Universidade de Aveiro Departamento de Engenharia Mecânica
2021

Pedro Miguel Monteiro da Rocha

Optimisation of Light-Weight Armour Plates for Energy Absorption

Otimização de Proteções Balísticas de Baixo Peso para Absorção de Energia

Universidade de Aveiro Departamento de Engenharia Mecânica
2021

Pedro Miguel Monteiro da Rocha

Optimisation of Light-Weight Armour Plates for Energy Absorption

Otimização de Proteções Balísticas de Baixo Peso para Absorção de Energia

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de João Alexandre Dias de Oliveira, Professor Auxiliar da Universidade de Aveiro, e de Filipe Teixeira-Dias, Reader da School of Engineering da The University of Edinburgh (United Kingdom).

o júri / the jury

presidente / president

Prof. Doutor António Gil D'Orey de Andrade Campos
Professor Auxiliar c/ Agregação da Universidade de Aveiro

Doutora Ana Virgínia Ferreira Azevedo
Researcher da *Royal Military Academy*

Prof. Doutor João Alexandre Dias de Oliveira
Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Ao Professor Doutor João Alexandre Dias de Oliveira pelo apoio e disponibilidade contínuos, a sua mentoria neste projeto foi de grande importância e completamente indispensável. Também gostaria de agradecer toda a motivação e conhecimento transmitido.

Ao Professor Doutor Filipe Teixeira-Dias que sempre esteve disponível para apoiar e orientar neste trabalho.

À minha família, que foi muito importante no meu percurso académico e que me forneceram sempre apoio incondicional. Sem ela, este meu percurso não seria possível.

Aos meus amigos, que constituem muitas boas memórias desta etapa da minha vida e que me ajudaram bastante no meu desenvolvimento pessoal e profissional.

Finalmente, queria agradecer a todas as pessoas que influenciaram o meu percurso académico e com quem aprendi bastante.

keywords Ballistic impact, lightweight armour, layer optimisation, finite element analysis, wave propagation, non-linear material behaviour, Abaqus Python API, particle swarm optimisation

abstract

Technology advances continue to revolutionise military equipment. The development of new firepower induces an interest in the enhancement of protection gear, both for transportation vehicles and personnel. There has been a significant amount of research of methods to increase protection capabilities without increases in the weight of a given defence system. This dissertation seeks to develop an optimisation tool that results in light-weight armour plates without compromising protection capabilities. A thorough study on the propagation of elastic and plastic stress waves aims for a better understanding of how an armour system behaves upon ballistic impact.

The first part of this dissertation focuses on the development of a Python script that provides an efficient approach to model generation in Abaqus. It enables the user to avoid time consuming actions when designing ballistic test models to later simulate through the software. This script is also used to validate the theory behind elastic and plastic stress wave propagation while also being able to access output databases and interpret obtained results. The importance of the script is relevant for the second part of the dissertation, which takes advantage of the Abaqus Python Application Programming interface (API) to perform optimisation procedures automatically. Focusing particularly on the application of the particle swarm optimisation algorithm, this work continuously improves the efficiency and accuracy of the mentioned algorithm by dividing three different optimisation problems into several experiments. Each one of the experiments is carefully defined to highlight the impact of a specific operating parameter of the algorithm.

A validation of the stress wave propagation and how it is affected upon contact with layered media is carefully conducted through a series of different analysis approaches. It is shown that the plastic stress wave propagates slower than the elastic one and that plastic deformation affects the properties of the generated stress wave, such as wavelength. The implemented particle swarm optimisation algorithm proved to be an effective approach to problem solving, however, for complex problems the operational parameters must be carefully chosen.

palavras-chave

Impacto balístico; proteção de baixo peso; otimização de camadas, análise de elementos finitos, propagação de onda, comportamento não-linear de material, Abaqus Python API, otimização por enxame de partículas (PSO)

resumo

Os avanços na tecnologia continuam a revolucionar equipamentos militares. O desenvolvimento de novas armas de fogo induz interesse no aprimoramento de equipamento de proteção, para veículos de transporte e pessoal. Tem havido uma quantidade significativa de investigação de métodos para aumentar as capacidades de proteção sem aumento de peso de um dado sistema de proteção. Esta dissertação tem como objetivo o desenvolvimento de uma ferramenta de otimização que resulta em placas de armadura de baixo peso sem comprometer capacidades de proteção. Um estudo cuidadoso acerca da propagação de ondas de tensão elásticas e plásticas procura compreender a forma como um sistema de armadura reage após um impacto balístico.

A primeira parte desta dissertação foca-se no desenvolvimento de um código em Python que fornece uma abordagem eficiente à geração de modelos no Abaqus. Isto permite que o utilizador evite ações que consumam tempo ao criar modelos de teste balístico para simular mais tarde através do software. Este código é também usado para validar a teoria por detrás da propagação de ondas de tensão elásticas e plásticas e ao mesmo tempo habilitar o acesso a dados de saída do software e interpretar resultados obtidos. A importância do código é relevante para a segunda parte da dissertação, que tira vantagem da interface de aplicação e programação do Abaqus Python (API) para executar procedimentos de otimização de forma automática. Com foco em particular na aplicação do algoritmo de otimização por enxame de partículas, este trabalho melhora continuamente a eficácia e precisão do algoritmo mencionado através da divisão de três diferentes problemas de otimização em várias experiências. Cada uma das experiências é cuidadosamente definida para destacar o impacto de um parâmetro operacional específico do algoritmo.

A validação da propagação da onda de tensão e como é afetada após contacto com um meio material de múltiplas camadas é cuidadosamente estudada através de séries de diferentes análises. É mostrado que a onda de tensão plástica se propaga mais lentamente que a elástica e que deformação plástica afeta as propriedades da onda de tensão gerada, tal como o comprimento de onda. O algoritmo de otimização por enxame de partículas implementado prova ser uma abordagem eficaz para a resolução de problemas, no entanto, para problemas complexos os parâmetros operacionais devem ser escolhidos com cuidado.

Contents

List of Tables

Intentionally blank page.

List of Figures

Chapter 1

Introduction

Protection systems have been very important throughout the years. Warfare and criminal activity demand development of new protection resources. Prevention is essential and therefore the need for defensive gear is unavoidable. There are many high-risk professions that, due to the natural risks involved, require some type of safety ensurance. Nowadays, the bullet-proof vest is an example of protection gear that greatly increases the rate of survivability in violent scenarios.

Since thousands of years ago, the efficiency of a soldier has both been proved by agility proficiency or defensive endurance [Yadav *et al.* 2016]. This means that having soldiers who are faster than their enemies results in tactical advantage. On the other hand, having soldiers with better defensive gear helps withstanding more blows and therefore increasing their resilience. The focus on developing improved defensive gear has clear purposes. Increasing the endurance capabilities of soldiers without sacrificing their speed and comfort is the key to maximise the ratio of survivability. Engineers have been studying new materials and physical phenomena in order to improve existing products or develop new ones. New materials have been replacing metals for defence purposes and the introduction of fibres has revolutionised this industry. For example, the development of Kevlar® fabric armour meant a great reduction in weight while maintaining impact absorption capabilities. Modular protective gear has been improving as well, depending on the type of threat.

A documentary produced by [DeHart *et al.* 2012] depicts the constant need for improvement in relation to armoured vehicles. They were highlighted in the two world wars and were a force to be reckoned with. Although with each improvement made to armoured vehicles, enhancements on firepower were made as well, such as armour piercing projectiles. To be able to withstand such projectiles, new ideas had to be tested. Increasing thickness was not a viable option since it would make the vehicle too heavy and hard to manoeuvre. Changes in geometry were proven to be helpful at absorbing stronger impacts. Moreover, combinations between metal plates and ceramic plates would help absorbing heat rounds which otherwise penetrated thick armour plates. Explosive devices such as HESH (High Explosive Squash Head), IED (Improvised Explosive Device) and EFP (Explosively Formed Penetrator) are strong forces to be reckoned with and purposely built protection systems would be developed to deal with such threats. Bulky and heavy vehicles have been replaced by lighter and faster vehicles for evasion purposes. The combination of geometrical, material and structural factors would result in strong and light-weight armoured vehicles.

Throughout time, the development of armour systems has been in the context of action-reaction events. A new armour concept is developed and, in order to overcome its improved defensive capabilities, new firepower is developed, which results in the need for new armour enhancements for it to withstand that type of weaponry. Researchers study new materials and designs every day to come up with new defence prototypes.

1.1 Design and Material Choices for Armour Systems

Advances in science and technology are constantly enabling the production of new materials and enhancing them. Biomimetics, hybrid and composite materials introduce a new level of complexity when designing security devices for armour applications. [Martini and Barthelat 2016] developed a flexible bio-inspired armour based on overlapping ceramic scales. This armour proved to be more flexible, resistant and damage tolerant compared to a continuous layer of uniform ceramic. [Grujicic *et al.* 2017] research a new class of biomimetic ceramic-tablet-reinforced polymer-matrix composite materials, modelled after natural nacre, the inner layer of mollusc shells that is characterised by a superb combination of mechanical properties. This new class of composite materials would reduce the weight of military vehicles, that due to heavy metallic armour, have reduced fuel efficiency and mobility.

Hybrid materials are combinations of two or more materials assembled in such a way to have attributes not offered by either one alone [Ashby 2005]. The sandwich structure is a type of hybrid that is used in many armour systems. [Azevedo 2012] tested the influence of cores of micro-agglomerated cork in sandwich armour plates and demonstrated how this implementation can raise impact energy absorption capabilities. The use of composite materials and hybrid structures opened the door to many light-weight armour solutions. In 2006, an article covering an innovative composite armour system demonstrated that improved ballistic performance can be achieved through the design process [Zheng-Dong *et al.* 2006]. The authors also declared that the hybrid materials have the advantage of providing additional important design variables besides thickness and size. [Medvedovski 2006] proved that combining a light-weight ceramic-polymer layer between a backing layer of aramid and a ceramic plate would result in high level ballistic performance. With interest in light-weight armour solutions, [Sangamesh *et al.* 2018] conducted a research on jute-epoxy and natural rubber sandwich composites. The authors proved the enhanced ballistic protection provided by combination of the two materials, due to higher energy absorption from the rubber and superior structural stability from the presence of jute-epoxy. An article written by [Crouch 2019] mentions light-weight material solutions such as Ultra High Molecular Weight Polyethylene (UHMWPE) that are being used as a backing material in layered armour systems as a revolutionising new material solution.

[Tasdemirci and Hall 2007] mention the importance of wave reflection and transmission phenomena at the interface of two dissimilar elastic solids. In their paper, the propagation characteristics of impact waves across a planar interface between ceramic/metal and metal/metal layers are reported and plasticity effects are shown for the first time to be particularly important in these multi-layer materials comprising metal layers. These authors also sought to demonstrate and understand the effect of plastic deformation on stress wave propagation. [Saleh *et al.* 2020] stated that ballistic performance can be evaluated in the light of complex interaction of the impact stress waves and material strength and its fracture/failure properties. Testing how pressure waves propagate throughout armour and possibly causing blunt trauma due to large displacements, [Gilson *et al.* 2020] performed tests using ballistic gelatin which, due to similar behaviour to soft tissues, is considered a relevant witness material. The tests and models showed several pressure waves propagating into the gelatin over a short delay.

1.2 Optimisation Studies

Optimisation studies have been focusing on finding the best combination of given variables to achieve the best overall performance for developed armour systems. Optimisation methods allow the researcher to minimise the weight of defence systems without sacrificing defensive capabilities. [Liu *et al.* 2003] researched the design of functionally graded ceramic/metal armour and, using the conjugate gradient method, the authors were able to increase the armour's protection capabilities. [Rahul *et al.* 2005] mentioned the importance of heuristics approaches towards composite material selection and design. The authors also stated that optimisation methods based on genetic algorithms have demonstrated the potential to overcome many of the problems associated with gradient-based methods. These algorithms are most effective when the design space is large. [Chen 2001] introduces a practical approach for impact structure optimisation taking advantage of the global-searching ability of the genetic algorithm while also considering the instability of explicit finite element analysis. This metaheuristic was adopted by [Yong *et al.* 2008] to optimise the response of a composite laminate subjected to impact, coupling it to a commercial finite element package LS Dyna to perform the impact analysis. After a comparison between the commercial optimisation package, LS OPT, and the genetic algorithm, results showed that the adopted algorithm was a robust, capable optimisation tool that produced near optimal designs while performing well with respect to LS OPT. [Wang and Zheng 2012] stated that the particle swarm optimisation algorithm is conceptually simple and easy to implement and that it has demonstrated is efficiency in a wide range of continuous and combinatorial optimisation problems. The authors presented a new particle swarm optimisation algorithm and experimental results showed that it was able to achieve good solution quality with low computational costs.

Having many optimisation methods at disposal, researchers have proposed the application of these tools to increase the performance of multi-layered armour plates. [Park *et al.* 2005] conducted a study for an optimal design of a multi-layered plate to endure ballistic impact using NET2D, a Lagrangian explicit time-integration finite element code for impact analysis. For the simulation, the Johnson-Cook model is used as the constitutive models, to account the effects of strain rate hardening, strain hardening and thermal softening. To solve this problem, the response surface method based on the design of experiments is used to obtain the optimal design. For the objective function, the average temperature and average EQPS (Equivalent Plastic Strain) are chosen. As an effort to minimise the areal density of a two-component armour system, [Kędzierski *et al.* 2015] present an approach that approximates the objective function with a neural network and then searches for its optimum by applying a hybrid adaptive simulated annealing algorithm (ASA). Carrying out a study to find the optimal composite configuration in a sandwich structure, [Zahir *et al.* 2019] selected the Analytic Hierarchy Process (AHP) to be used as the multi-criteria decision making method. In order to find the best combination of strength, weight and cost, ten design configurations of composites laminates were evaluated. An optimisation framework for the design of light-weight multi-layer plate configurations with high levels of blast and impact protection is carried by [Jiang *et al.* 2020]. The efficient optimisation process used is based on the reduced-order multi-layer plate model and provides a fast and reliable routing to identify the optimal multi-layer plate among a large number of alternative configurations. It helps designers to select proper materials and determine the thickness of each sub-layer in an early design stage of a multi-layer plate armour design, which has a capacity to reduce the occupant injury probability and structural weight simultaneously. [Reis 2019] wrote a dissertation with aims to develop and understanding of non-linear optimisation algorithms applied to a complex engineering design problem: a multi-layer plate under a ballistic impact. Due to the complexity of the development of a model that simulates a ballistic impact, there was a need to understand the physics and mechanics behind the propagation of stress waves. Such insights were used in Abaqus Python scripting to construct and validate a proper simulation model of the transient event that constitutes a low speed ballistic impact, analysing the propagation of elastic stress waves and ultimately prepare a script to be fully integrated in three optimisation algorithms: Particle Swarm Optimisation, Genetic Algorithm and Simulated Annealing.

1.3 Objectives

This dissertation is focused on developing methodologies towards the attaining of a multi-layered armour plate that is subjected to a non-linear optimisation algorithm in search of suitable engineering solutions for energy absorption of a ballistic impact. This work can be divided into four different objectives:

- Study the propagation of stress waves and how it is affected by the mechanical properties of different materials as well as how it interacts when reaching an interface between two materials. Using a simple designed ballistic test model in Abaqus, the generated stress wave from the contact between a moving projectile and a stationary rectangular plate is analysed in several different tests conditions involving different projectile velocities and geometrical configurations.

- Create new strategies for material modelling and optimisation using the Python-Abaqus scripting interface. The scripts are used to develop numerical models in Abaqus Explicit FEM package, with different materials and layer configurations, as well as additional useful parameters for the Abaqus model generation;

- Understand how interlayer interfaces and material properties influence the total energy absorption and impact mitigation in layered armour configurations. Expanding on a previous work, this dissertation investigates the repercussions of different configurations, focusing on the effects of plastic deformation in the propagation of stress waves;

- Apply optimisation methodologies using the Python-Abaqus Scripting Interface, to solve problems with adequate variables and parameterisation to obtain optimal layered configurations for objective functions equating such as maximising protection to weight ratio. For this purpose, an application of the particle swarm optimisation algorithm is thoroughly analysed in terms of performance and accuracy in a set of prepared optimisation problems. The obtained results are analysed in terms of stress wave propagation and energy absorption.

1.4 Dissertation Structure

This document is divided and organised in six chapters, as follows:

- **Chapter 1**: This chapter contextualises the purpose of the dissertation, serving as an introduction and framework of the field of design optimisation and regarding relevant protection solutions. The objectives for this work are also presented in this chapter;
- **Chapter 2**: Focuses on the theoretical understanding of the propagation of stress waves, more specifically longitudinal elastic and plastic stress waves;
- **Chapter 3**: A development of the model generation script to implement on the various studies throughout this work is conducted in this chapter. Furthermore, the validation of the studied concepts on the previous chapter is carried out and analysed;
- **Chapter 4**: Introduction of optimisation fundamentals and methodologies. Sets up a preliminary structure for the optimisation problems analysed in the following chapter;
- **Chapter 5**: In this chapter, two optimisation problems are defined. The purpose of the two problems is to evaluate possible light-weight armour solutions while enhancing the overall performance of the implemented particle swarm optimisation algorithm. Each problem is divided in a series of experiments which, testing the influence of the algorithm's operational parameters, evaluates the accuracy of solutions as well as the performance in terms of computational cost. The solutions returned by the algorithm are studied and compared in terms of stress wave propagation and the energy transfers between the elements of the geometrical model: projectile and armour plates;
- **Chapter 6**: The final remarks of this work, outlining major conclusions of this dissertation and discussing recommendations for future research on this field.

Intentionally blank page.

Chapter 2

Stress Wave Dynamics

This dissertation focus on the dynamics of impact collisions. When two objects with different velocities collide, it is possible to determine the latter velocities of those objects depending on the type of impact: elastic or plastic. Classic rigid body dynamics describes the impact between bodies, however, if a stress wave is created, instead of analysing the problem in terms of kinetic energy, it is possible to analyse it in terms of wave propagation [Svensson and Tell 2015].

