X_train_or_X_test)))
279     for pkat in range(len(X_train_or_X_test)):
280         areas_train[pkat] = simps(curves_train_or_curves_test[pkat, :],
                x_array)
281     # Low fidelity model. Construct the array that will contain the areas
        below the low fidelity curves
282     areas_train_dummy = np.zeros((len(X_train_dummy)))
283     for hkat in range(len(X_train_dummy)):
284         areas_train_dummy[hkat] = simps(curves_train_dummy[hkat,:],
                x_array)
285     for hkat in range(len(X_train_dummy)):
286         for pkat in range(len(X_train_or_X_test)):
287             if np.array_equal(X_train_dummy[hkat, :], X_train_or_X_test[
                    pkat,:]):
288                 mspe = mean_absolute_percentage_error(np.array([
                        areas_train[pkat]]), np.array([areas_train_dummy[hkat
                        ]]))
289                 mspe_errors_array[index_sample] = mspe * 100
290                 #print(f"FEA area: {areas_train[pkat]:8.2f} J, low
                        fidelity area with pred Pmax and x @ Pmax: {
                        areas_train_dummy[hkat]:8.2f} J, mspe error: {mspe
                        *100:4.2f} %")
291                 if False:
292                     plt.plot(x_array, curves_train_or_curves_test[pkat,
                            :],'k-*', markersize=1, label='FEA') ###
293                     plt.tick_params(axis='both', which='major', labelsize
                            =14)
294                     plt.plot(x_array, curves_train_dummy[hkat,:], '-*',
                            markersize=1, color='green',label='LF␣model' )  #
```

```
                        Plot the total array ith #a blue curve
295                     plt.legend()
296                     plt.xlabel('Displacement␣(mm)', fontsize=14)
297                     plt.ylabel('Load␣(kN)' , fontsize=14)
298                     plt.title('Sample:␣{}'.format(X_train_or_X_test[pkat])
                            , fontsize=14) ###
299                     plt.xlim(0, max(x_array))
300                     # Adding text box with print information
301                     textstr = '\n'.join((
302                         f'FEA␣area:␣{areas_train[pkat]:8.2f}␣J',
303                         f'LF␣␣␣area:␣{areas_train_dummy[hkat]:8.2f}␣J',
304                         f'mspe␣error:␣{mspe*100:4.2f}␣%'))
305                     plt.text(0.57, 0.05, textstr, transform=plt.gca().
                            transAxes,
306                              fontsize=14, verticalalignment='bottom', bbox=
                                 dict(boxstyle='round', facecolor='wheat',
                                 alpha=0.5))
307                     plt.show()
308     return X_train_dummy, curves_train_dummy
309     #print(f"Mean of Mean Squared Per. errors: {np.mean(mspe_errors_array)
            :2.2f} %")
310 """
311 utils.data.py
312 """
313 import numpy as np
314
315 def analytical(X_test):
316     mat, c, h = X_test[0], X_test[1], X_test[2]
317     Nc = 4   # Number of corners
318
319     Xcoef, Ycoef, Zcoef = 15.91, 1.03, 0.15 # Polygon (INEXT)
320
321     sigma_0_AA6060_T4 = 151 # (MPa)
322     sigma_0_AISI_316  = 624 # (MPa)
323
324     sigma_y_AA6060_T4 = 76  # (MPa)
325     sigma_y_AISI_316  = 467 # (MPa)
326
327     kapa_AA6060_T4 = 0.73
328     kapa_AISI_316  = 0.77
329
330     M0_AA6060_T4 = (sigma_0_AA6060_T4 * (h**2)) / 4
331     M0_AISI_316  = (sigma_0_AISI_316  * (h**2)) / 4
332
333     Pm_AA6060_T4 = ((M0_AA6060_T4 / kapa_AA6060_T4) * Xcoef * (Nc ** Ycoef
            ) * ( (c/h) ** Zcoef )) / 1000 # kN
334     Pm_AISI_316  = ((M0_AISI_316  / kapa_AISI_316 ) * Xcoef * (Nc ** Ycoef
            ) * ( (c/h) ** Zcoef )) / 1000 # kN
335     if mat==1:
336         return Pm_AA6060_T4
337     if mat==2:
338         return Pm_AISI_316
339
340 """
341 utils_design_vector.py
342 """
```