2.1 Types of Stress Waves

In a system composed by at least one deformable object, a collision originates stress waves in the material. Post impact, the rate of deformation along the material is a direct consequence of the propagation of stress waves originated from the impact region. To better understand the armour plate's reaction to the ballistic impact, it is important to consider the different types of stress waves originated by the collision. These stress waves traverse the material and the most common can be presented in three distinctive kinds: longitudinal, distortional and Rayleigh [Meyers 1994].

Longitudinal (or irrotational) waves correspond to the motion of the particles back and forth along the direction of wave propagation such that the particle velocity (U_p) is parallel to the wave velocity (U) [Meyers 1994]. In infinite and semi-infinite media, these waves are established as "dilatational" waves. An approach to how the particles move under such wave can be seen in Figure 2.1.

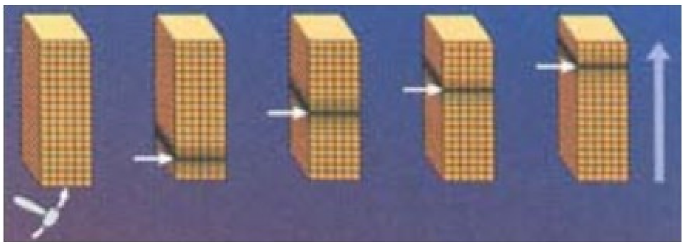


Figure 2.1: Motion characteristics of particles under longitudinal stress waves [Survey 2004].

The propagation of a distortional (or shear, or transverse, or equivolumal) wave causes the particles to move in a perpendicular direction to the wave. There is no change in density and all longitudinal strains are null [Meyers 1994]. Figure 2.2 represent the motion of the particles when submitted to a shear stress wave.

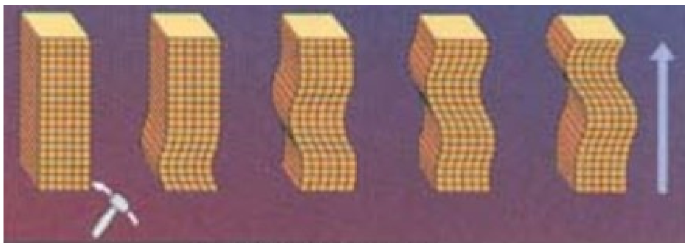


Figure 2.2: Motion characteristics of particles under distortional stress waves [Survey 2004].

Rayleigh waves only occur in regions adjacent to surfaces, thus they are also called surface waves. The particle's velocity (U_p) decreases exponentially as the wave passes by, describing an elliptical trajectory [Meyers 1994]. An approximation of this type of wave is the motion profile on water surface upon a mass drop, as shown in Figure 2.3.

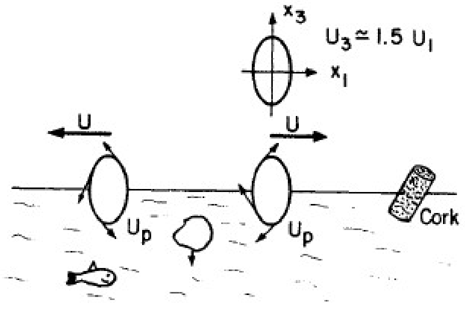


Figure 2.3: A mass drops in water, generating a wave. The cork in the right will move in a elliptical trajectory [Meyers 1994].

2.2 Propagation of Stress Waves

This dissertation analyses the propagation of stress waves in an armour system generated from a projectile impact. [Reis 2019] studied the formation of elastic waves generated from a projectile impact with a velocity of 5 m/s. This work has the purpose of investigating the propagation of elastic and plastic stress waves, testing ballistic impacts with increased velocity in order to generate plastic deformation.

Most materials display a linear stress-strain relation for low deformation. When stress surpasses the yield strength of the material, the relation between stress and deformation is no longer linear, affecting the dynamic behaviour of the system and stress wave propagation.

2.2.1 Elastic Stress Waves

The velocity of propagation of longitudinal elastic stress waves in linear-elastic solids such as a rod is

$$C_0 = \sqrt{\frac{E}{\rho}} \ , \tag{2.1}$$

where E is the modulus of elasticity (or Young's modulus) and ρ the density of the material.

In the case of an armour system, the structure can be defined as a semi-infinite media. For this type of media, the longitudinal velocity of propagation of the stress wave is given by [Meyers 1994]

$$C_\mathrm{L} = \sqrt{\frac{(1-\nu)}{(1+\nu)(1-2\nu)}\frac{E}{\rho}} \ , \tag{2.2}$$

where ν is the Poisson's ratio.

Similarly, the transverse wave speed can be written as

$$C_\mathrm{T} = \sqrt{\frac{E}{2(1+\nu)\rho}} \ . \tag{2.3}$$

2.2.2 Wave Transmission and Reflection

Developing layered media for energy absorption requires attention to the interaction between the layers in the configuration and how stress propagates. According to [Meyers 1994], considering the layered interface to be in equilibrium under the three stress pulses σ_I (incident), σ_T (transmitted) and σ_R (reflected), relating as

$$\sigma_\mathrm{I} + \sigma_\mathrm{R} = \sigma_\mathrm{T} \ . \tag{2.4}$$

Assuming the continuity of the interface, the respective particle velocities are

$$U_\mathrm{pI} + U_\mathrm{pR} = U_\mathrm{pT} \ . \tag{2.5}$$

The mechanical impedance of a material is [Macaulay 2012]

$$Z = \rho C \ , \tag{2.6}$$

where C is the velocity of the stress wave and ρ the density of the material. The conservation-of-momentum relationship states that [Meyers 1994]

$$\begin{aligned} F\mathrm{d}t &= \mathrm{d}(mU_\mathrm{p}) \ , \\ \sigma A\mathrm{d}t &= \rho A\mathrm{d}xU_\mathrm{p} \ , \\ \sigma &= \rho\frac{\mathrm{d}x}{\mathrm{d}t}U_\mathrm{p} \ , \\ \sigma &= \rho C U_\mathrm{p} \ , \end{aligned} \tag{2.7}$$

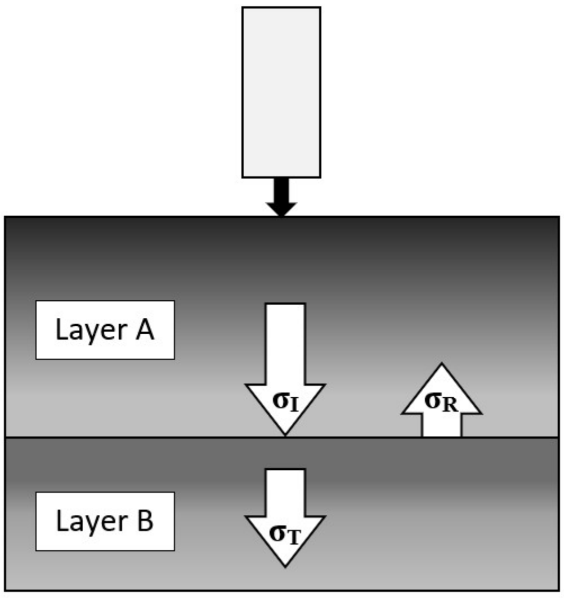


Figure 2.4: When a stress wave (σ_{I}) reaches an interface, it is reflected back as a different stress wave (σ_{R}) while still propagating into the adjacent media with different amplitude (σ_{T}).

where F, t and m are the force, time and mass, respectively, while A and x are the surface area and direction of the applied load, respectively. In a two-layered medium (layers A and B) such as in Figure 2.4, through Equation 2.7, the following particle velocities are obtained as

$$U_{\mathrm{pI}} = \frac{\sigma_{\mathrm{I}}}{\rho_{\mathrm{A}} C_{\mathrm{A}}} \ , \quad U_{\mathrm{pT}} = \frac{\sigma_{\mathrm{T}}}{\rho_{\mathrm{B}} C_{\mathrm{B}}} \ , \quad U_{\mathrm{pR}} = \frac{-\sigma_{\mathrm{R}}}{\rho_{\mathrm{A}} C_{\mathrm{A}}} \ . \tag{2.8}$$

Substituting Equation 2.7 into Equation 2.5 leads to

$$\frac{\sigma_{\mathrm{I}}}{\rho_{\mathrm{A}} C_{\mathrm{A}}} - \frac{\sigma_{\mathrm{R}}}{\rho_{\mathrm{A}} C_{\mathrm{A}}} = \frac{\sigma_{\mathrm{T}}}{\rho_{\mathrm{B}} C_{\mathrm{B}}} \ , \tag{2.9}$$

which, combined with Equation 2.4, yields:

$$\frac{\sigma_T}{\sigma_I} = \frac{2\rho_B C_B}{\rho_B C_B + \rho_A C_A} \ , \tag{2.10}$$

$$\frac{\sigma_R}{\sigma_I} = \frac{\rho_B C_B - \rho_A C_A}{\rho_B C_B + \rho_A C_A} \ . \tag{2.11}$$

The impedances of materials A and B (ρC product) determine the ratio of amplitudes of the transmitted and reflected pulses.

2.2.3 Plastic Stress Waves

The limit of elasticity in a material defines the maximum stress it can handle without deforming permanently. Material strength can be strain and strain-rate dependent [Meyers 1994]. The stress strain curve of metals can be represented by a bi-linear function such as the one in Figure 2.5a. The function is divided in two stages, the first one (blue) is elastic and the second one (red) is plastic. The stress-strain curve of some metals can also be represented by a power law function such that [Matusevich *et al.* 2012]

$$\sigma = K\varepsilon^n \ , \tag{2.12}$$

where K is the strength coefficient and n is the tensile strain-hardening exponent. This type of stress-strain curve is illustrated in Figure 2.5b.

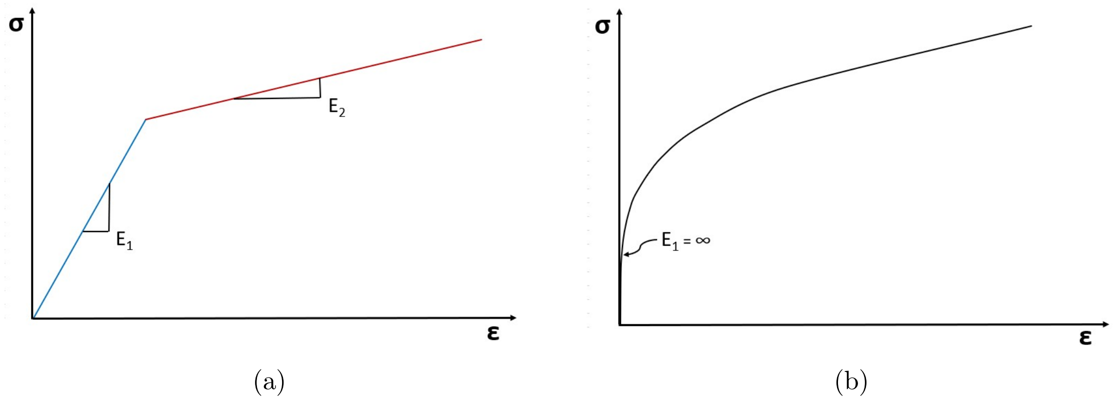


Figure 2.5: Stress-strain curves for ductile materials: (a) bilinear elastoplastic; (b) power law work hardening [Meyers 1994].

In Section 2.2.1, the velocity at which a stress wave propagated in finite medium is defined by Equation 2.1 and in a semi-infinite medium in Equation 2.2. The Young's modulus is the slope of the stress-strain curve in the initial stages of deformation:

$$E = \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon}\right)_{\mathrm{el}} \ . \tag{2.13}$$

Since the slope in the elastic range is steeper, by analysing Equation 2.1,

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon}\right)_{\mathrm{el}} > \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon}\right)_{\mathrm{pl}} \ , \tag{2.14}$$

which means that plastic waves propagate slower than elastic ones. The velocity of propagation of the stress wave for one dimensional cases is

$$C_{\mathrm{pl}} = \sqrt{\left(\frac{\mathrm{d}\sigma/\mathrm{d}\epsilon}{\rho}\right)} \ . \tag{2.15}$$

It is understandable how the velocity decreases by looking at Figure 2.6.

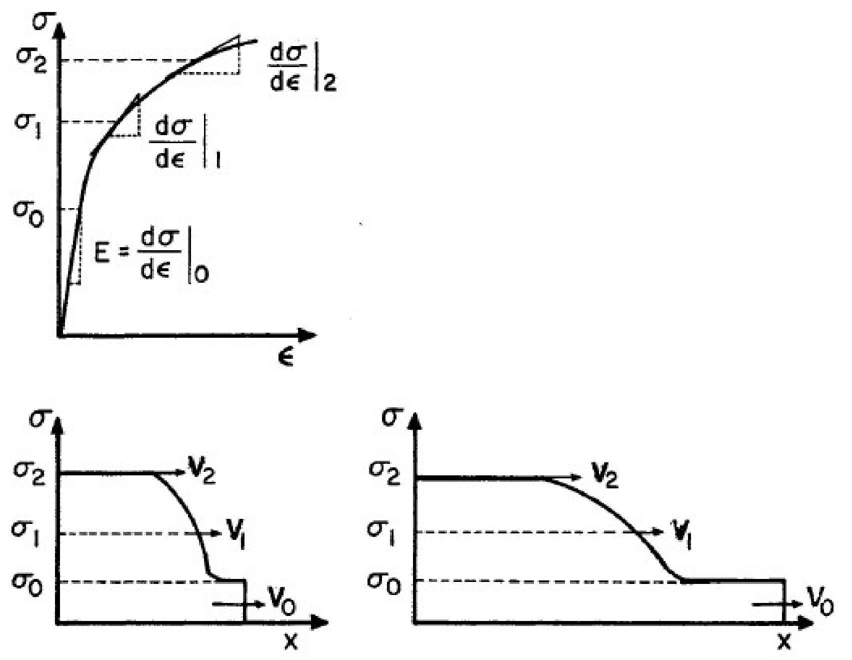

σ
σ_2
σ_1
σ_0
$\frac{d\sigma}{d\epsilon}\Big|_2$
$\frac{d\sigma}{d\epsilon}\Big|_1$
$E = \frac{d\sigma}{d\epsilon}\Big|_0$
ϵ
V_2
V_1
V_0
x

Figure 2.6: Shape of plastic wave front as a function of time; dispersion of wave is observed due to decrease in wave velocity with increasing stress [Meyers 1994].

Chapter 3

Model Generation

This dissertation targets the development of a ballistic impact model composed of a projectile and a multi-layered armour plate in Abaqus. Throughout this chapter, the model design and modelling techniques are discussed. Furthermore, the contents from the previous chapter are implemented and validated using finite element analysis, focusing in the elastic and plastic wave propagation.

Using a Python-Abaqus scripting interface, parameterising the model increases the speed of the process while also providing effortless ways of changing essential input data for model generation.

3.1 Model Design

The layout of the ballistic test is actually very simple. It consists of a minimum of two parts: a projectile and a layered armour system (which is composed by at least one plate). The configuration in this chapter is displayed with an assembly of a projectile and three plates: front, middle and rear. In the following chapters, the algorithm returns optimal configurations, with the number of plates to use as a variable. Nonetheless, a 3-plate configuration is enough to understand how the model is composed (Figure 3.1).

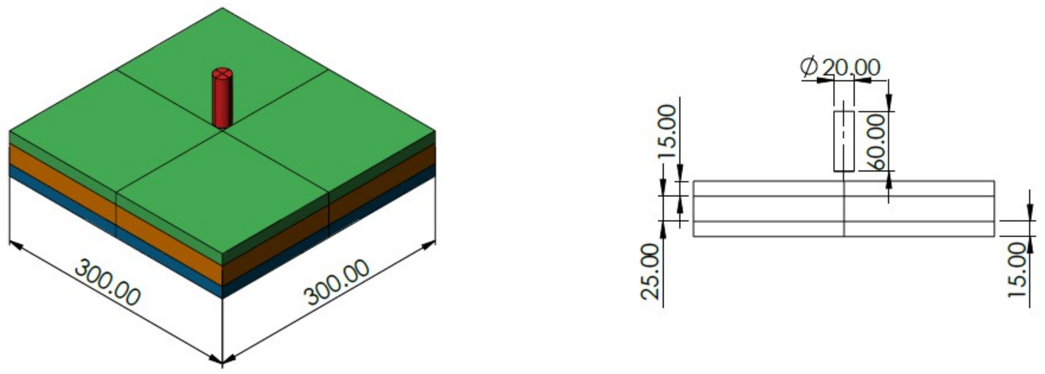


Figure 3.1: Model dimensions (in mm) and basic configuration.

Previous studies carried by [Pittman 2017] and [Reis 2019] serve as the starting point for further research on the ballistic impact phenomena. [Pittman 2017] designed his model with rectangular plates of 500 mm of length and 300 mm of width. While starting with those model dimensions, [Reis 2019] optimised the model for low velocity impacts, resulting in a design of square plates of 140 mm of width. With the purpose of performing ballistic tests with higher velocities and selecting boundary conditions that approximate to current ballistic test setups the model used in this dissertation features square plates of 300 mm of width. This enables a fraction of the impact energy to be absorbed by plate bending. The projectile is defined as a cylindrical 3D analytical rigid surface of 60 mm of length, diameter of 20 mm and has an assigned point of inertia correspondent to 0.147025 kg.

To decrease computational times, only a quarter of the plate is modelled and in order to maximise mesh efficiency, the geometry is divided into nine cells. The most important cells (labelled as 1 to 5 in Figure 3.2) define the area under impact and are used to analyse the propagation of stress waves. Overall, following the mesh structure presented by [García-González *et al.* 2015], the geometry is divided mainly in three zones. Zone A (red) defines the area under impact, which is twice the area of the projectile, Zone B (orange) defines the mesh transition zone between the impact zone and the rest of the plate, which is Zone C (yellow).

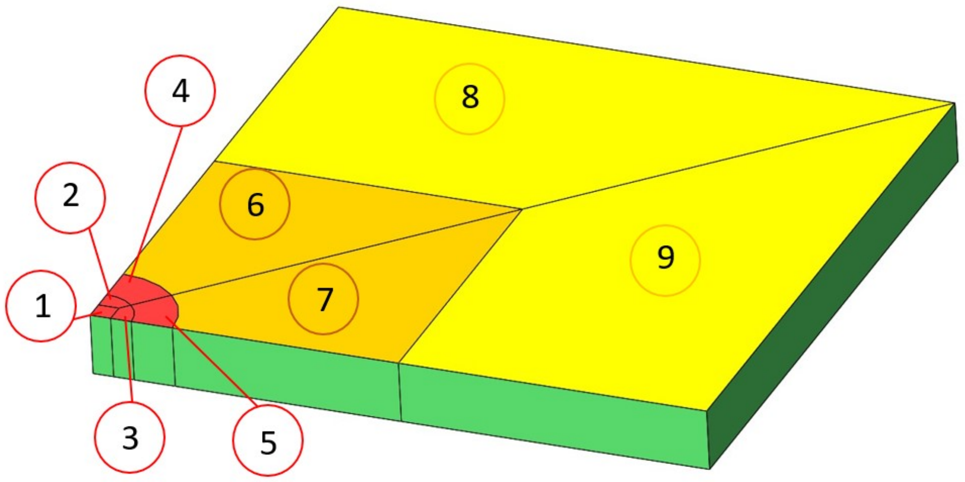


Figure 3.2: Each plate is divided in nine cells for easier analysis.

3.2 Python-Abaqus Scripting Interface

Methods for complex problem solving often involve iterative processes which can take advantage of multiple programming paradigms. Tools that allow quick iteration and easy interaction are essential for such processes [James 2018]. Python has a vast toolbox that provides data scientists with a large array of general- and special-purpose functionality.

The Abaqus Scripting Interface is an application programming interface (API) to the models and data used by Abaqus. It is an extension of the Python object-oriented programming language, relying on Python scripts [Systèmes 2011]. During the development of a model in Abaqus/CAE through the graphical user interface (GUI), commands are issued internally after every operation. The GUI generates commands in Python and sends them to the Abaqus/CAE kernel. The kernel processes all the information and displays it through the GUI. The Abaqus Scripting Interface provides the advantage of communicating every operation directly with the kernel through a script. The interaction between Abaqus Scripting Interface commands and the Abaqus/CAE kernel is represented in Figure 3.3.

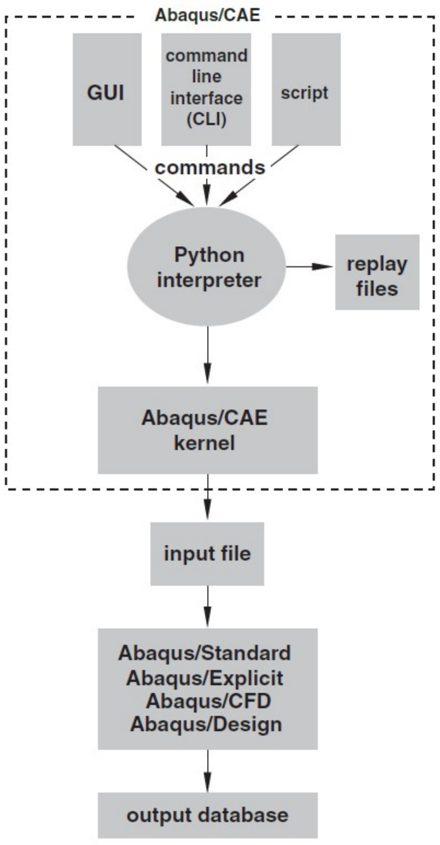


Figure 3.3: Abaqus Scripting Interface and Abaqus/CAE interactions flowchart [Systèmes 2011].

The scripting interface is a valuable tool for this dissertation. Not only serves as a support for the writing of optimisation algorithms (to discuss later in Chapters 4 and 5) but also optimises the whole model development process when considering a modular design approach. Through the programming of a Python script, the repetitive tasks during the building of the model in Abaqus/CAE GUI can be saved and later processed, generating information, such as a library of materials, reasonably fast. It also provides the advantage of generating a parametric study, which contributes to the development of a modular script containing a multitude of different parameters that can be edited effortlessly, saving valuable time. Similarly, a script can also contain useful parameters to perform repetitive operations when accessing results from the simulations, such as calculated stress levels. It is also possible to retrieve data and post process it from the Python script, writing it to a text file and loading it in a post processing software. The script used in this dissertation can be found in Appendix B.

3.3 Materials

A library of materials is predefined in the script, meaning that a wide selection of materials is available for any type of desired study in Abaqus. Each material is defined both for elastic and plastic regime. The material database is essential for the optimisation part of the dissertation since the algorithm returns the optimal material selection for all plates used in the ballistic simulation.

Essentially, the list of materials to use are split in two main classifications: elastic and elasto-plastic materials. While the dissertation focuses more on elasto-plastic behaviour, it is also important to understand the behaviour of the materials studied by [Reis 2019]. The materials used in his research were applied for low energy impacts, which naturally display elastic behaviour when subjected to low stress. The materials to be considered in low energy impacts are detailed in Table 3.1 [Pittman 2017, Reis 2019].

Table 3.1: Materials and respective mechanical properties for elastic stress waves tests.

Material	Density (kg/m^3)	Young's Modulus (GPa)	Yield Strength (MPa)	Poisson's Ratio
Aluminium	2700	70	276	0.33
Nylon-6	1140	3	82	0.35
Steel	7850	200	350	0.25
EPDM	960	2.5×10^{-3}	16.8	0.499
Cork	293	9	1	0.3
Aluminium Foam	410	103.08×10^{-3}	1.24	0.05
Polycarbonate	1300	1.8	63	0.3182
Epoxy	1540	3.5	15	0.33
Titanium	4430	113.8	880	0.342

This work intends to analyse the propagation of plastic stress waves. This implicates that the ballistic impact generates stress levels that surpass the yield strength of the material, resulting in plastic strains. In Chapter 5, the optimisation procedures feature the elasto-plastic materials that are listed in Table 3.2. The properties of these materials are extracted from literature and to define the plastic behaviour of each one of them, the stress-strain plot curves are shown in Figures 3.4 and 3.5.

Table 3.2: Materials and respective mechanical properties for elastic and plastic stress waves tests.

Material	Density (kg/m^3)	Young's Modulus (GPa)	Yield Strength (MPa)	Poisson's Ratio
Aluminium 1100	2710	68.9	55	0.33
Nylon-6	1140	3	100	0.35
AISI 4340 Steel	7850	200	1355	0.29

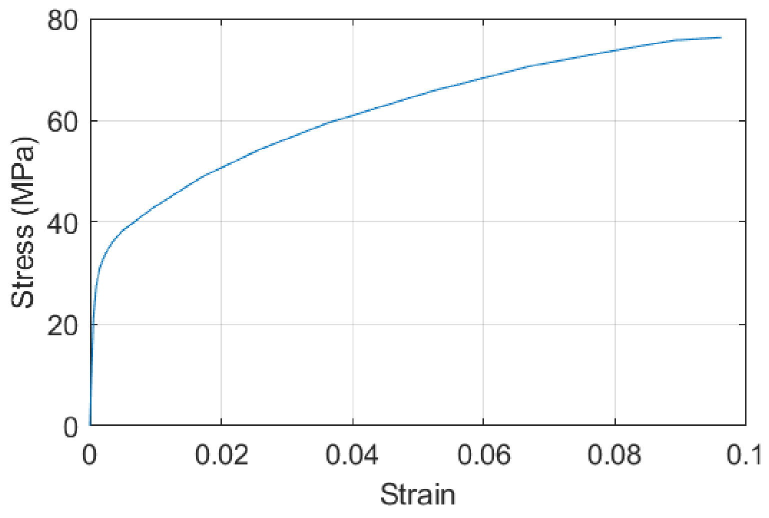


Figure 3.4: True stress-strain curve for the 1100 series aluminium alloy [Li and You 2019].

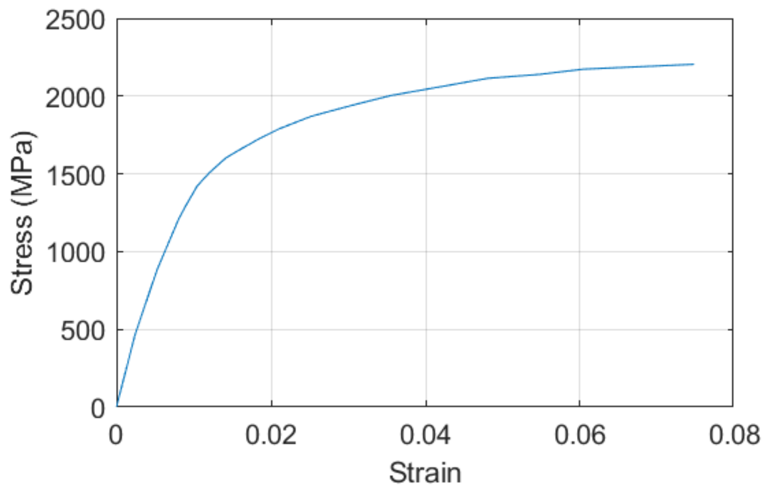


Figure 3.5: True stress-strain curve for the AISI 4340 Steel [Li *et al.* 2019].

3.4 Numerical Simulation

In this section, the conditions of the simulation are set. Important details about boundary conditions, duration of simulation and contacts are defined. The setup of the results is configured throughout the following sub-sections.

3.4.1 Boundary Conditions

In Section 3.1, it is mentioned that only a quarter of the plate geometry is modelled. This helps reducing the amount of total finite elements to use in calculations, which reduces the duration of the simulation. Using symmetry conditions to the surfaces in planes OXZ and OYZ (Figure 3.6), it is possible to perform the calculations for a quarter of the model without compromising results.

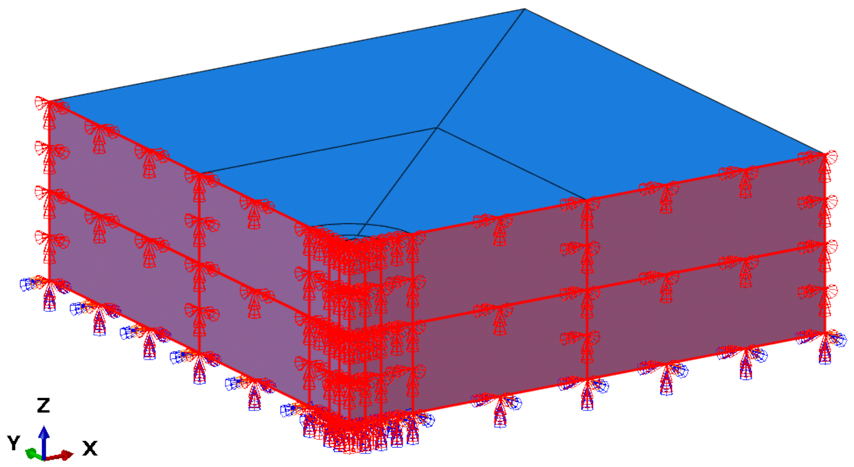


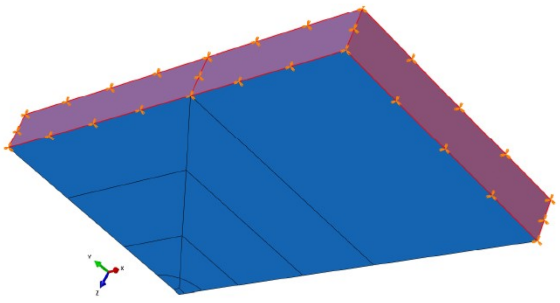
Figure 3.6: Surfaces chosen for the symmetry boundary condition.

Displacement boundary conditions need to be applied to the geometry. In the dissertation written by [Reis 2019], a study was carried out to find the optimal boundary condition to suppress linear motion of the plates post projectile impact. For that purpose, tests were performed on three different clamped geometries, shown in Figure 3.7.

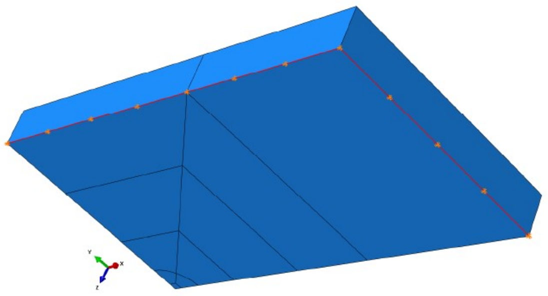
After analysing the resultant stress on the rear surface for all the three studied cases, throughout the entire impact period, Figure 3.7c (third scenario) provided more uniform results throughout the entire impact time period. The explanation is that this particular boundary condition minimises the propagation of stress waves in the OX and OY directions. Moreover, the propagating stress waves in the OZ direction suffer less interference when comparing to the results from the other boundary conditions [Reis 2019]. Under these circumstances, the results obtained for the third scenario are more reliable for interpretation and to analyse the propagation of elastic stress waves, thus this was the chosen configuration for his analysis.

This dissertation is focused on carrying on from the previous study by [Reis 2019], introducing plasticity effects during stress wave propagation. In order to generate plastic deformation, higher velocity impacts have to be considered. The third clamped configuration limits strain effects in all directions, which has a considerable influence in stress wave propagation and material deformation. Furthermore, in the interest of approaching the model design in Abaqus to the regular experimental ballistic test setup, similar to a backing material fixture used in National Institute of Justice (NIJ) standards [NIJ Standard-0101.062008], the first clamped configuration (Figure 3.7a) will be used as the main configuration for the projectile impacts.

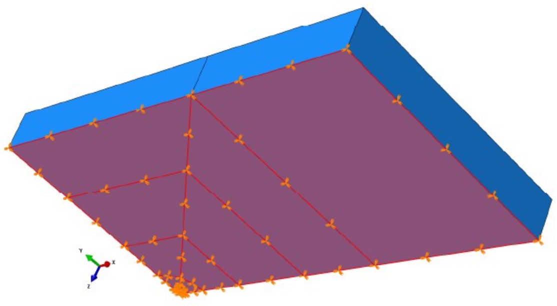
The next and final step is to define the projectile velocity as initial condition. This



(a)



(b)



(c)

Figure 3.7: Three boundary condition studies [Reis 2019]: (a) plate clamped by the side surfaces of the plate, (b) plate pinned at the side edges of the rear surface of the plate, (c) plate clamped at the rear surface of the plate.

velocity is defined by a vector pointing in the OZ direction (towards the plate) and selected magnitude. The projectile linear velocity vector is constrained exclusively to that direction.

3.4.2 Transient Analysis Setup and Part Interactions

The simulation is focused on a time-based event, with the software calculating the generated stress levels on the plate upon the impact of the projectile. It is necessary to designate the duration of the event and a results output frequency.

The last conditions of the simulation refer to the behaviour between the different parts of the assembly, which contains one projectile and one or more plates. [Reis 2019] researched the influence of different interactions combinations and concluded that "surface-to-surface" contact algorithm between the projectile and front surface of the first plate and "tie" constraint between surfaces of adjacent layers was the combination that required less computational processing, thus the interactions and constraints are defined according to his studies.

3.5 Mesh Parameters

The level of discretisation defines the computational cost of the study. The optimal configuration consists in a mesh sufficiently refined to be able to reproduce the propagation of stress waves while also minimising the overall computational times.

The smallest element in the mesh defines the stable time increment necessary to which the calculations converge. To determine the minimum element size to use in the mesh, it is important to verify the Courant-Friedrichs-Lewy condition, which is defined by [Lin 1996]

$$C_{\mathrm{CFL}} = C\frac{\Delta t}{\Delta x} \leq (C_{\mathrm{CFL}})_{\max}. \tag{3.1}$$

C_{CFL} is the Courant number, which for explicit analyses must be lower or equal to one. To calculate the Courant number, each time increment (Δt) must be known as well as the minimum length of the smallest element in the mesh (Δx) and the speed of the travelling wave.

Testing for a wave speed on aluminium (properties in Table 3.1), from Equation 2.2, a propagation velocity of $C = 6198$ m/s and with a projectile impact at 5 m/s, the duration of impact is $5.94 \times 10^{-}5$ s, as seen in Figure 3.8. To capture the stress wave accurately, the mesh needs to be sufficiently refined. Multiplying the wave's speed by the duration of the impulse it is possible to calculate the length of the wave. Having $L_{\mathrm{wave}} = C_{\mathrm{L}} \times \Delta t_{\mathrm{pulse}} = 6198 \times 5.94 \times \ 10^{-5} \ = 0.368$ m, it is noticeable that the resulting length is more than the thickness of the plate. This means that it is possible to accurately capture the stress wave regardless of the number of elements. However, the Courant-Friedrichs-Lewy condition must still be verified.

The smallest elements are present in partition 1 (Figure 3.2), for which the smallest length is 1 mm. Using Equation 3.1, the Courant number can be calculated as $C_{\mathrm{CFL}} = 6198 \times \frac{5.0\times10^{-8}}{1\times10^{-3}} = 0.31$, which is lower than one, hence the conditions are valid and it is possible to proceed with the minimum size of 1 mm for the mesh.

For faster processing, the ideal mesh consists of eight-node brick elements with reduced integration (C3D8R). However, this type of element uses one fewer integration point in each direction than the fully integrated elements. This causes that the "surface-to-surface" contact between projectile and plate generates instability when extrapolating stress values in the area under impact. The fully integrated eight-node brick element

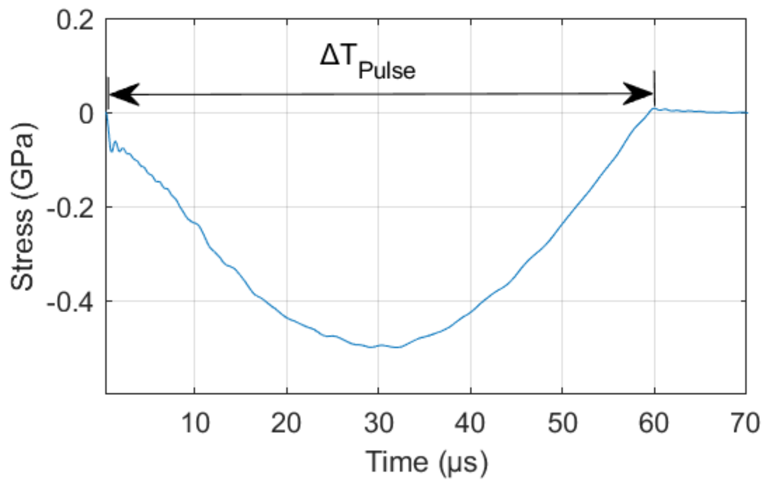


Figure 3.8: Average Stress in the Frontal Impact Region.