```
343 import numpy as np
344 import pandas as pd
345 import csv
346 import re
347 from scipy.integrate import simps
348 from sklearn.compose import TransformedTargetRegressor
349 from sklearn.cross_decomposition import PLSRegression
350 from sklearn.dummy import DummyRegressor
351 from sklearn.gaussian_process import GaussianProcessRegressor
352 from sklearn.isotonic import IsotonicRegression, isotonic_regression
353 from sklearn.linear_model import ARDRegression, ElasticNetCV,
        GammaRegressor, Huber, Lars, LarsCV, Lasso, LassoCV, LassoLars,
        LassoLarsCV, LassoLarsIC, LinearRegression, LogisticRegression,
        MultiTaskElasticNet, MultiTaskElasticNetCV, MultiTaskLasso,
        MultiTaskLassoCV, OrthogonalMatchingPursuit,
        OrthogonalMatchingPursuitCV, PassiveAggressiveRegressor,
        PoissonRegressor, QuantileRegressor, RANSACRegressor, Ridge, RidgeCV,
        TheilSenRegressor, TweedieRegressor, enet_path, lars_path_gram
354 import numpy as np
355 from sklearn.manifold import Isomap, LocallyLinearEmbedding
356 from sklearn.neural_network import MLPRegressor
357 from sklearn.gaussian_process.kernels import DotProduct, WhiteKernel
358 import utils_data as ud
359 import utils_design_vector as udv
360 import utils_plot as uplot
361
362 from catboost import CatBoostRegressor
363 from xgboost.sklearn import XGBRegressor
364 #
365 from scipy.optimize import OptimizeWarning
366 #
367 from sklearn.svm import SVR, LinearSVR, NuSVR
368 from sklearn.multioutput import RegressorChain, MultiOutputRegressor
369 from sklearn.linear_model import BayesianRidge, ElasticNet, HuberRegressor
370 from sklearn.model_selection import GridSearchCV
371 from sklearn.linear_model import LinearRegression
372 from sklearn.neighbors import KNeighborsRegressor,
        RadiusNeighborsRegressor
373 from sklearn.tree import DecisionTreeRegressor, ExtraTreeRegressor
374 from sklearn.ensemble import AdaBoostRegressor, BaggingRegressor,
        ExtraTreesRegressor, HistGradientBoostingRegressor,
        RandomForestRegressor, ExtraTreesClassifier, StackingRegressor,
        VotingRegressor
375 from sklearn.ensemble import GradientBoostingRegressor
376 from sklearn.kernel_ridge import KernelRidge
377 from sklearn.exceptions import ConvergenceWarning
378 #
379 import warnings
380 warnings.simplefilter(action='ignore', category=ConvergenceWarning)
381 warnings.simplefilter(action='ignore', category=FutureWarning)
382 warnings.simplefilter(action='ignore', category=OptimizeWarning)
383 #
384 from sklearn.linear_model import SGDRegressor
385 from sklearn.metrics import custom_score
386 from scipy.integrate import simps
387 import matplotlib.pyplot as plt
```

```
388 from sklearn.metrics import mean_absolute_percentage_error
389
390 def regression_model_advanced(method,X_test,X_train, y_train, predict):
391     test_sample = 0 # test sample is assumed to be only one so scalar 0
392     y_pred = None  # Initialize y_pred
393
394     if method == 'GradientBoost':
395         model = GradientBoostingRegressor()
396         if predict == 'multi-output':
397             wrapper = MultiOutputRegressor(model)
398             wrapper.fit(X_train, y_train)
399             geom_input = X_test[test_sample,:]
400             y_pred = wrapper.predict([geom_input])
401
402         if predict == 'single-output':
403             model.fit(X_train, y_train)
404             geom_input = X_test[test_sample,:]
405             y_pred = model.predict([geom_input])
406
407     if method == 'RF':
408         model = RandomForestRegressor()
409         model.fit(X_train, y_train)
410         geom_input = X_test[test_sample,:]
411         y_pred = model.predict([geom_input])
412
413     if method == 'GP':
414         model = GaussianProcessRegressor(alpha=10)
415         model.fit(X_train, y_train)
416         geom_input = X_test[test_sample,:]
417         y_pred = model.predict([geom_input])
418
419     if method == 'Linear':
420         model = LinearRegression()
421         model.fit(X_train, y_train)
422         geom_input = X_test[test_sample,:]
423         y_pred = model.predict([geom_input])
424
425     if method == 'Ridge':
426         model = KernelRidge()
427         model.fit(X_train, y_train)
428         geom_input = X_test[test_sample,:]
429         y_pred = model.predict([geom_input])
430
431     if method == 'TheilSenRegressor':
432         model = TheilSenRegressor()
433         if predict == 'multi-output':
434             wrapper = MultiOutputRegressor(model)
435             wrapper.fit(X_train, y_train)
436             geom_input = X_test[test_sample,:]
437             y_pred = wrapper.predict([geom_input])
438
439         if predict == 'single-output':
440             model.fit(X_train, y_train)
441             geom_input = X_test[test_sample,:]
442             y_pred = model.predict([geom_input])
443
```

```
444     if method == 'Transform␣Target␣Regressor':
445         model = TransformedTargetRegressor(regressor=LinearRegression(),
                func=np.log, inverse_func=np.exp)
446         model.fit(X_train, y_train)
447         geom_input = X_test[test_sample,:]
448         y_pred = model.predict([geom_input])
449 
450     if method == 'AdaBoost':
451         model = AdaBoostRegressor(n_estimators=100, random_state=0)
452 
453         if predict == 'multi-output':
454             wrapper = MultiOutputRegressor(model)
455             wrapper.fit(X_train, y_train)
456             geom_input = X_test[test_sample,:]
457             y_pred = wrapper.predict([geom_input])
458 
459         if predict == 'single-output':
460             model.fit(X_train, y_train)
461             geom_input = X_test[test_sample,:]
462             y_pred = model.predict([geom_input])
463 
464     if method == 'Poisson␣Regressor':
465         model = PoissonRegressor()
466         if predict == 'multi-output':
467             wrapper = MultiOutputRegressor(model)
468             wrapper.fit(X_train, y_train)
469             geom_input = X_test[test_sample,:]
470             y_pred = wrapper.predict([geom_input])
471 
472         if predict == 'single-output':
473             model.fit(X_train, y_train)
474             geom_input = X_test[test_sample,:]
475             y_pred = model.predict([geom_input])
476 
477     if method == 'Gamma':
478         model = GammaRegressor()
479         if predict == 'multi-output':
480             wrapper = MultiOutputRegressor(model)
481             wrapper.fit(X_train, y_train)
482             geom_input = X_test[test_sample,:]
483             y_pred = wrapper.predict([geom_input])
484 
485         if predict == 'single-output':
486             model.fit(X_train, y_train)
487             geom_input = X_test[test_sample,:]
488             y_pred = model.predict([geom_input])
489 
490     if method == 'RadNeig.':
491         model = RadiusNeighborsRegressor(radius=0.5)
492         model.fit(X_train, y_train)
493         geom_input = X_test[test_sample,:]
494         y_pred = model.predict([geom_input])
495 
496     if method == 'DecTree':
497         model = DecisionTreeRegressor()
498         model.fit(X_train, y_train)
```