(C3D8) fixes the instabilities at the expense of slower calculations. Under these circumstances, cells in Zone A (Figure 3.2) feature C3D8 elements while the remaining feature C3D8R elements. The mesh is shown in Figure 3.9.

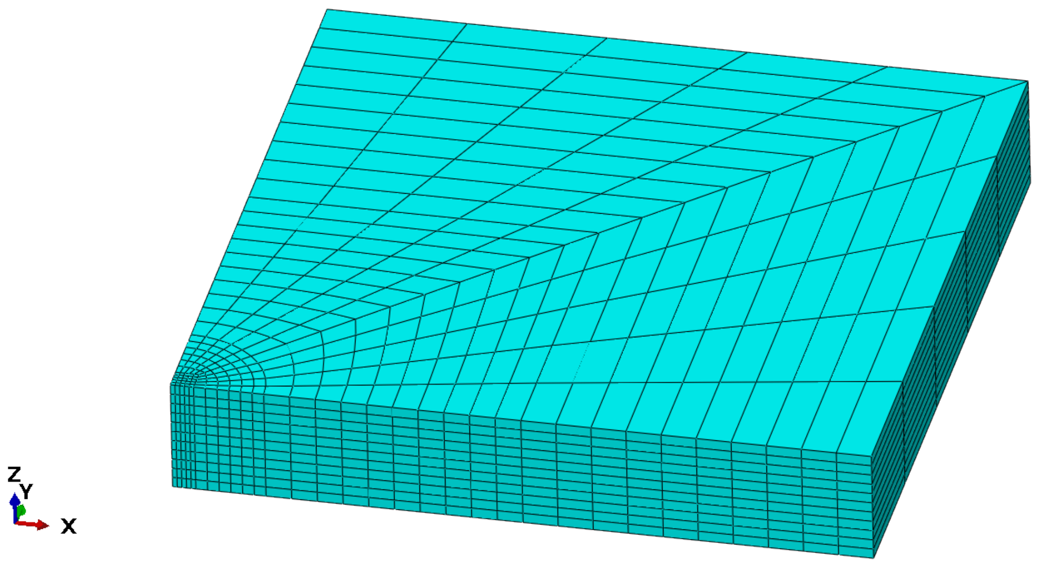


Figure 3.9: Example of the mesh for testing.

3.6 Model Validation

3.6.1 Elastic Wave Speed

A series of tests are conducted to analyse the dynamics of wave propagation discussed in Chapter 2. For a square aluminium plate with a length of 300 mm and a 25 mm thickness, measuring the average stress of the elements in the front and rear impact regions, shown in Figure 3.10, a 5 m/s projectile impact generates the stress plots in Figure 3.11. Furthermore, the stress is analysed exclusively from the OZ direction and since the plate thickness is equal to 25 mm, by measuring the time that the stress wave takes to travel between the front and rear surfaces, it is possible to calculate the stress wave's speed.

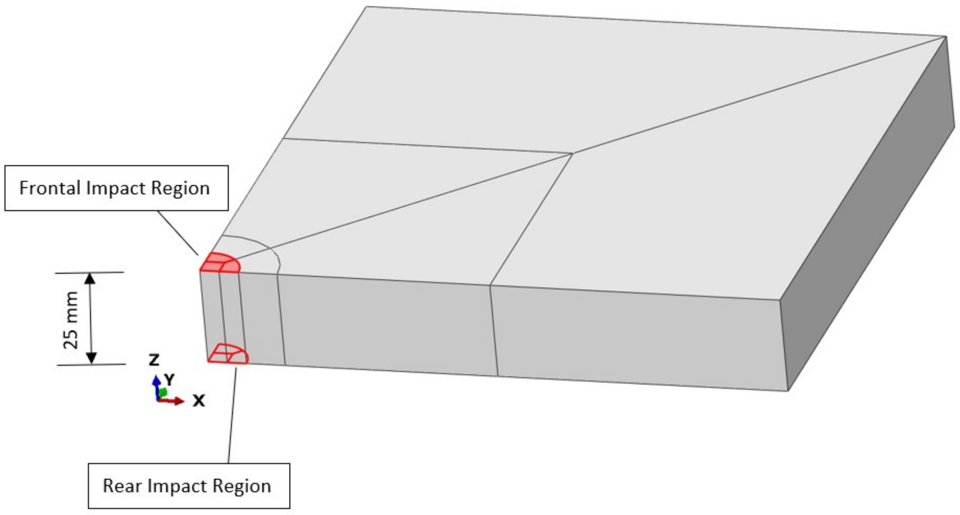


Figure 3.10: Front and rear impact regions selected for elastic stress wave speed validation tests.

Two sets of results are stored to calculate the stress wave speed and both are measured for the two analysed impact regions. The first set contains information relative to the generated first stress peak while the other one refers to the end of the pulse itself. Tables 3.3 and 3.4 list the results, with T_{front} and T_{rear} being the instants for the set of results for the measured stress in the front and rear surfaces, respectively.

Table 3.3: Data for stress wave speed calculation.

Set	T_{front} (μs)	T_{rear} (μs)	$\Delta T_{r,f}$ (μs)
First Stress Peak	0.90	5.15	4.25
End of Pulse	59.85	63.95	4.10

Having the time instants, from a 25 mm distance between front and rear surface, it is possible to calculate the velocity of the wave dividing the total travelled distance with the time the wave took to reach the rear surface, $\Delta T_{r,f}$ (Table 3.3). The calculated velocity for the stress wave is listed in Table 3.4.

To help visualise how the stress propagates throughout the thickness of the plate,

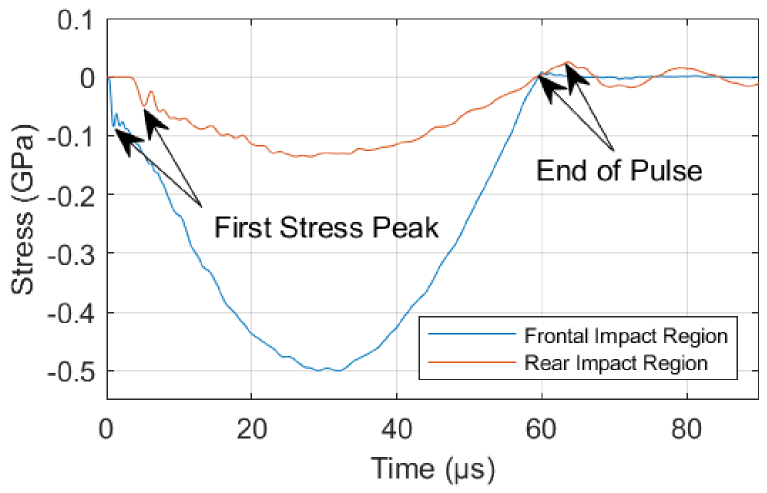
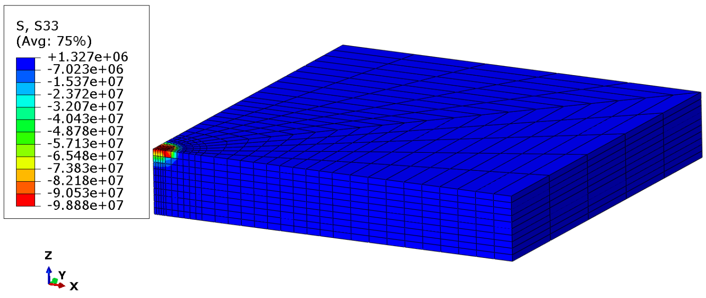


Figure 3.11: Average stress measured on the frontal impact region and rear impact region for an impact velocity of 5 m/s.

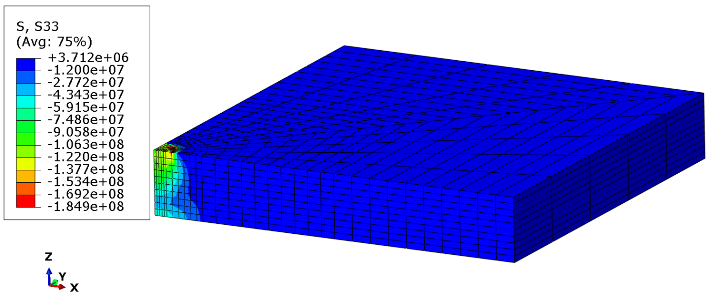
Table 3.4: Calculated stress wave speed.

Set	$C_{\mathrm{L,calc.}}$ (m/s)	$C_{\mathrm{L,t}}$ (m/s)	Deviation (%)
First Stress Peak	5882.35	6197.82	5.10
End of Pulse	6097.56	6197.82	1.62

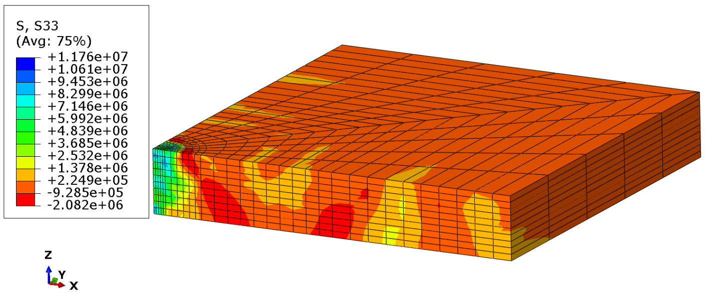
Figure 3.12 represent the stress in the OZ direction at the instants listed in Table 3.3. It is possible to see from Figure 3.12a the peak of stress on the front impact region and subsequently the progression of the wave towards the rear of the plate. By the end of the pulse, the stress oscillations on the plate in Figures 3.12c and 3.12d are due to the free vibration of the system post impact.



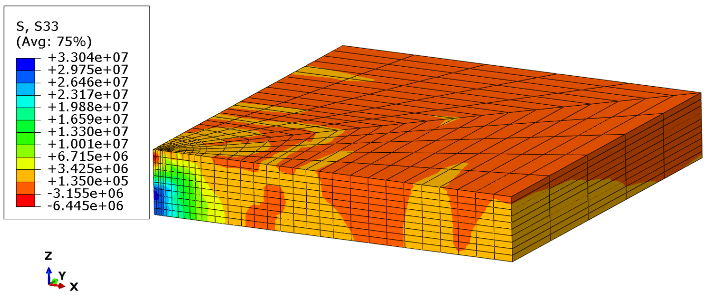
(a)



(b)



(c)



(d)

Figure 3.12: Stress state of the geometry at the four different analysed times: (a) first stress peak measured at the front impact region ($t = 0.90\ \mu$s), (b) first stress peak measured at the rear impact region ($t = 5.15\ \mu$s), (c) end of pulse measured at the Front Impact Region ($t = 59.85\ \mu$s), (d) end of pulse measured at the Rear Impact Region ($t = 63.95\ \mu$s).

3.6.2 Wave Interaction between Layers

Upon finding a new surface, a stress wave continues to propagate through the new media as a fraction of the original stress wave. In Equation 2.4 and Figure 2.4 it is demonstrated how the stress wave behaves when it reaches an interface.

The stress wave is similar to a square wave and, for data acquisition in terms of stress magnitude, the analyses are run through three different stages of the wave. The first stage is upon the contact between the projectile and the plate, where it generates a stress disturbance, followed by a second stress peak where the stress levels stabilise for a determined amount of time, which is the second stage. The final stage starts after the projectile transfers its initial kinetic energy into the armour system, resulting in strain energy from deformation. Strain energy resulting from elastic deformation will be transferred back into the projectile under the form of kinetic energy, ultimately contributing to the projectile moving away from the armour system at constant velocity. At this stage, the stress levels decrease towards the final stress condition, depending on the amount of deformation of the material. Post impact, the stress on the plate results from plate vibration, however, depending on the plastic strain in the material, residual stresses may be generated. The stages of the stress wave analysed can be better understood through the graphical visualisation of Figure 3.13.

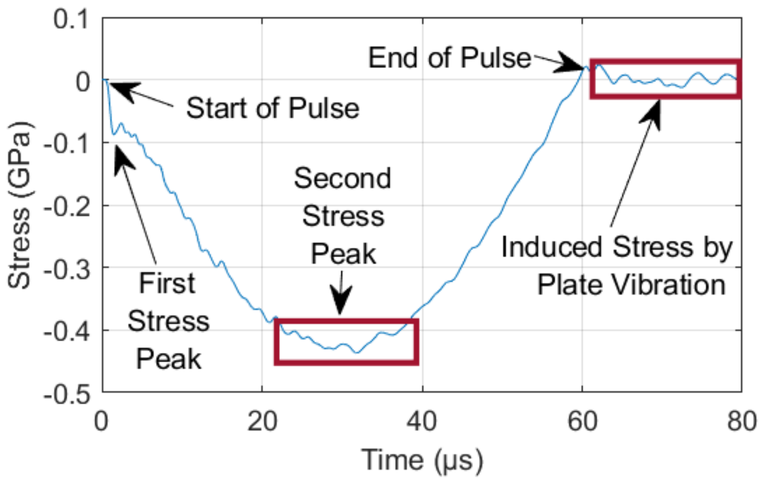


Figure 3.13: A generic stress wave generated by the impact of a projectile in a square metal plate.

Modelling in Abaqus the same example as for the demonstration of elastic stress waves in the previous sub-section, another plate was added at the back of the original. The setup for this section's validation model is composed by two square plates of 300 mm of length with the top and bottom plates having a thickness of 20 mm and 10 mm, respectively. The conditions for the simulation are defined in Section 3.4 and to perform the tests on how the stress wave propagates between different material layers, three distinct impact velocities are used: 5, 40 and 100 m/s. Taking into account the spectrum of chosen velocities, the encastre configuration is the clamping of the side surfaces of the plates shown in Figure 3.7a, due to the considerable amount of longitudinal strain generated from the 40 and 100 m/s projectile impacts.

Measuring the magnitude of the first and second stress peaks on adjacent surfaces, specifically on the surface of cell 1, it is possible to analyse how they compare between the contact region of each plate. To simplify denominations, the top plate will be referred as Plate A and the bottom plate as Plate B. For comparable results, the tests to be conducted include all combinations of materials on Plate A and Plate B, from the range of materials listed in Table 3.5, which are extracted from Table 3.2. It is important to note that the following materials are modelled according to the perfectly plastic material model and that their respective longitudinal stress wave speeds are calculated using Equation 2.2.

Table 3.5: Properties and theoretical longitudinal stress wave speeds for the materials to use in wave interaction analysis.

Material	Density (kg/m^3)	Young's Modulus (GPa)	Poisson's Ratio	Yield Stress (MPa)	Stress Wave Speed (m/s)
Steel	7850.0	200.0	0.29	972.0	5778.2
Aluminium	2700.0	68.9	0.33	276.0	6148.9
Titanium	4430.0	113.8	0.342	880.0	6313.4

The tests conducted feature all combinations between the three materials and return the stress wave on the rear central impact region of Plate A and the front central impact region of Plate B. The first and second stress peaks are measured and with results retrieved from both impact regions it is possible to determine the transmitted and incident stresses, σ_T and σ_I. The incident stress is the one measured on Plate A while the transmitted is the one on Plate B. The region used for measurement is depicted in Figure 3.14. The stress wave returned is the average between the stresses measured in all finite elements which belong to the referred region.

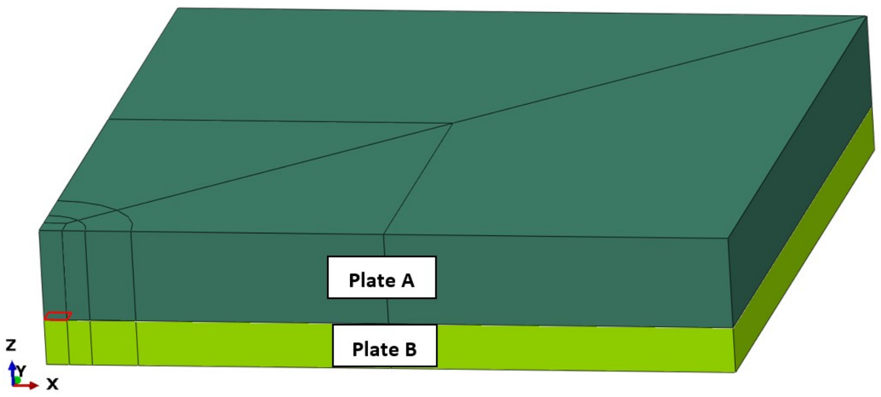


Figure 3.14: Example of the square impact region used for the analysis ran for studies on wave interaction between the different layers of materials.

An example of the analysed stress in the central impact region directly at the interface of the two plates is shown in Figure 3.15. This particular plot results from the setup of a steel Plate A and aluminium Plate B, impacted at 40 m/s. The plot draws the stress wave for the contact region in Plate A and in Plate B. From Figure 2.4 it is possible to infer that the incident stress is measured in the contact region in Plate A while the transmitted stress is measured in the contact region in Plate B. Using Equation 2.10 and the properties of the materials in Table 3.5 it is possible to calculate the theoretical ratio of transmitted stress to incident stress. Two different instants, one for the first stress peak (green) and one for the second stress peak (blue), are selected for the measured stress in the two contact regions. Dividing the two magnitudes for each of the stress peaks it is possible to compare the results with the theoretical values.

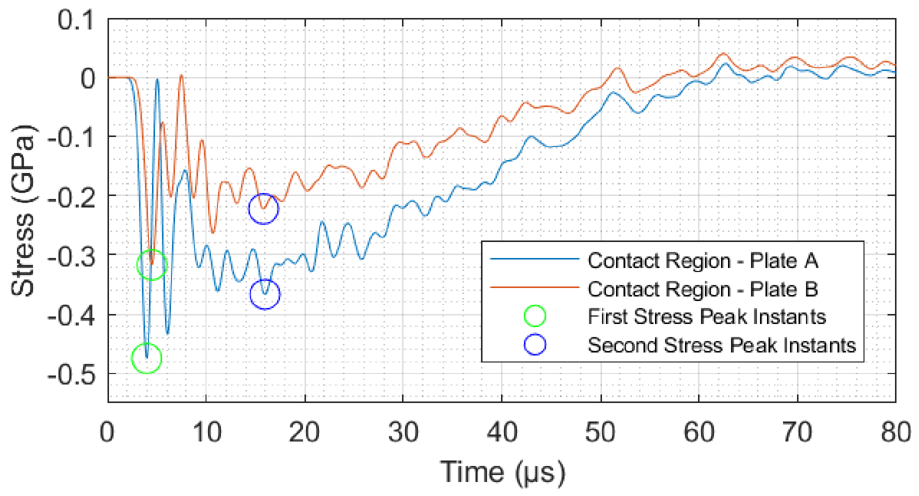


Figure 3.15: Example of the chosen instants for the analysis of the relation between transmitted and incident stresses, with the green and blue circles marking the sets of data for first and second stress peaks, respectively.

The results for the calculated relation between transmitted and incident stresses are listed in Tables 3.6 and 3.7, along with the theoretical relation. The first table evaluates the relation for the first stress peak while the second table focuses on the second stress peak.

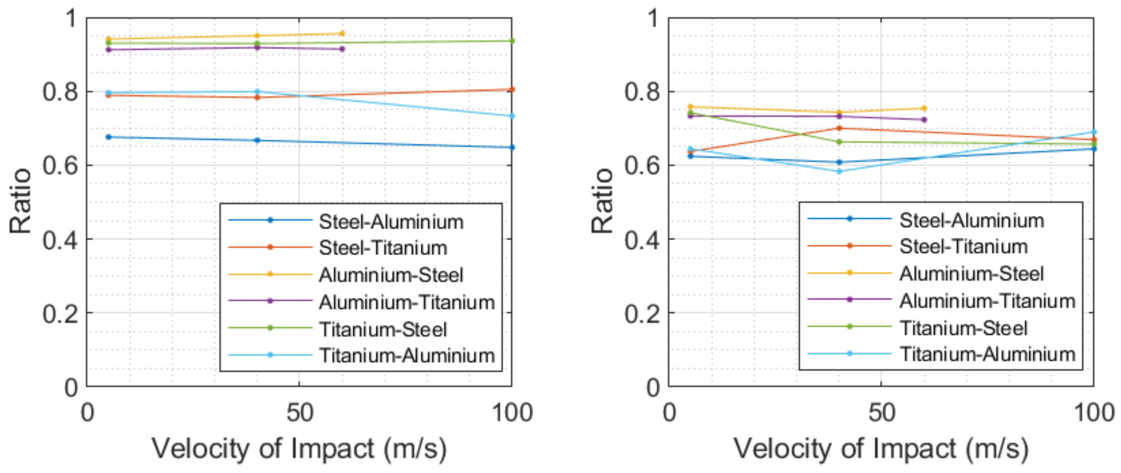
Table 3.6: Transmitted and incident stress relation based on the first stress peak, calculated for different material combinations.

Material of Plate A	Material of Plate B	$(\sigma_T/\sigma_I)_t$	$(\sigma_T/\sigma_I)_{calc.}$	Deviation (%)
Steel	Aluminium	0.54	0.66	23.9
Steel	Titanium	0.76	0.79	3.88
Aluminium	Steel	1.46	0.95	35.2
Aluminium	Titanium	1.26	0.91	27.1
Titanium	Steel	1.24	0.93	24.7
Titanium	Aluminium	0.74	0.78	4.16

The combinations with aluminium assigned on Plate A culminated in higher deviation of the results. For the aluminium as material for Plate A, it is not possible to conduct the test with an impact velocity of 100 m/s due to high distortion ratio between finite elements, thus a replacement by an impact velocity of 60 m/s is done for these cases. Through Figures 3.16a and 3.16b, it is seen that, despite the differences in impact velocity, the relation between transmitted and incident stresses is not affected significantly.

Table 3.7: Transmitted and incident stress relation based on the second stress peak, calculated for different material combinations.

Material of Plate A	Material of Plate B	$(\sigma_T/\sigma_I)_t$	$(\sigma_T/\sigma_I)_{calc.}$	Deviation (%)
Steel	Aluminium	0.54	0.63	16.7
Steel	Titanium	0.76	0.67	12.3
Aluminium	Steel	1.46	0.75	48.7
Aluminium	Titanium	1.26	0.73	41.9
Titanium	Steel	1.24	0.69	44.4
Titanium	Aluminium	0.74	0.64	14.2



(a) (b)

Figure 3.16: Relation between transmitted and incident stress across different impact velocities: (a) measured at the first stress peak, (b) measured at the second stress peak.

3.6.3 Plastic Stress Wave Speed

The procedure implemented to test the velocity of propagation of the plastic stress wave is equivalent to the test on the elastic stress wave in Section 3.6.1. For this test, a fictional steel is modelled with plastic strains from the stress-strain power law in Equation 2.12 with a strength coefficient of $K = 2500$ MPa and the tensile strain hardening exponent $n = 0.15$. To implement this material in Abaqus, its elastic and plastic properties must be defined. The elastic properties are listed in Table 3.9 while the plastic properties are extracted directly from the stress-strain curve shown in Figure 3.17, where the yield strength of the material is 1160 MPa .

Table 3.8: Elastic properties of the fictional steel to use in plastic wave speed validation.

Material	Density (kg/m^3)	Young's Modulus (GPa)	Poisson's Ratio
Fictional Steel	7850	200	0.33

As described in Chapter 2, elastic stress waves propagate faster than plastic waves. From Equation 2.2 it is possible to calculate the theoretical longitudinal stress wave

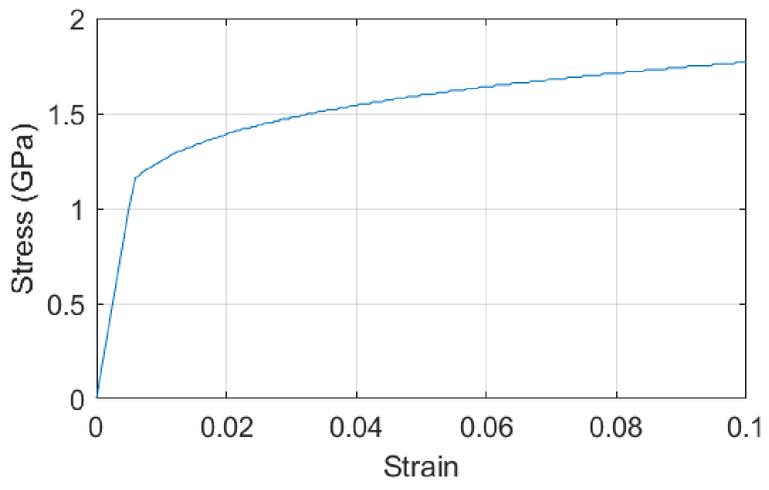


Figure 3.17: Stress-strain plot of the fictional steel used for the velocity test of plastic stress waves.

speed. From Equation 2.15 it is possible to calculate the velocity of the stress wave, which is plotted in Figure 3.18 as a function of stress. Upon reading the yield stress of the material, the plastic stress wave is formed, propagating at a different velocity than the elastic stress wave.

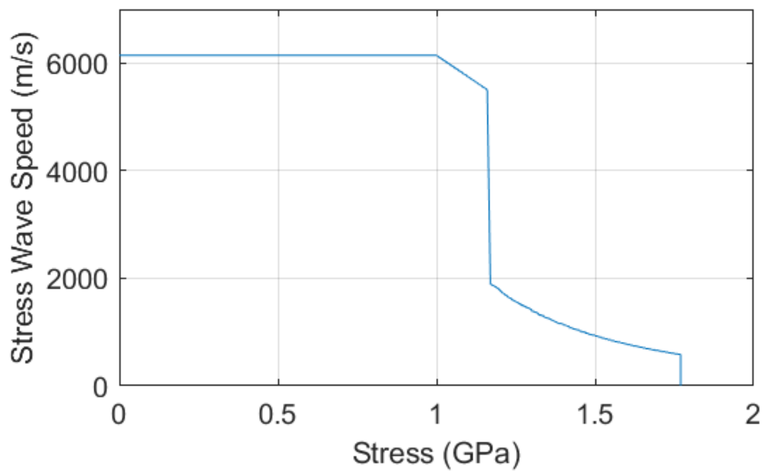


Figure 3.18: Stress wave speed as a function of the induced stress.

To validate the velocity of plastic stress waves, two finite elements are chosen to extract data from. The model to use is exactly the same as the one from Section 3.6.1, except for the material used and the magnitude of the projectile's velocity, which for this test is 40 m/s. Two finite elements are chosen to measure the elapsed time of the first and last peaks of the stress wave. The stress is calculated in the integration points of the elements, hence the distance between the two elements is measured from their centroid. The chosen finite elements are represented in Figure 3.19 as well as their designations. The reason behind the choice of the second element from the top is to avoid the interference of the contact between finite elements upon the impact of the projectile.

In Abaqus, the plasticity model defines plastic yield in terms of von Mises stress.

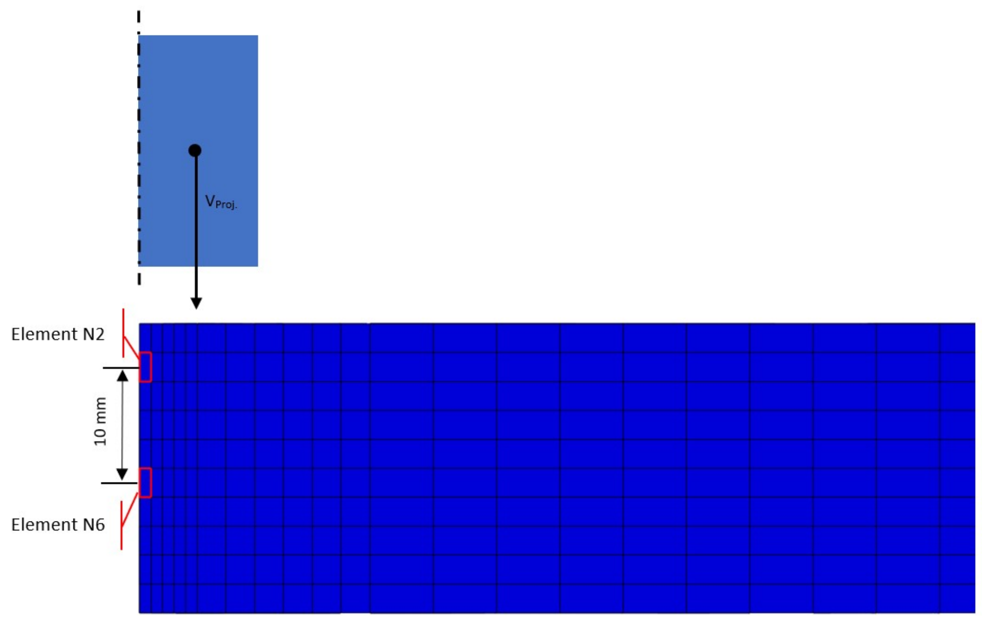


Figure 3.19: Chosen finite elements for the velocity test of plastic stress waves.

For this reason, measured stress in the OZ direction may be higher than the material's maximum defined stress of ~ 1.77 GPa, from the stress-strain plot. The von Mises yield criterion can be expressed as [Dowling *et al.* 2013]

$$\sigma_{\mathrm{VM}} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_{\mathrm{x}}-\sigma_{\mathrm{y}})^2+(\sigma_{\mathrm{y}}-\sigma_{\mathrm{z}})^2+(\sigma_{\mathrm{z}}-\sigma_{\mathrm{x}})^2+6(\tau_{\mathrm{xy}}^2+\tau_{\mathrm{yz}}^2+\tau_{\mathrm{zx}}^2)}\ . \tag{3.2}$$

The measured stress is shown in Figure 3.20 and it should be noted that, in some instants, the magnitude for the two elements is higher than the maximum stress in the stress-strain plot.

The measured von Mises stress cannot exceed the maximum values from the stress-strain plot in Figure 3.17. The von Mises stress extracted for the two selected elements is shown in Figure 3.21 and it is possible to see that over the time of the event the magnitude did not exceed the maximum stress from the stress-strain plot. Moreover, the instants of the detected peaks in the von Mises stress plot coincide with the ones detected in the plotted stresses in the OZ direction, in Figure 3.20. In the figure two stress curves are plotted, one for each element. Measuring the time that the stress wave takes to travel from Element N2 to Element N6, the stress wave speed can be calculated. For that purpose, two instants where the stress is higher than the yield stress are selected.

The results are extracted for two instants, the first stress peak and the last stress peak. Tables 3.9 and 3.10 list the results, having $T_{\mathrm{FE,N2}}$ and $T_{\mathrm{FE,N6}}$ being the instants for the set of results for the measured stress in Element N2 and Element N6, respectively. The distance of the elements' centroids is 10 mm, having ΔT_{10mm} as the time taken for the stress wave to travel from one element to the other.

From Table 3.9 it is possible to see the time it took for the stress wave to travel from Element N2 to Element N6. Table 3.10 lists the results for the wave speed measured at the first and last stress peaks.

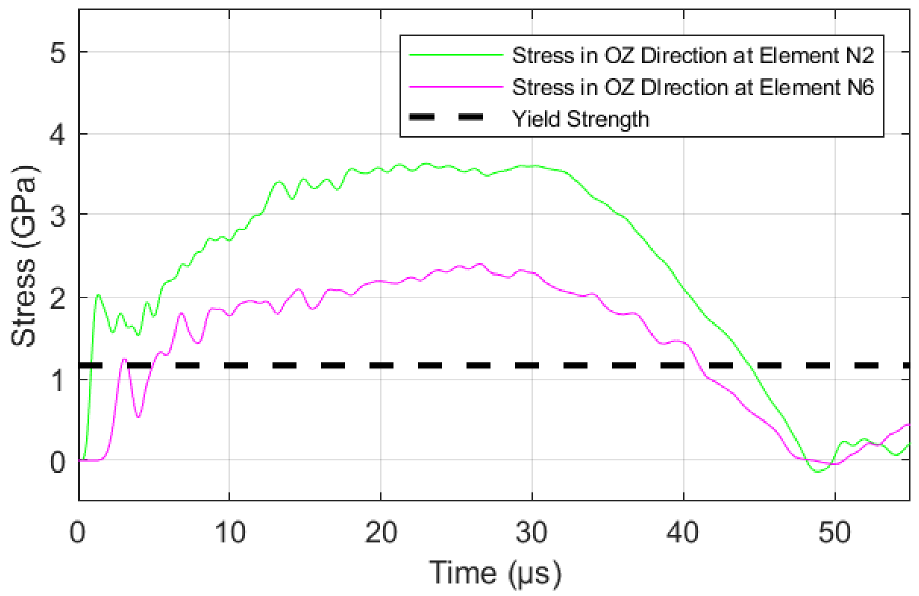


Figure 3.20: Measured stress in the OZ direction of the two selected finite elements.

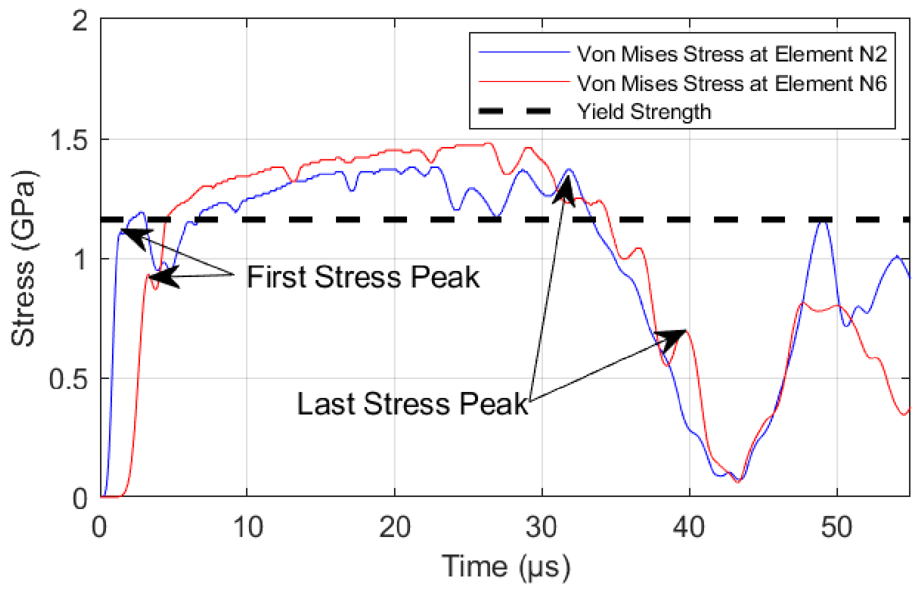


Figure 3.21: Measured von Mises stress of the two selected finite elements.

Table 3.9: Data for plastic stress wave speed calculation.

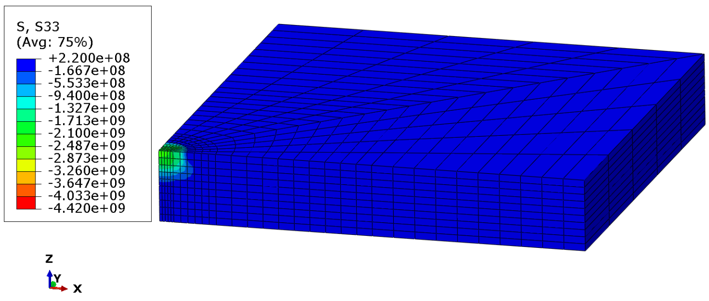
Set	$T_{\mathrm{FE,N2}}$ (μs)	$T_{\mathrm{FE,N6}}$ (μs)	ΔT_{10mm} (μs)
First stress peak	1.458	3.190	1.732
Last stress peak	31.790	39.680	7.890

In Table 3.10 the magnitude of the first stress peak is lower than the yield strength of the material, which means the stress wave at that instant is elastic. The magnitude of the last stress peak, however, is higher than the yield strength of the material and thus,

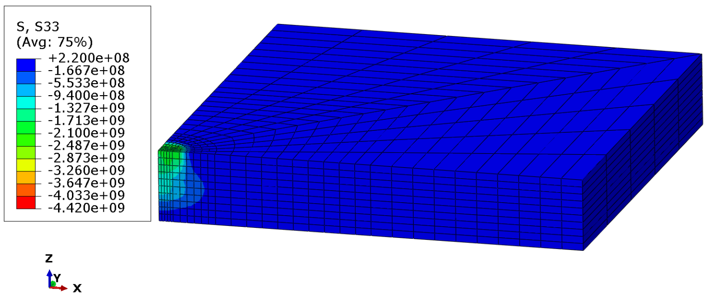
Table 3.10: Calculated plastic stress wave speed.