```
499         geom_input = X_test[test_sample,:]
500         y_pred = model.predict([geom_input])
501 
502     if method == 'ElNet':
503         #model = Lasso(alpha=0.0000001, max_iter= 10000, selection='random
                ') # for the ones workes a=100 as the only attribute
504         model = ElasticNet()
505         model.fit(X_train, y_train)
506         geom_input = X_test[test_sample,:]
507         y_pred = model.predict([geom_input])
508     return y_pred
509 
510 
511 def regression_model(method,X_test,X_train, y_train, predict,
        cross_section):
512     test_sample = 0 # test sample is assumed to be only one so scalar 0
513     y_pred = None  # Initialize y_pred
514 
515     if method == 'Tran.␣Targ.':
516         model = TransformedTargetRegressor(regressor=LinearRegression(),
                func=np.log, inverse_func=np.exp)
517         model.fit(X_train, y_train)
518         geom_input = X_test[test_sample,:]
519         y_pred = model.predict([geom_input])
520 
521     if method == 'AdaBoost␣':
522         model = AdaBoostRegressor(n_estimators=100, random_state=0)
523 
524         if predict == 'multi-output':
525             wrapper = MultiOutputRegressor(model)
526             wrapper.fit(X_train, y_train)
527             geom_input = X_test[test_sample,:]
528             y_pred = wrapper.predict([geom_input])
529 
530         if predict == 'single-output':
531             model.fit(X_train, y_train)
532             geom_input = X_test[test_sample,:]
533             y_pred = model.predict([geom_input])
534 
535     if method == 'Poisson␣␣␣':
536         model = PoissonRegressor()
537         if predict == 'multi-output':
538             wrapper = MultiOutputRegressor(model)
539             wrapper.fit(X_train, y_train)
540             geom_input = X_test[test_sample,:]
541             y_pred = wrapper.predict([geom_input])
542 
543         if predict == 'single-output':
544             model.fit(X_train, y_train)
545             geom_input = X_test[test_sample,:]
546             y_pred = model.predict([geom_input])
547 
548     if method == 'Rad.␣Neig.':
549         model = RadiusNeighborsRegressor(radius=0.5)
550         model.fit(X_train, y_train)
551         geom_input = X_test[test_sample,:]
```

```
552         y_pred = model.predict([geom_input])
553 
554     if method == 'Dec. Tree':
555         model = DecisionTreeRegressor()
556         model.fit(X_train, y_train)
557         geom_input = X_test[test_sample,:]
558         y_pred = model.predict([geom_input])
559 
560     if method == 'Custom':
561         estimators = [
562              ('lr', RidgeCV()),
563              ('svr', LinearSVR(dual="auto", random_state=42))]
564         model = BaggingRegressor()
565 
566         decision =1
567 
568         if decision == 1:
569             model.fit(X_train, y_train)
570             geom_input = X_test[test_sample,:]
571             y_pred = model.predict([geom_input])
572         if decision == 2:
573             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                regression
574             wrapper.fit(X_train, y_train)
575             geom_input = X_test[test_sample,:]
576             y_pred = wrapper.predict([geom_input])
577         if decision == 3:
578             wrapper = RegressorChain(model)  # Direct multioutput
                regression
579             wrapper.fit(X_train, y_train)
580             geom_input = X_test[test_sample,:]
581             y_pred = wrapper.predict([geom_input])
582 
583 
584     if method == 'PLS':
585         if cross_section == 'Square cross section':
586             model = PLSRegression(n_components=1)
587         if cross_section == 'Circular cross section':
588             model = PLSRegression(n_components=4, scale=False, max_iter
                =8000, tol=1e-20, copy=True)
589         if cross_section == 'Hexagonal cross section':
590             model = PLSRegression(n_components=2, scale=False, max_iter
                =8000, tol=1e-20, copy=True)
591         if cross_section == 'Octagonal cross section':
592             model = PLSRegression(n_components=1)
593         decision = 1
594 
595         if decision == 1:
596             model.fit(X_train, y_train)
597             geom_input = X_test[test_sample,:]
598             y_pred = model.predict([geom_input])
599         if decision == 2:
600             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                regression
601             wrapper.fit(X_train, y_train)
602             geom_input = X_test[test_sample,:]
```