Set	Stress (GPa)	$C_{\mathrm{L,calc.}}$ (m/s)	$C_{\mathrm{L,t}}$ (m/s)	Deviation (%)
First stress peak	1.11	5773.67	6144.02	6.03
Last stress peak	1.37	1267.43	1198.27	5.77

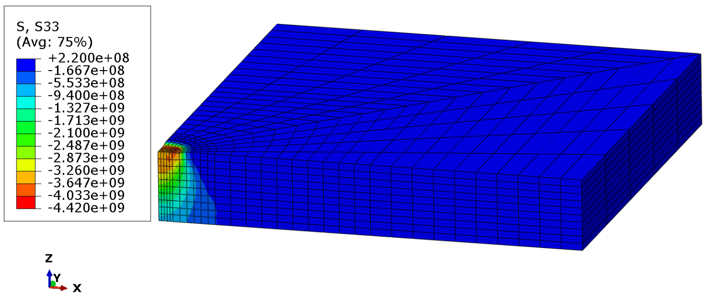
at that instant, it is a plastic stress wave propagating. To help visualise how the stress propagates throughout the thickness of the plate, Figure 3.22 represent the generated stress in the OZ direction at the analysed instants in Table 3.9. It is possible to see the plastic deformation at the front surface of the plate and the progression of the stress wave towards the rear surface of the plate.



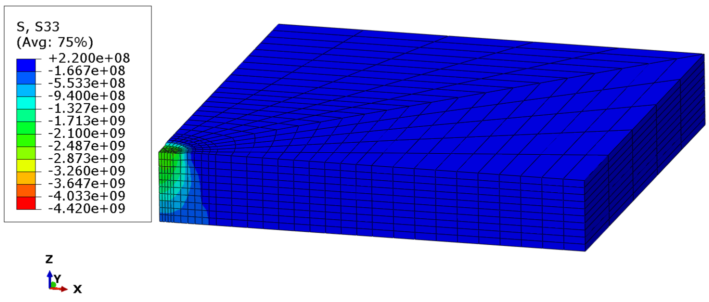
(a)



(b)



(c)



(d)

Figure 3.22: Stress state on the plate at the four different analysed times: (a) first stress peak measured at Element N2 ($t = 1.458$ μs), (b) first stress peak measured at Element N6 ($t = 3.190$ μs), (c) last stress peak measured at Element N2 ($t = 31.790$ μs), (d) last stress peak measured at Element N6 ($t = 39.680$ μs).

3.7 Plasticity Effects on the Stress Wave Propagation

As discussed previously, an impact between two objects generates a stress wave. Depending on the magnitude of the pulse, the wave might induce large strains in the material. The two types of stress waves analysed in the dissertation are the elastic and plastic stress waves.

A material subjected to stresses lower than its yield strength, behaves as an elastic material. Under those circumstances, after an impact, an elastic stress wave forms. However, if post impact, the generated stress exceed the yield strength of the material, two stress waves, elastic and plastic, start to propagate. These waves are initiated at the same time but propagate at different speeds. The combination of the two stress waves induces a non-linear material behaviour, as the stress-strain ratio of the material changes. This section analyses a group of scenarios and verifies the consequences of plastic strain in the material's response to stress pulses.

3.7.1 Pulse Duration

In order to understand if the material has any influence on the duration of contact between the projectile's bottom surface and the plate's front surface, which corresponds to the impact pulse duration, a test was conducted using three different approaches to modelling steel, which have the following common properties:

- Density = 7850 kg/m^3 ;
- Young's Modulus = 200 GPa ;
- Poisson's Ratio = 0.29 .

Essentially, the elasticity model is the same for the three steels, what differs is the plasticity model. For this analysis, the three developed models consist in perfectly elastic and perfectly plastic steel as well as a steel with work (or strain) hardening. The stress-strain curves for the perfectly plastic and work hardened steels are shown in Figure 3.23.

The assembly model used for the test consists in the same configuration used earlier in the validation, a square plate of length 300 mm and thickness 30 mm, which is impacted by a projectile with the dimensions from Figure 3.1. The impact pulse is measured in the elements highlighted in Figure 3.19.

The analysis is performed using three different velocities for the projectile on each material model and the pulse duration, which is the time between the start and end of the pulse as seen in the example of Figure 3.13, is calculated upon the generation of the stress curves from the measured elements. The results are displayed in Figure 3.24 and it is possible to visualise that the duration of pulse changes exclusively for the steels with plastic regime. This means that a fraction of the kinetic energy from the projectile is converted in plastic deformation energy. This converted energy is irreversible, which causes that amidst impact, the energy that is transferred to the projectile is exclusively from elastic strain energy. Consequently, the final velocity of the projectile is inferior to its initial, extending the duration of contact with the plate. Using an impact velocity of 5 m/s, all the materials produce pulses with similar duration. The perfectly elastic steel produces a constant pulse duration across the three measured velocities while the other

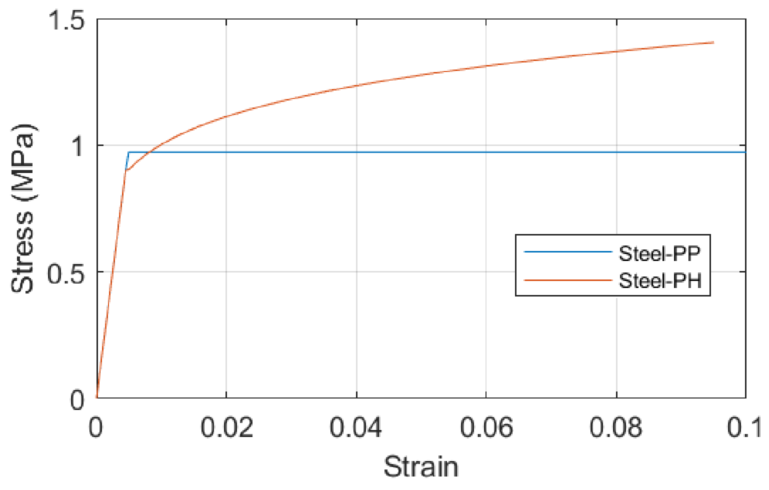
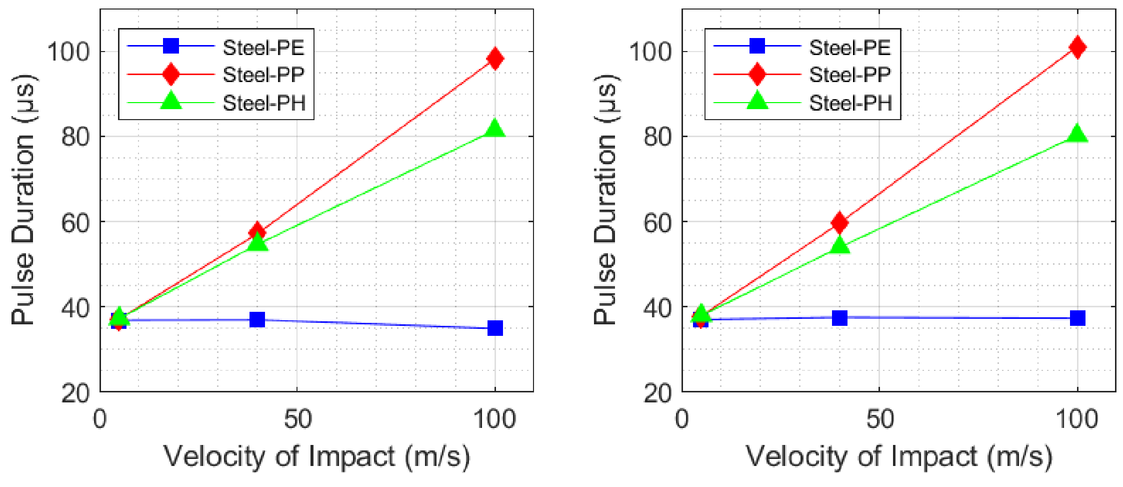


Figure 3.23: Stress-strain plot for two different modelling approaches of steel, a perfectly plastic model (Steel-PP) and a work hardened model (Steel-PH).

two materials increase the duration proportional to the impact velocity. Furthermore, the work hardened steel provides shorter pulses than the perfectly plastic. Analysing Figures 3.24a and 3.24b, both plots are very similar to each other with a slight increase on the pulse duration measured on Element N6.



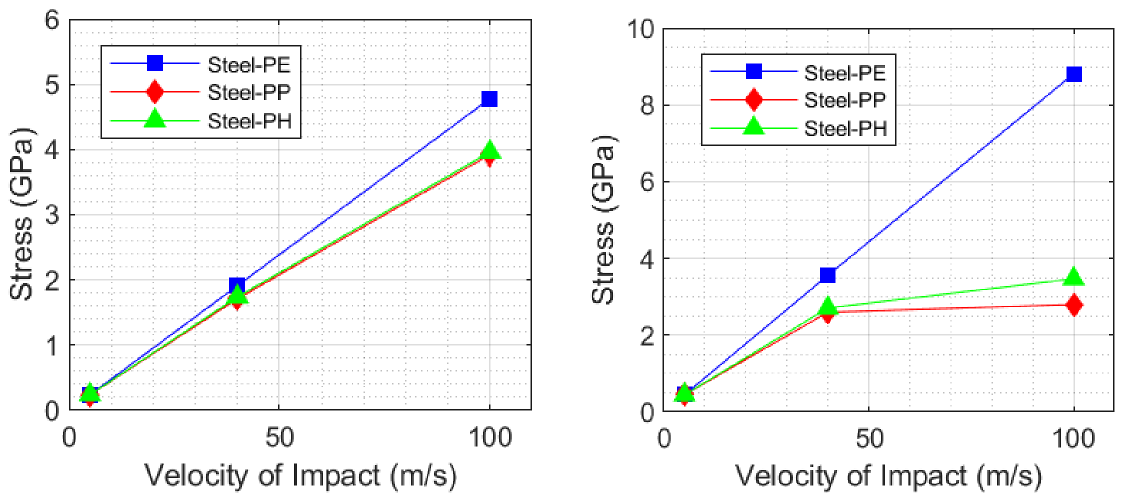
(a) Element N2 (b) Element N6

Figure 3.24: Pulse duration results.

In summary, when measuring different impact velocities, plastic strain exclusively affects the duration of the complete stress pulse. The increase in ratio is more noticeable for materials with plasticity behaviour similar to perfectly plastic. Furthermore, an elastic behaviour is preferable if the duration of the pulse is of importance.

3.7.2 Stress Magnitude

The generated impact pulse wave is similar to a square wave as seen in Figure 3.13. The following tests aim to demonstrate how and why the magnitude of the generated stress wave fluctuates between the two vertical aligned elements. For this test, the selected elements remain from the previous analysis, as well as the same set of steel modelling approaches. The evolution of the magnitude of both first and second stress peaks throughout the pulse for the three different types of steel models is shown in Figure 3.25.



(a) First stress peak (b) Second stress peak

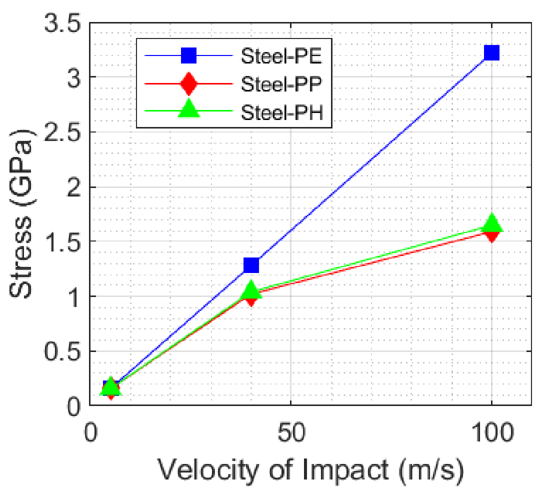
Figure 3.25: Stress magnitude measured in Element N2.

The three curves in Figure 3.25a show an almost linear relation across the three velocities of impact. However, the curves in Figure 3.25b show that the stress for the material models with plasticity does not increase linearly with the impact velocity. Furthermore, the measured stress in the elasto-plastic material models is identical for the first stress peak. However, for the second stress peak, the 100 m/s impact velocity generates higher stress for the material model with strain hardening.

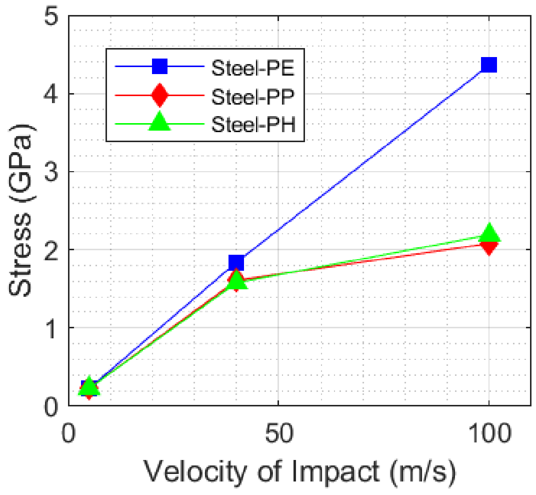
To analyse the effects of the propagation of the elastic and plastic stress waves in the plate, Figures 3.26a and 3.26b display the results for the stresses measured in Element N6.

The results are similar to the ones presented in Figure 3.25b, as the generated stress in Element N6 for the first and second stress peaks do not increase linearly. To visualise the stress dampening in between Elements N2 and N6, retrieving the data from Figures 3.25a to 3.26b it was possible to demonstrate the dampening effect of the three steels on the stress of the first and second peaks, shown in Figures 3.27a and 3.27b.

The dampening effect is more noticeable on the first stress peak where the perfectly plastic and strain hardened steels contribute with an increased dampening efficiency for higher velocity impacts. From Figures 3.27a and 3.27b it is possible to see that the measured stress for the perfectly elastic material model decreases linearly along the plate's thickness, independently from the magnitude of the stress. The material models with plasticity are affected by the amount of generated stress in the plate, as it is possible

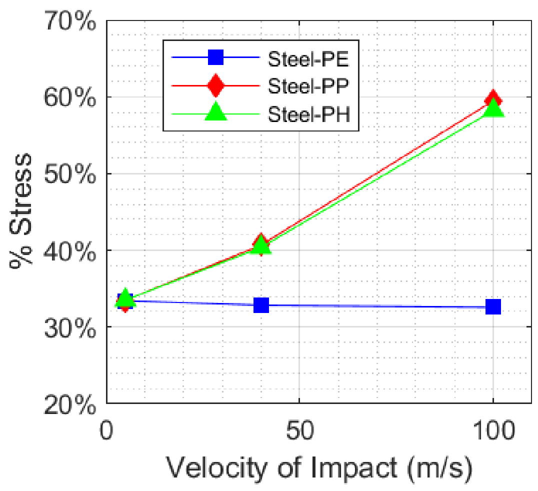


(a) First stress peak

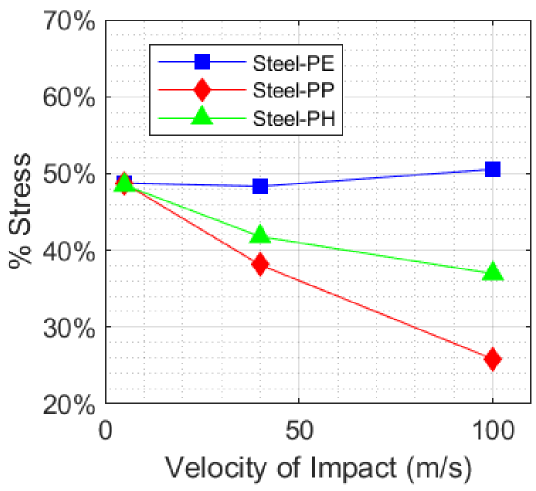


(b) Second stress peak

Figure 3.26: Stress magnitude measured in Element N6.



(a) First stress peak



(b) Second stress peak

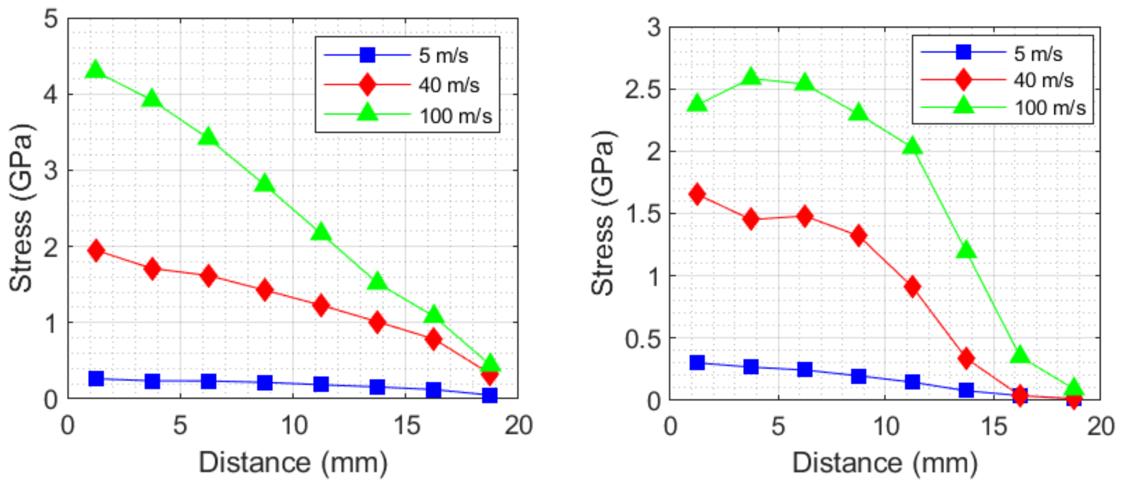
Figure 3.27: Percentage of stress dampened between Elements N2 and N6.

to see that the percentage of dampened stress along the 10 mm distance between the two elements varies with impact velocity. Furthermore, the percentage of dampened stress increases with the amount of generated stress for the first stress peak while for the second, it decreases.

In this section the magnitude of the stress wave was evaluated. The generated stresses increase proportionally to the impact velocity for a perfect elastic material model. For higher stress magnitudes, the increase in the generated stress with impact velocity is lower, for the perfectly plastic material model. The work hardened steel model behaves similarly with the perfectly plastic, although the generated stresses are slightly higher. Finally, the difference between the amount of generated stress in the perfectly elastic steel and the elasto-plastic models is noticeable for the impact velocity of 100 m/s, highlighting the importance of the absorption of energy under the form of plastic deformation.

3.7.3 Wave Progression

In this section, the wave progression throughout the thickness of a plate is analysed, featuring steel, aluminium and titanium as materials which mechanical properties are listed in Table 3.5. The plate is square with a width of 300 mm and has 20 mm of thickness. The stress wave propagates through eight C3D8 finite elements, each with 2.5 mm of thickness, and data is retrieved through the analysis of its eight integration points, returning an average stress value to be plotted in respect to the geometrical coordinate of the measured finite element's centroid. To better understand the position of said finite elements, the chosen elements for this analysis are the eight elements in the centre along the thickness of the plate. The tests are ran for the same three projectile velocities studied earlier in this chapter. To quantify the amount of stress generated throughout the plate it is important to split the stress wave into three stages, which are explained in Section 3.6.2 and illustrated in Figure 3.13. Running the tests on steel, the graphs in Figure 3.28 describe the stress progression along the thickness of the material for the first and second stress peaks.



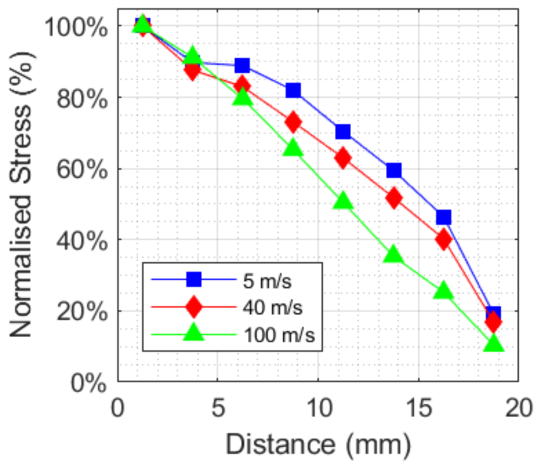
(a) First stress peak (b) Second stress peak

Figure 3.28: Stress progression along the thickness for the steel plate.

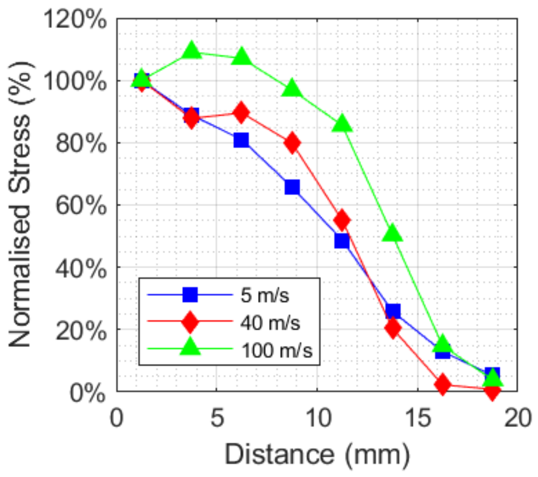
The first stress peak assumes a linear relation along the plate's thickness while the second one progresses differently depending on the velocity of the projectile. Figure 3.28 shows three different curves for each impact velocity used for testing, thus generating different levels of stress that are complicated to compare. In order to overcome that issue, the stresses were computed into normalised stress values, in respect to the first element, resulting in the plots in Figure 3.29.

With a uniform stress scale it is easier to evaluate and compare the stress wave along the thickness of the plate. It is possible to see that the for the first stress peak along the thickness of the plate, the slopes of the curves are approximately constant, being more noticeable for the 100 m/s impact.

For the second stress peak, the curves' slopes are not constant. For impact velocities higher than 5 m/s, the magnitude of the stress does not decrease considerably between 0 and 9 mm. This is due to plastic strain that is formed in the material. It can be seen that after ~ 9 mm, the magnitude of the stress harshly falls and stabilises around lower



(a) First stress peak

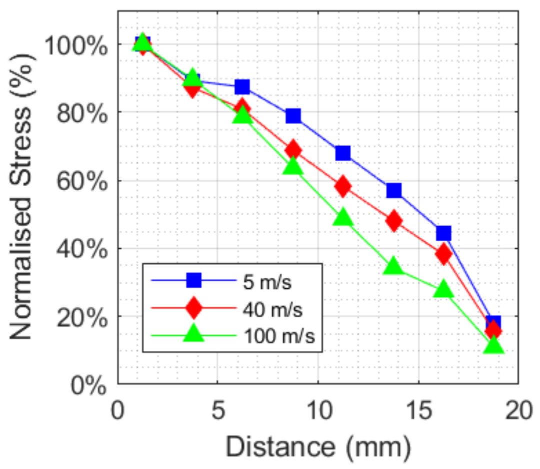


(b) Second stress peak

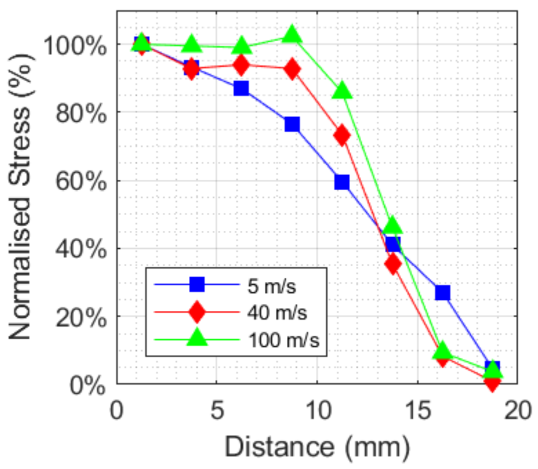
Figure 3.29: Normalised stress progression along the thickness for the steel plate.

levels between 15 and 20 mm.

In Figure 3.30 it is shown the normalised first and second stress peak plots for aluminium.



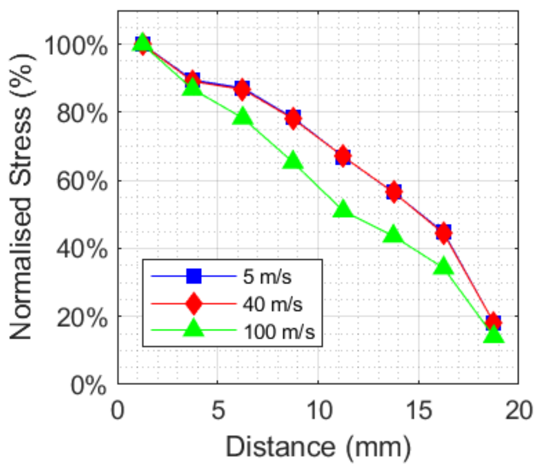
(a) First stress peak



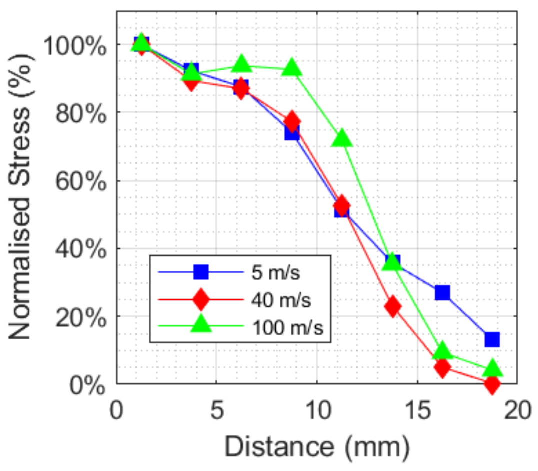
(b) Second stress peak

Figure 3.30: Normalised stress progression along the thickness for the aluminium plate.

The first stress peak progression analysis is fairly similar to steel while differences are noticeable for the second stress peak. For higher impact velocities, in the lower half of the plate's thickness, the stress levels are very identical. Taking a look at the progression of stress levels for steel and aluminium, it is already possible to see that the progression in metals is similar, resembling a pattern of evolution along the plate's thickness. For titanium, in Figure 3.31a it is possible to see that the analysed first stress peak along the plate for impact velocities of 5 and 40 m/s are akin. Furthermore, in Figure 3.31b the curves resemble a middle ground between the equivalent curves for steel and aluminium, highlighting a correlation between the Young's modulus of the studied materials, as the titanium's is close to the average value of the other materials'.



(a) First stress peak



(b) Second stress peak

Figure 3.31: Normalised stress progression along the thickness for the titanium plate.

To directly compare the generated stress along the plate's thickness between each material, the magnitude of the first and second stress peaks for the three selected impact velocities is shown in Appendix A.1.

The results for the first stress peak for each chosen impact velocity indicate that the stress magnitude decreases along the plate's thickness identically, regardless of the impact velocity. It is noticeable that the generated stress is higher for the steel plate, due to its higher Young's modulus. However, approaching the rear surface of the plate, the stress magnitudes for the three materials are contiguous. The second stress peak maintains the same stress level hierarchy when comparing with the first peak, with steel generating more stress then titanium, and aluminium generating the least of all the three metals.

Chapter 4

Setup for the Optimisation Process

Any problem in which certain parameters need to be determined to satisfy constraints can be formulated as an optimisation problem [Arora 2016]. Optimisation methods have a wide range of applicability in diverse fields. Throughout this chapter, necessary optimisation methodologies are explained as they play an important role in the optimisation problems' formulation and procedure.

4.1 Optimisation Fundamentals

4.1.1 Mathematical Formulation

The formal mathematical formulation of an optimisation problem can be expressed as [Andrade-Campos *et al.* 2015]

$$\begin{aligned} &\text{minimise} && f(x)\ , \\ &\text{subjected to} && g_{\mathrm{j}}(x) \leq 0,\ j\ =\ 1,2,...,m\ , \\ &&& h_{\mathrm{k}}(x) = 0,\ k\ =\ 1,2,...,l\ , \\ &&& x_{\mathrm{i}}^{\min}\ \leq\ x_{\mathrm{i}}\ \leq\ x_{\mathrm{i}}^{\max},\ i\ =\ 1,2,...,n\ , \end{aligned} \tag{4.1}$$

where $f(x)$ is the objective function to be minimised (or maximised, depending on the problem), $\boldsymbol{g(x)}$ are m inequality constraints, $\boldsymbol{h(x)}$ are l equality constraints and $\boldsymbol{x}$ are n variables within the established range.

4.1.2 Penalty Function Method

Optimisation procedures require the use of numerical methods to aid in the solving process. Methods that solve a constrained optimisation problem by transforming it into one or more unconstrained problems are called transformation methods [Arora 2016]. Having a case where the purpose is to minimise $f(x)$ while $h(x) = 0$ and $g(x) \leq 0$, the transformation function defined in Equation 4.2 converts the constrained objective function $f(x)$ into an unconstrained one. The function is [Arora 2016]

$$\Phi(x,r) = f(x) + P\big(h(x), g(x), r\big)\ , \tag{4.2}$$

where r is a vector of penalty parameters that controls the penalty action over a real-value function P.

The Penalty Function Method defines function P in Equation 4.2. If the constraints of the problem are violated, the objective function $f(x)$ is penalised by the addition of a positive value. While there are a number of penalty functions that can be defined, the one used in this dissertation is the *quadratic loss function*, defined as [Arora 2016]

$$P\big(h(x), g(x), r\big) = r\left\{ \sum_{i=1}^{p} \big[h_i(x)\big]^2 + \sum_{i=1}^{m} \big[g_i^+(x)\big]^2 \right\}; \quad g_i^+(x) = \max\big(0, g_i(x)\big) \ , \tag{4.3}$$

where $r > 0$ is a scalar penalty parameter. When the $g(x)$ and $h(x)$ constraints are violated, Equation 4.3 returns a positive value for the function P, penalising the cost function $f(x)$.

4.2 Optimisation Algorithms

There are numerous optimisation algorithms and some of them draw inspiration from different sources. Nature-inspired algorithms are commonly used such as Genetic Algorithms (GA), Differential Evolution Algorithm (DEA), Ant-Colony Optimisation (ACO) algorithm and Particle Swarm Optimisation (PSO) algorithm [Arora 2016].

The optimisation process of a multi-layered armour plate involves the evaluation of multiple possible configurations, whereas the combination of different materials and design variables such as order of each plate's material and assigned thicknesses. The amount of total possible configurations results in a wide array of possible solutions, although many of the solutions might not identify with the optimal. Under these circumstances, it is justifiable to resort to stochastic optimisation, which is the general class of algorithms and techniques which employ some degree of randomness to find optimal (or as optimal as possible) solutions to hard problems [Luke 2009]. Genetic algorithms and the particle swarm optimisation algorithm are population methods that involve optimisation using a collection of design points, called individuals [Kochenderfer and Wheeler 2019]. Having a large number of individuals distributed throughout the design space can help the algorithm avoid becoming stuck in a local minimum, which is advantageous for a procedure with a diverse amount of possible solutions. Furthermore, information at different points in the design space can be shared between individuals to globally optimise the objective function.

[Reis 2019] studied the influence of three distinct optimisation algorithms on stress mitigation upon ballistic impact in a three-layered armour system: Genetic Algorithm (GA), Particle Swarm Optimisation (PSO) and Simulated Annealing (SA). Testing all the mentioned optimisation algorithms, it was possible to evaluate their efficiency, precision and accuracy when applied to four different benchmarks, each under the same circumstances and objectives. Among the four benchmarks performed, the Particle Swarm Optimisation (PSO) algorithm proved its efficiency consistently, finding favourable solutions in the least number of evaluations across five distinct runs. Proving to be the most efficient algorithm and due to its ease of application, PSO was chosen as the algorithm to use for the optimisation studies ahead.

4.2.1 Particle Swarm Optimisation (PSO) Algorithm

This algorithm imitates bird flocks' fishing methods and it belongs to the class of *meta-heuristics* and *swarm intelligence* methods [Arora 2016]. The algorithm starts by producing a random set of solutions, called the initial population. Each solution from the initial population is called a *particle*. Each particle has its own current *particle position* that is updated as the algorithm proceeds with the cycle. The parameter that defines how the position of each particle evolves is the *particle velocity*. The step-by-step procedure is the following [Arora 2016]:

- **Step 0: Initialisation**. Select N_p, c_1, c_2 and t_max, assigning the maximum number of iterations to run. c_1 and c_2 are the cognitive and social parameters, respectively, and are normally set to 2. Define the starting velocity of each particle $v^{(i,0)}$ as zero. Set iteration counter to $t = 1$.
- **Step 1: Initial Population**. Generate N_p particles $x^{i,0}$ within defined boundaries using random procedures. Calculate $f\big(x^{(i,0)}\big)$ of each particle and evaluate their fitness.
- **Step 2: Update Personal and Global Best Location**. Pick each particle's best position as its $P_\text{best}^{(i,t)}$ and the global best position G_best^t. The personal best and global best are selected under the conditions of the following equations:

$$\begin{aligned} &\text{If } f\Big(x^{(i,t+1)}\Big) \le f\Big(P_\text{best}^{(i,t)}\Big), \quad \text{then} \quad P_\text{best}^{(i,t+1)} = x^{(i,t+1)} \ , \\ &\text{otherwise} \quad P_\text{best}^{(i,t+1)} = P_\text{best}^{(i,t)} \quad \text{for each} \quad i = 1 \text{ to } N_\text{p} \ . \end{aligned} \tag{4.4}$$

$$\text{If } f\Big(P_\text{best}^{(i,t+1)}\Big) \le f\big(G_\text{best}^t\big), \quad \text{then} \quad G_\text{best}^{t+1} = P_\text{best}^{(i,t+1)}, \quad i = 1 \text{ to } N_\text{p} \ . \tag{4.5}$$

- **Step 3: Calculate Velocities and Update each Particle's Position**. Each particle updates their position according to Equations 4.6 and 4.7. The parameter ω is the inertia weight and r_1 and r_2 are random numbers between 0 and 1.

$$v^{(i,t+1)} = \omega v^{(i,t)} + c_1 r_1 \left(P_\text{best}^{(i,t)} - x^{(i,t)}\right) + c_2 r_2 \left(G_\text{best}^{t} - x^{(i,t)}\right); \quad i = 1 \text{ to } N_\text{p} \ , \tag{4.6}$$

$$x^{(i,t+1)} = x^{(i,t)} + v^{(t,t+1)}; \quad i = 1 \text{ to } N_\text{p} \ . \tag{4.7}$$

- **Step 4: Check Stopping Criteria**. If the stopping criteria is not satisfied, the next iteration starts, jumping to Step 2. In case the stopping criteria is satisfied, the algorithm provides an acceptable result, returning the final G_best as the solution and stopping further iterations.

The algorithm is also represented in the form of a flowchart in Appendix A.2.

4.3 Optimisation Process through Scripting

In Chapter 3, it was explained how scripting would facilitate the process of the whole model generation. The script is the main tool used in this dissertation and provides all the means to an easier and faster interaction with Abaqus. This section demonstrates how the script not only has the purpose of aiding with the many possible configurations of model generations, but also as a powerful tool to control the desired optimisation routines, accessing different functions to acquire and process data. The script is written in Python and is used to implement the optimisation procedure automatically using four essential functions:

- **Python Optimisation Routine**: This is the main function of the script, containing the definition of the Particle Swarm Optimisation algorithm, which controls the other three functions;
- **Model Generator**: A parameterized function that generates the model along with each simulation calculation. The available parameters for the model generation are detailed in Appendix A.3;
- **Data Acquisition**: Whenever the calculations for the previous function are completed, this function processes and saves the required results (e.g. maximum stress on the rear plate);
- **Objective Function**: Contains the objective functions accessed by the algorithm and relies on parameters of the Model Generator function to return the value (e.g weight calculation).