```
603             y_pred = wrapper.predict([geom_input])
604         if decision == 3:
605             wrapper = RegressorChain(model)  # Direct multioutput
                    regression
606             wrapper.fit(X_train, y_train)
607             geom_input = X_test[test_sample,:]
608             y_pred = wrapper.predict([geom_input])
609 
610     if method == 'GradBoost':
611 
612         model = GradientBoostingRegressor()
613 
614         if predict=='single-output':
615             decision = 1
616         else:
617             decision = 2
618 
619         if decision == 1:
620             model.fit(X_train, y_train)
621             geom_input = X_test[test_sample,:]
622             y_pred = model.predict([geom_input])
623         if decision == 2:
624             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
625             wrapper.fit(X_train, y_train)
626             geom_input = X_test[test_sample,:]
627             y_pred = wrapper.predict([geom_input])
628         if decision == 3:
629             wrapper = RegressorChain(model)  # Direct multioutput
                    regression
630             wrapper.fit(X_train, y_train)
631             geom_input = X_test[test_sample,:]
632             y_pred = wrapper.predict([geom_input])
633 
634     if method == 'ElNet':
635         #model = Lasso(alpha=0.0000001, max_iter= 10000, selection='random
                ') # for the ones workes a=100 as the only attribute
636         model = ElasticNet()
637         model.fit(X_train, y_train)
638         geom_input = X_test[test_sample,:]
639         y_pred = model.predict([geom_input])
640 
641     if method == 'GP␣Regr.':
642         model = GaussianProcessRegressor(alpha=10)
643         model.fit(X_train, y_train)
644         geom_input = X_test[test_sample,:]
645         y_pred = model.predict([geom_input])
646 
647     if method == 'Huber':
648         if cross_section == 'Hexagonal␣cross␣section':
649             model = HuberRegressor(
650             epsilon= 1.5,
651             max_iter= 2000,
652             alpha= 10000,
653             warm_start= True,
654             fit_intercept= True,
```

```
655             tol= 0.0000001)
656         else:
657             model = HuberRegressor()
658 
659         if predict=='single-output':
660             decision = 1
661         if predict=='multi-output':
662             decision = 2
663 
664         if decision == 1:
665             model.fit(X_train, y_train)
666             geom_input = X_test[test_sample,:]
667             y_pred = model.predict([geom_input])
668         if decision == 2:
669             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
670             wrapper.fit(X_train, y_train)
671             geom_input = X_test[test_sample,:]
672             y_pred = wrapper.predict([geom_input])
673         if decision == 3:
674             wrapper = RegressorChain(model)  # Direct multioutput
                    regression
675             wrapper.fit(X_train, y_train)
676             geom_input = X_test[test_sample,:]
677             y_pred = wrapper.predict([geom_input])
678 
679     if method == 'TheilSenRegressor':
680         model = TheilSenRegressor(n_subsamples=5)
681         decision = 2
682         if decision == 1:
683             model.fit(X_train, y_train)
684             geom_input = X_test[test_sample,:]
685             y_pred = model.predict([geom_input])
686         if decision == 2:
687             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
688             wrapper.fit(X_train, y_train)
689             geom_input = X_test[test_sample,:]
690             y_pred = wrapper.predict([geom_input])
691         if decision == 3:
692             wrapper = RegressorChain(model)  # Direct multioutput
                    regression
693             wrapper.fit(X_train, y_train)
694             geom_input = X_test[test_sample,:]
695             y_pred = wrapper.predict([geom_input])
696 
697     if method == 'LarsCV':
698         model = LarsCV(max_iter = 5000, max_n_alphas=2000)
699         decision = 2
700         if decision == 1:
701             model.fit(X_train, y_train)
702             geom_input = X_test[test_sample,:]
703             y_pred = model.predict([geom_input])
704         if decision == 2:
705             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
```

```
706             wrapper.fit(X_train, y_train)
707             geom_input = X_test[test_sample,:]
708             y_pred = wrapper.predict([geom_input])
709         if decision == 3:
710             wrapper = RegressorChain(model)  # Direct multioutput
                    regression
711             wrapper.fit(X_train, y_train)
712             geom_input = X_test[test_sample,:]
713             y_pred = wrapper.predict([geom_input])
714 
715 
716 
717     if method == 'Linear␣Regr.':
718         model = LinearRegression()
719         model.fit(X_train, y_train)
720         geom_input = X_test[test_sample,:]
721         y_pred = model.predict([geom_input])
722     if method == 'KNeighbors␣Regr.':
723         model = KNeighborsRegressor()
724         model.fit(X_train, y_train)
725         geom_input = X_test[test_sample,:]
726         y_pred = model.predict([geom_input])
727     if method == 'Kernel␣Ridge':
728         model = KernelRidge()
729         model.fit(X_train, y_train)
730         geom_input = X_test[test_sample,:]
731         y_pred = model.predict([geom_input])
732 
733     if method == 'Random␣Forest␣Regr.':
734         model = RandomForestRegressor()
735         model.fit(X_train, y_train)
736         geom_input = X_test[test_sample,:]
737         y_pred = model.predict([geom_input])
738 
739     if method == 'Huber␣Regr.':
740         model = HuberRegressor()
741         if predict == 'multi-output':
742             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
743             wrapper.fit(X_train, y_train)
744             geom_input = X_test[test_sample,:]
745             y_pred = wrapper.predict([geom_input])
746         if predict == 'single-output':
747             model.fit(X_train, y_train)
748             geom_input = X_test[test_sample,:]
749             y_pred = model.predict([geom_input])
750 
751     if method == 'XGB␣Regr.':
752         if predict == 'multi-output':
753             model = XGBRegressor()
754             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
755             wrapper.fit(X_train, y_train)
756             geom_input = X_test[test_sample,:]
757             y_pred = wrapper.predict([geom_input])
758         if predict == 'XGBRegressor':
```

```
759             model = XGBRegressor()
760             model.fit(X_train, y_train)
761             geom_input = X_test[test_sample,:]
762             y_pred = model.predict([geom_input])
763 
764     if method == 'Elastic␣Net':
765         if predict == 'multi-output':
766             model = ElasticNet()
767             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
768             wrapper.fit(X_train, y_train)
769             geom_input = X_test[test_sample,:]
770             y_pred = wrapper.predict([geom_input])
771         if predict == 'single-output':
772             model = ElasticNet()
773             model.fit(X_train, y_train)
774             geom_input = X_test[test_sample,:]
775             y_pred = model.predict([geom_input])
776 
777     if method == 'Bayesian␣Ridge':
778         if predict == 'multi-output':
779             model = BayesianRidge()
780             wrapper = MultiOutputRegressor(model)  # Direct multioutput
                    regression
781             wrapper.fit(X_train, y_train)
782             geom_input = X_test[test_sample,:]
783             y_pred = wrapper.predict([geom_input])
784         if predict == 'single-output':
785             model = BayesianRidge()
786             model.fit(X_train, y_train)
787             geom_input = X_test[test_sample,:]
788             y_pred = model.predict([geom_input])
789     return y_pred
790 
791 def assign_arrays(list_test,list_train,test_train_list,
    Include_integral_in_the_training, predict):
792     """
793     This part extracts the design vector and defines the x-axis of the
            Load-displacement function.
794     """
795     # Define a regular expression pattern that accommodates different
            codes
796     pattern = re.compile(r'Output_m(\d+)_(\w+)_(\d+)_(\dp\d+|\d+)_(\d{3})'
        )
797 
798     # Extracting relevant information and writing to CSV
799     with open('tube_features.csv', 'w', newline='') as csvfile:
800         csv_writer = csv.writer(csvfile)
801         for item in test_train_list:
802             match = pattern.match(item)
803 
804             if match:
805                 groups = match.groups()
806                 tube_features = (groups[0], groups[2], groups[3].replace('
                        p', '.'), groups[4])
807                 csv_writer.writerow(tube_features)
```

```
808
809     specs = 4 # geometrical parameters that vary accross the samples
810     column_names1 = ['feat' + str(i) for i in range(1, specs + 1)]
811     dfeatures = pd.read_csv("tube_features.csv", header=None, names=
            column_names1)
812
813     X=dfeatures.iloc[ : , :].values
814
815     test_dataset = [0]
816     # Seperate test and training features data
817     X_test = X[test_dataset]
818     X_train = np.delete(X, test_dataset, axis=0)
819
820     # dimentions of the load displacement arrays (train and test)
821     k_test = len(list_test)                    # Number of samples
822     n_test = len(list_test[0].iloc[1:, 1])     # Assuming all dataframes
            have the same shape
823     k_train = len(list_train)                  # Number of samples
824     n_train = len(list_train[0].iloc[1:, 1])   # Assuming all dataframes
            have the same shape
825
826     """
827     This part is when we predict the 'multi-output' and then we compute
            the integral
828     """
829     if predict == 'multi-output' and Include_integral_in_the_training ==
            False:
830         # Definition with zeros
831         y_test , y_test_without_Int , disp_test  = np.zeros((3, k_test,
                n_test))   # Initialize the arrays
832         y_train, y_train_without_Int, disp_train = np.zeros((3, k_train,
                n_train)) # Initialize the arrays
833
834         # Iterate over the test list and assign values to the load and
                displacement array
835         for i, df_test in enumerate(list_test):
836             disp_test[i, :]          = df_test.iloc[1:, 0]
837             y_test_without_Int[i, :] = df_test.iloc[1:, 1]
838             y_test[i, :]             = df_test.iloc[1:, 1]
839
840         tet_sample = 0
841         x_array = disp_test[tet_sample, :]
842
843
844         # Iterate over the train list and assign values to the load and
                displacement array
845         for i, df_train in enumerate(list_train):
846             disp_train[i, :]          = df_train.iloc[1:, 0]
847             y_train_without_Int[i, :] = df_train.iloc[1:, 1]
848             y_train[i, :]             = df_train.iloc[1:, 1]
849
850         #target_integrals = np.zeros((k_train, n_train-1)) # Initialize
                the arrays
851         #for i in range(k_train):       # rows
852         #    for j in range(n_train-1): # columns
853         #        target_integrals[i, j] = simps(y_train[i, 0:j+2] / 1000,
```

```
            x_array[ 0:j+2])
854
855         #orange = y_train
856         #y_train = []
857         #y_train = np.hstack((orange, target_integrals))
858
859     """
860     This part is when we predict directly the Integral
861     """
862     if predict == 'single-output':
863         # Definition with zeros
864         y_test  = np.zeros(k_test)
865         y_train = np.zeros(k_train)
866         #
867         y_test_without_Int , disp_test  = np.zeros((2, k_test, n_test))
                # Initialize the arrays
868         y_train_without_Int, disp_train = np.zeros((2, k_train, n_train))
                # Initialize the arrays
869
870         # Iterate over the test list and assign values to the load and
                displacement array
871         for i, df_test in enumerate(list_test):
872             disp_test[i, :]          = df_test.iloc[1:, 0]
873             y_test_without_Int[i, :] = df_test.iloc[1:, 1]
874
875         tet_sample = 0
876         x_array = disp_test[tet_sample, :]
877
878         # Iterate over the train list and assign values to the load and
                displacement array
879         for i, df_train in enumerate(list_train):
880             disp_train[i, :]          = df_train.iloc[1:, 0]
881             y_train_without_Int[i, :] = df_train.iloc[1:, 1]
882
883         # Calculate the new column using Simpson's rule integration
884         for i in range(k_test):
885             y_test[i] = simps(y_test_without_Int[i, :] / 1000, x_array)
886         # Calculate the new column using Simpson's rule integration
887         for i in range(k_train):
888             y_train[i] = simps(y_train_without_Int[i, :] / 1000, x_array)
889
890
891     return X_train, y_train, X_test, y_test, x_array#, target_integrals
892
893
894 def assign_Pmax_and_x_at_Pmax(list_test,list_train,test_train_list):
895     """
896     This part extracts the design vector and defines the x-axis of the
            Load-displacement function.
897     """
898     # Define a regular expression pattern that accommodates different
            codes
899     pattern = re.compile(r'Output_m(\d+)_(\w+)_(\d+)_(\dp\d+|\d+)_(\d{3})'
            )
900
901     # Extracting relevant information and writing to CSV
```

```
902     with open('tube_features.csv', 'w', newline='') as csvfile:
903         csv_writer = csv.writer(csvfile)
904         for item in test_train_list:
905             match = pattern.match(item)
906 
907             if match:
908                 groups = match.groups()
909                 tube_features = (groups[0], groups[2], groups[3].replace('
                        p', '.'), groups[4])
910                 csv_writer.writerow(tube_features)
911 
912     specs = 4 # geometrical parameters that vary accross the samples
913     column_names1 = ['feat' + str(i) for i in range(1, specs + 1)]
914     dfeatures = pd.read_csv("tube_features.csv", header=None, names=
            column_names1)
915 
916     X=dfeatures.iloc[ : , :].values
917 
918     test_dataset = [0]
919 
920     # Seperate test and training features data
921     X_test = X[test_dataset]
922     X_train = np.delete(X, test_dataset, axis=0)
923 
924     # dimentions of the load displacement arrays (train and test)
925     k_test = len(list_test)                    # Number of samples
926     n_test = len(list_test[0].iloc[1:, 1])     # Assuming all dataframes
            have the same shape
927     k_train = len(list_train)                  # Number of samples
928     n_train = len(list_train[0].iloc[1:, 1])   # Assuming all dataframes
            have the same shape
929 
930 
931     # Definition with zeros
932     values_to_predict = 4
933     y_test  = np.zeros((k_test,values_to_predict ))
934     y_train = np.zeros((k_train,values_to_predict))
935     #
936     curves_test , disp_test  = np.zeros((2, k_test, n_test))   #
            Initialize the arrays
937     curves_train, disp_train = np.zeros((2, k_train, n_train)) #
            Initialize the arrays
938 
939     # Iterate over the test list and assign values to the load and
            displacement array
940     for i, df_test in enumerate(list_test):
941         disp_test[i, :]          = df_test.iloc[1:, 0]
942         curves_test[i, :] = df_test.iloc[1:, 1] / 1000 # in kN
943     tet_sample = 0
944     x_array = disp_test[tet_sample, :]
945 
946     # Iterate over the train list and assign values to the load and
            displacement array
947     for i, df_train in enumerate(list_train):
948         disp_train[i, :]          = df_train.iloc[1:, 0]
949         curves_train[i, :] = df_train.iloc[1:, 1] / 1000 # in kN
```

```
950
951     for i in range(k_test):
952         y_test[i,0] = np.max(curves_test[i, :500])  # Pmax in kN
953         index = []
954         index = np.argmax(curves_test[i, :500])
955         y_test[i,1] = index
956 
957         y_test[i,2] = np.max(curves_test[i, 500:])  # Pmax in kN
958         index = []
959         index = np.argmax(curves_test[i, 500:])
960         y_test[i,3] = index
961 
962 
963     for i in range(k_train):
964         y_train[i,0] = np.max(curves_train[i, :500])  # Pmax in kN
965         index = []
966         index = np.argmax(curves_train[i, :500])
967         y_train[i,1] = index
968 
969         y_train[i,2] = np.max(curves_train[i, 500:])  # Pmax in kN
970         index = []
971         index = np.argmax(curves_train[i, 500:])
972         y_train[i,3] = index
973 
974     return X_train, y_train,    X_test,    y_test, x_array, curves_test,
        curves_train
975 
976 """
977 utils_plot.py
978 """
979 import numpy as np
980 import pandas as pd
981 from scipy.integrate import simps
982 import matplotlib.pyplot as plt
983 from sklearn.metrics import mean_absolute_percentage_error
984 
985 def smooth(y, box_pts):
986     box = np.ones(box_pts)/box_pts
987     y_smooth = np.convolve(y, box, mode='same')
988     return y_smooth
989 
990 def moving_average(x, w):
991     return np.convolve(x, np.ones(w), 'valid') / w
992 
993 # Apply the moving average filter to smooth the data
994 window_size = 20
995 #smoothed_mean = moving_average(mean[0]/1000, window_size)
996 
997 def plot_graph(y_pred, y_test, x_array, list_train,
    Include_integral_in_the_training, cross_section, material, method,
    predict,method_list):
998     if predict == 'multi-output':
999         test_sample = 0
1000         f = 1000
1001 
1002         # calculate the areas below the curves with sims rule
```

```
        area_test = simps(y_test[0, :f] / 1000, x_array)
        area_pred = simps(smooth(y_pred[0, :f] / 1000,19), x_array)

        # evaluate the curve wrt the target
        mspe1 = mean_absolute_percentage_error(y_test[test_sample, :f
            ]/1000,  y_pred[0, :f]/1000)
        # evaluate the area wrt the target area
        mspe2 = mean_absolute_percentage_error(np.array([area_test]), np.
            array([area_pred]))
        mspe3 = mean_absolute_percentage_error(np.array([area_test]), np.
            array([y_pred[0, -1]]))
        if Include_integral_in_the_training == True:
            print(f"Method: {method:20}, target: {area_test:.2f}, 
                area_pred: {area_pred:.2f} & {mspe2*100:5.2f} & {y_pred[0, 
                -1]:.2f} & {mspe3*100:5.2f} ")
        if Include_integral_in_the_training == False:
            print(f"Method: {method:20}, target: {area_test:.2f} & {
                area_pred:.2f} & {mspe2*100:5.2f} & ")

        plt.plot(x_array, smooth(y_pred[0, :f]/1000,19), 's-', markersize
            =1, linewidth=1, label='{} , MSPE: {:.2f}%'.format(method,
            mspe1*100))
        plt.xlabel('Displacement (mm)', fontsize=14)
        plt.ylabel('Load (kN)' , fontsize=14)
        plt.title('{} - #Samples: {} - {}'.format(cross_section, len(
            list_train), material), fontsize=14)
        plt.xlim(0, max(x_array))

        if method == method_list[-1]:
            ###
            if cross_section == 'Square cross section':
                data = np.genfromtxt("gp-sq-matern(r3).csv", delimiter=","
                    )
            if cross_section == 'Circular cross section':
                data = np.genfromtxt("gp-circ.csv", delimiter=",")
            if cross_section == 'Octagonal cross section':
                data = np.genfromtxt("gp-oct.csv", delimiter=",")
            if cross_section == 'Hexagonal cross section':
                data = np.genfromtxt("gp-hex.csv", delimiter=",")
            y_pred_gp = data
            method = 'GP'
            area_pred = simps(y_pred_gp / 1000, x_array)
            mspe2 = mean_absolute_percentage_error(np.array([area_test]),
                np.array([area_pred]))
            print(f"Method: {method:20}, target: {area_test:.2f} & {
                area_pred:.2f} & {mspe2*100:5.2f} & ")

            mspe1 = mean_absolute_percentage_error(y_test[test_sample, :f
                ]/1000,  y_pred_gp/1000)
            plt.plot(x_array, smooth(y_pred_gp/1000,19),'m^-', markersize
                =1, label='GP , MSPE: {:.2f}%'.format(mspe1*100))
            ###
            plt.plot(x_array, y_test[0, :f]/1000,'k^-', markersize=1,
                label='Target')
            plt.tick_params(axis='both', which='major', labelsize=14)
            #plt.legend(loc='upper center', fontsize=14)
```

```
            plt.legend(loc='upper center', bbox_to_anchor=(0.45, -0.15),
                ncol=3, columnspacing=0.12, fontsize=12)
            plt.show()
            ###

    if predict == 'single-output':
        mspe2 = mean_absolute_percentage_error(np.array([y_test[0]]), np.
            array([y_pred[0]]))
        print(f"Method: {method:30}, y_test:{y_test[0]:.2f}, y_pred: & {
            y_pred[0]:.2f} & {mspe2*100:5.2f} ")
    return


def plot_graph_advanced(y_pred, X_test_fea, y_test_fea, x_array, predict):
    if predict == 'multi-output':
        # calculate the areas below the curves with sims rule
        area_test = simps(y_test_fea[0, :], x_array)
        area_pred = simps(y_pred, x_array)
        mspe = mean_absolute_percentage_error(np.array([area_test]), np.
            array([area_pred])) * 100
```

B Datasets from the numerical model

Table B.1: Datasets used for the square cross-section (in-extensional crushing mode)

material	side length (mm)	thickness (mm)	tube length (mm)
AA6060-T4	30	1.0	200
AA6060-T4	30	1.5	200
AA6060-T4	30	1.25	200
AA6060-T4	30	1.75	200
AA6060-T4	30	2.0	200
AA6060-T4	30	2.5	200
AA6060-T4	30	2.25	200
AA6060-T4	40	1.0	200
AA6060-T4	40	1.5	200
AA6060-T4	40	1.25	200
AA6060-T4	40	1.75	200
AA6060-T4	40	2.0	200
AA6060-T4	40	2.5	200
AA6060-T4	40	2.25	200
AA6060-T4	50	1.0	200
AA6060-T4	50	1.5	200
AA6060-T4	50	1.25	200
AA6060-T4	50	1.75	200
AA6060-T4	50	2.0	200
AA6060-T4	50	2.5	200
AA6060-T4	50	2.25	200
AA6060-T4	60	1.0	200
AA6060-T4	60	1.5	200
AA6060-T4	60	1.25	200
AA6060-T4	60	1.75	200
AA6060-T4	60	2.0	200
AA6060-T4	60	2.5	200
AA6060-T4	60	2.25	200
AISI-316	30	1.0	200
AISI-316	30	1.5	200
AISI-316	30	1.25	200
AISI-316	40	1.0	200
AISI-316	40	1.5	200
AISI-316	40	1.25	200
AISI-316	50	1.0	200
AISI-316	50	1.5	200
AISI-316	50	1.25	200
AISI-316	60	1.0	200
AISI-316	60	1.5	200
AISI-316	60	1.25	200

Table B.2: Datasets used for the circular cross-section (in-extensional crushing mode)

material	side length (mm)	thickness (mm)	tube length (mm)
AA6060-T4	20	1.75	200
AA6060-T4	15	1	100
AA6060-T4	15	1.5	100
AA6060-T4	15	1.25	100
AA6060-T4	15	1.75	100
AA6060-T4	15	2	100
AA6060-T4	20	1	100
AA6060-T4	20	1.5	200
AA6060-T4	20	1.25	200
AA6060-T4	20	2	200
AA6060-T4	25	1	100
AA6060-T4	25	1.5	200
AA6060-T4	25	1.25	100
AA6060-T4	25	1.75	200
AA6060-T4	25	2	200
AA6060-T4	30	1	100
AA6060-T4	30	1.5	200
AA6060-T4	30	1.25	200
AA6060-T4	30	1.75	200
AA6060-T4	30	2	200
AISI-316	15	1	100
AISI-316	15	1.5	50
AISI-316	15	1.25	50
AISI-316	20	1	100
AISI-316	20	1.5	100
AISI-316	20	1.25	100
AISI-316	25	1	100
AISI-316	25	1.5	100
AISI-316	25	1.25	200
AISI-316	30	1	200
AISI-316	30	1.5	100
AISI-316	30	1.25	100

Table B.3: Datasets used for the hexagonal cross-section (in-extensional crushing mode)

material	side length (mm)	thickness (mm)	tube length (mm)
AA6060-T4	50	1.75	200
AA6060-T4	30	1	200
AA6060-T4	30	1.5	200
AA6060-T4	30	1.25	200
AA6060-T4	30	1.75	200
AA6060-T4	30	2	200
AA6060-T4	40	1	200
AA6060-T4	40	1.5	200
AA6060-T4	40	1.25	200
AA6060-T4	40	1.75	200
AA6060-T4	40	2	200
AA6060-T4	50	1	200
AA6060-T4	50	1.5	200
AA6060-T4	50	1.25	200
AA6060-T4	50	2	200
AA6060-T4	60	1	200
AA6060-T4	60	1.5	200
AA6060-T4	60	1.25	200
AA6060-T4	60	1.75	200
AA6060-T4	60	2	200
AISI-316	30	1	200
AISI-316	30	1.5	200
AISI-316	30	1.12	200
AISI-316	30	1.25	200
AISI-316	30	1.37	200
AISI-316	40	1	200
AISI-316	40	1.5	200
AISI-316	40	1.12	200
AISI-316	40	1.25	200
AISI-316	40	1.37	200
AISI-316	50	1	200
AISI-316	50	1.5	200
AISI-316	50	1.12	200
AISI-316	50	1.25	200
AISI-316	50	1.37	200
AISI-316	60	1	200
AISI-316	60	1.5	200
AISI-316	60	1.12	200
AISI-316	60	1.25	200
AISI-316	60	1.37	200

Table B.4: Datasets used for the octagonal cross-section (in-extensional crushing mode)

material	side length (mm)	thickness (mm)	tube length (mm)
AA6060-T4	30	1	200
AA6060-T4	40	1	200
AA6060-T4	40	1.25	200
AA6060-T4	50	1	200
AA6060-T4	50	1.5	200
AA6060-T4	50	1.25	200
AA6060-T4	60	1	200
AA6060-T4	60	1.25	200
AA6060-T4	60	1.5	200
AA6060-T4	60	1.75	200
AISI-316	30	1	200
AISI-316	30	1.12	200
AISI-316	40	1	200
AISI-316	40	1.12	200
AISI-316	40	1.25	200
AISI-316	40	1.37	200
AISI-316	50	1	200
AISI-316	50	1.5	200
AISI-316	50	1.12	200
AISI-316	50	1.25	200
AISI-316	50	1.37	200
AISI-316	60	1	200
AISI-316	60	1.5	200
AISI-316	60	1.12	200
AISI-316	60	1.25	200
AISI-316	60	1.37	200

C HF vs. HFLF with the Decision Tree Regressor

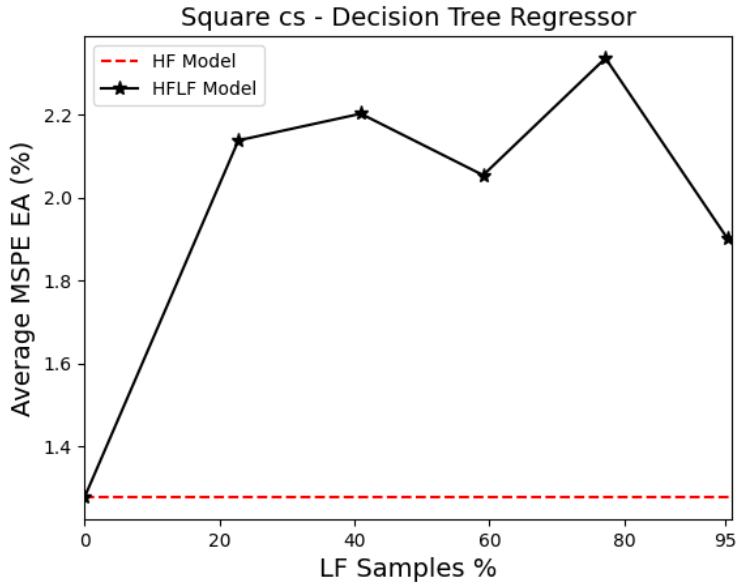


Figure C.1: Regression Solely Based on HF samples vs. Regression Based on HF and LF samples with the Decision Tree MO Regr. - Square cs - HF Samples = 40