To better understand how the four functions operate, the flowchart in Figure 4.1 illustrates the cycle of the overall optimisation process. Each evaluation of the algorithm requires a simulation of the impact test in Abaqus, hence powerful hardware is required to minimise computational times. A workstation provided by the Department of Mechanical Engineering of the University of Aveiro is used to solve the optimisation problems. The workstation has two Intel® Xeon® E5-2690 v4 @2.60 GHz processors with 35 MB cache and 14 cores each, 256 GB of RAM and an NVIDIA® Quadro® P4000 GPU with 8 GB of dedicated memory. Using the same settings of [Reis 2019], the simulation procedures are previously programmed to take advantage of Abaqus parallelization and use 8 processors to run, which results in lower calculation times.

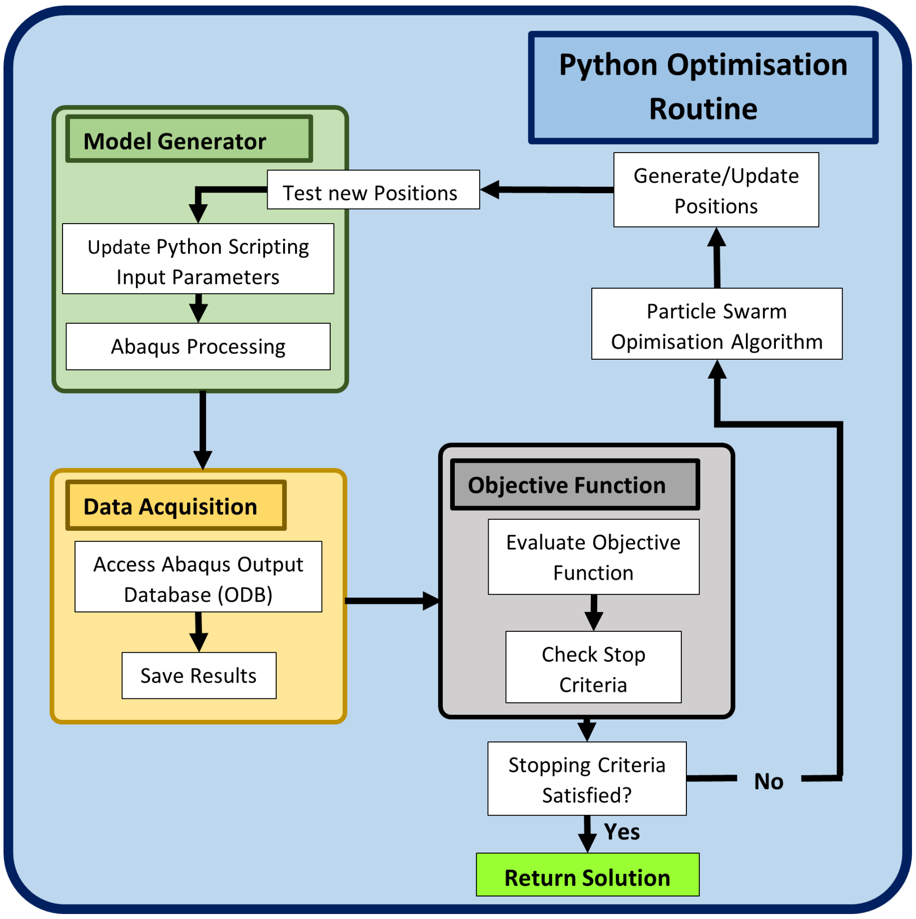

Python Optimisation Routine
Model Generator
Test new Positions
Generate/Update Positions
Update Python Scripting Input Parameters
Abaqus Processing
Particle Swarm Opimisation Algorithm
Data Acquisition
Access Abaqus Output Database (ODB)
Save Results
Objective Function
Evaluate Objective Function
Check Stop Criteria
Stopping Criteria Satisfied?
No
Yes
Return Solution

Figure 4.1: Flowchart of the optimisation process.

4.4 Benchmark — Multi-layer Armour Plate Impacted by Projectile with a Velocity of 5 m/s

This section serves as a preliminary test of the Particle Swarm Optimisation algorithm. Using as reference the first benchmark conducted by [Reis 2019], a replication attempt is made trying different parameters for this algorithm. A diagram of the model used for this benchmark is depicted in Figure 4.2. Serving as a starting point and with the purpose of comparing results, the model in Abaqus uses the following configuration:

- **Plate Geometry**: Square plates of 140 mm of width;
- **Number of plates**: 3 (A, B and C, A being the front plate);
- **Plate Boundary Condition**: Rear surface of the rear plate is clamped;
- **Projectile Velocity**: 5 m/s.

The material properties for this model are listed in Table 4.1.

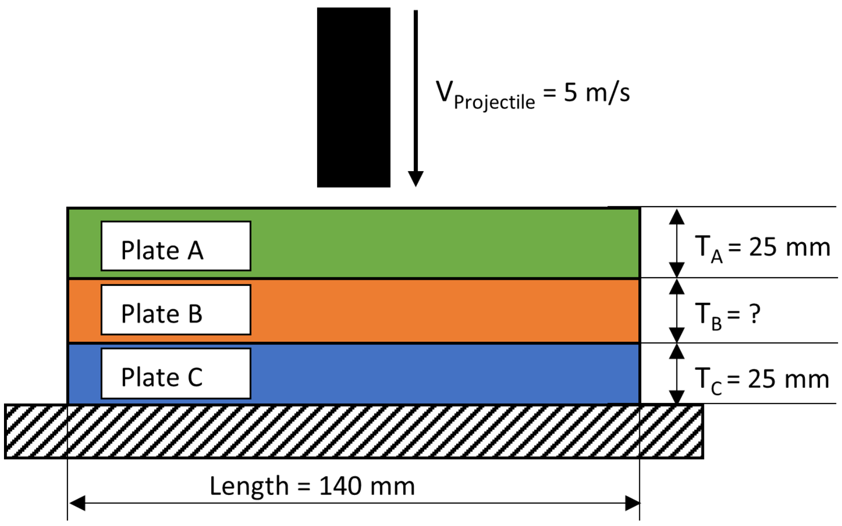


Figure 4.2: Diagram for the Benchmark.

Table 4.1: Plate materials and corresponding mechanical properties.

Plate	Material	Density (kg/m^3)	Young's Modulus (GPa)	Yield Strength (MPa)	Poisson's Ratio
A	Aluminium	2700	70	276	0.33
B	Nylon-6	1140	3	82	0.35
C	Steel	7850	200	350	0.25

4.4.1 Problem Formulation

The objective is to minimise the weight of the system composed by the three plates by finding the optimal thickness for the middle plate (Plate B) while respecting the condition that the maximum stress ($\sigma_{\mathrm{z,\,max}}$) on the rear impact region of Plate C (check Figure 3.2) is bellow the allowed value of 20 MPa.

Mathematically, the problem can be stated as:

$$\begin{aligned} &\text{minimise} && f(t_{\mathrm{B}}, r) = W(t_{\mathrm{B}}) + P(t_{\mathrm{B}}, r)\ , \\ &\text{subjected to} && t_{\mathrm{B,\,min}} \leq t_{\mathrm{B}} \leq t_{\mathrm{B,\,max}}\ , \\ & && g(x) = \sigma_{\mathrm{z}} - \sigma_{\mathrm{z,\,max}} \leq 0 \end{aligned} \tag{4.8}$$

where

$$\begin{aligned} W(t_{\mathrm{B}}) &= 4L_{1,\mathrm{A}}L_{2,\mathrm{A}}\big(\rho_{\mathrm{A}}t_{\mathrm{A}} + \rho_{\mathrm{B}}t_{\mathrm{B}} + \rho_{\mathrm{C}}t_{\mathrm{C}}\big) \\ &= 4 \times 0.07^2 \times \big(67.5 + 1140t_{\mathrm{B}} + 196.25\big), \end{aligned} \tag{4.9}$$

where the objective function $f(t_{\mathrm{B}}, r)$ is the sum of the weight of the system $W(t_{\mathrm{B}})$ and the penalty function $P(t_{\mathrm{B}}, r)$. The weight is calculated relative to Plate A's dimensions, L_1 (length) and L_2 (width), and the density of Plates A, B and C ($\rho_{\mathrm{A,B,C}}$) as well as the thickness of each plate ($t_{\mathrm{A,B,C}}$). It is relevant to clarify that since only a quarter of the plate is modelled, the total weight is four times greater than the numerical model processed in Abaqus and considering that the plates are square, L_1 (length) and L_2 (width) are equal.

This benchmark evaluates the evolution of the total weight of the system by finding the optimal value for the thickness of Plate B. Setting a low projectile velocity of 5 m/s, this test focuses on a linear problem of elastic stress wave propagation.

As discussed in Chapter 3, the model consists of square plates of width 300 mm, thus two tests are made: one with the exact same sizes used in the model of [Reis 2019] and the other featuring the proposed size. Testing the two different plate sizes not only serves as an opportunity to analyse the influence of different PSO operating parameters but also as an opportunity to check how the plate size affects the overall computational times. Moreover, these tests setup the optimisation procedures for Chapter 5.

4.4.2 Setup and Implementation

The first setup for this benchmark is the one used by [Reis 2019] in the tests conducted for Benchmark I. The purpose is to compare the performance and effectiveness of the PSO algorithm at finding the optimal solution. This benchmark is divided into three experiments: Experiment α, Experiment β and Experiment Ω, in order to compare the performance of the combination between different parameters. This strategy provides a method to evaluate the impact of the cognitive and social parameters on the performance and accuracy of the algorithm. The performance of the algorithm throughout the three experiments are taken into account when setting up the higher complexity problems in the following chapter. The parameters used for Experiment α are listed in Table 4.2.

Table 4.2: PSO Operational parameters for Experiment α.

Number of particles	Number of iterations	ω	c_1	c_2	r
15	30	0.5	1	2	2

Benchmark – Experiment α

Due to the stochastic nature of the PSO algorithm [Huang *et al.* 2012], it is necessary to run multiple tests. Taking into account the computational weight limitations of the optimisation procedure, as each evaluation requires a complete simulation in Abaqus, each set of parameters is processed three times. Since this is an optimisation problem of one variable, three runs are enough to return consistent results. The evolution of the best, worst and average values of the objective function at each iteration is shown in Figure 4.3. The effects of the random generated positions of the particles attributed to the initial individuals is noticeable, as shown by the discrepancy between the three curves, highlighting the stochastic nature of the algorithm.

The results from the optimisation procedures are computed using the average solution at each iteration, in order to be possible to compare the results from each run. In Figure 4.4a it is possible to see that the CPU time for Test 1 is above the other two. Test 3 was the fastest to converge to the optimal weight solution after 165 evaluations. The average

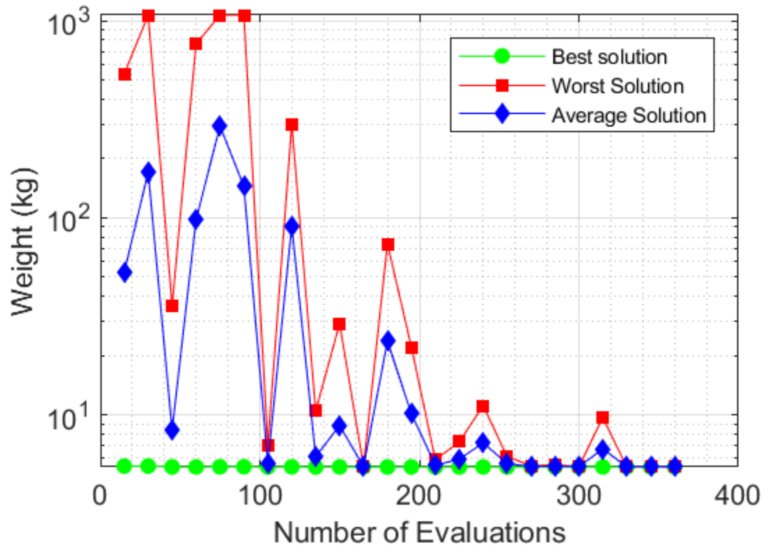


Figure 4.3: The evolution of the best, worst and average values of the objective function returned in Experiment α.

of the returned stress by all particles in each iteration is represented in Figure 4.4b. As the stress constraint was set to 20 MPa, the results show that while minimising the weight, the measured stress in the back face of Plate C is equal to the defined constraint. This means that the solutions found result in an armour plate that when impacted by a projectile with a velocity of 5 m/s generate approximately 20 MPa of stress in the rear surface of the back-plate, rendering them as local optimums. It is possible to see from Appendix A.4 that the evolution of the assigned thickness is identical to Figure 4.4a, since the weight function is defined by Equation 4.9 and is a function of a single variable, the assigned thickness of Plate B (t_B).

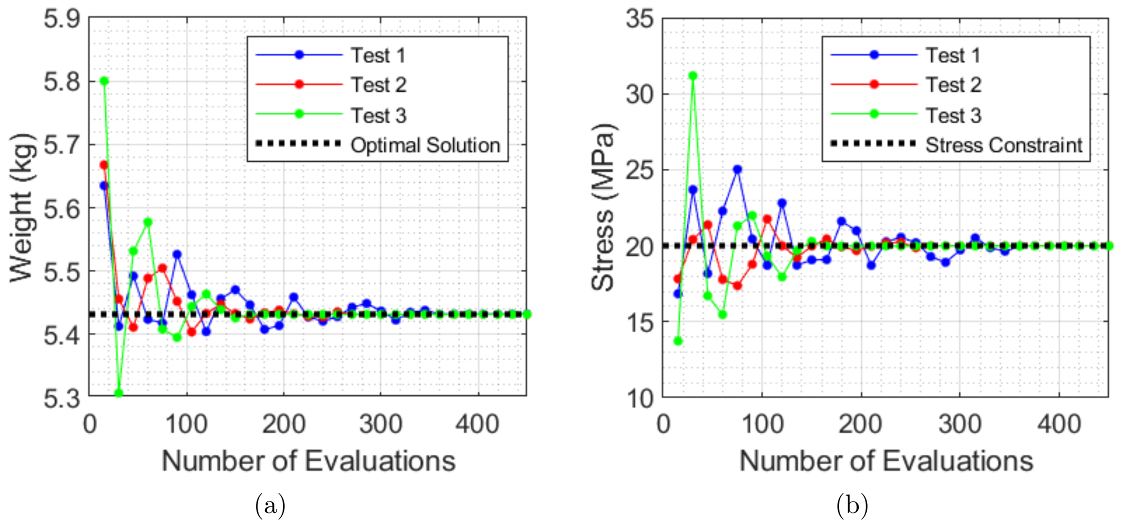


Figure 4.4: The evolution of: (a) the weight function, (b) maximum stress returned for Experiment α.

Benchmark – Experiment β

Changing the value of the cognitive parameter (c_1), running three more tests it is possible to evaluate the influence of this particular parameter in the accuracy of the algorithm and in the overall computational time. The parameters used in Experiment β are listed in Table 4.3.

Table 4.3: PSO Operational parameters for Experiment β.

Number of particles	Number of iterations	ω	c_1	c_2	r
15	30	0.5	2	2	2

In Figures 4.4a and 4.5a it can be observed that convergence is achieved before the final iteration for both experiments. However, using the parameters for Experiment α resulted in faster convergence times. Since the test is processed for low velocity impacts, it is safe to assume that under the conditions established, the natural weight configuration would be the one with the thickness of Plate B for which the maximum stress is equal to the defined one, which is 20 MPa. In the graphs of Figures 4.4b and 4.5b it is possible to observe that it is indeed the maximum stress for the optimal value of thickness.

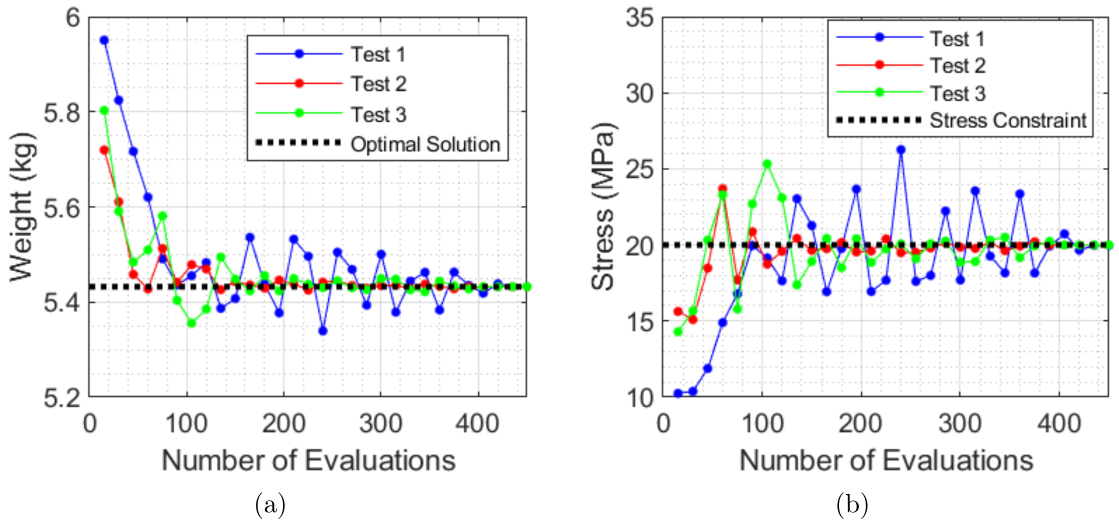


Figure 4.5: The evolution of: (a) the weight function, (b) maximum stress returned for Experiment β.

To discuss how the cognitive parameter (c_1) affects the performance of the algorithm, two bar charts are created. The first one, in Figure 4.6, displays the number of evaluations needed until the algorithm finds the solution while the second one, in Figure 4.7, evaluates the precision of the algorithm throughout the three experiments with standard deviation calculations. The charts contain data from the first benchmark conducted by [Reis 2019] and the two experiments conducted in this benchmark.

Examining the bar chart in Figure 4.6, it is possible to see that setting up the cognitive parameter as $c_1 = 1$ results in fewer evaluations. The stochastic nature of the algorithm noticeable for Experiment α, as seen by the discrepancy between the number of necessary evaluations across the three tests. In terms of the standard deviation calcu-

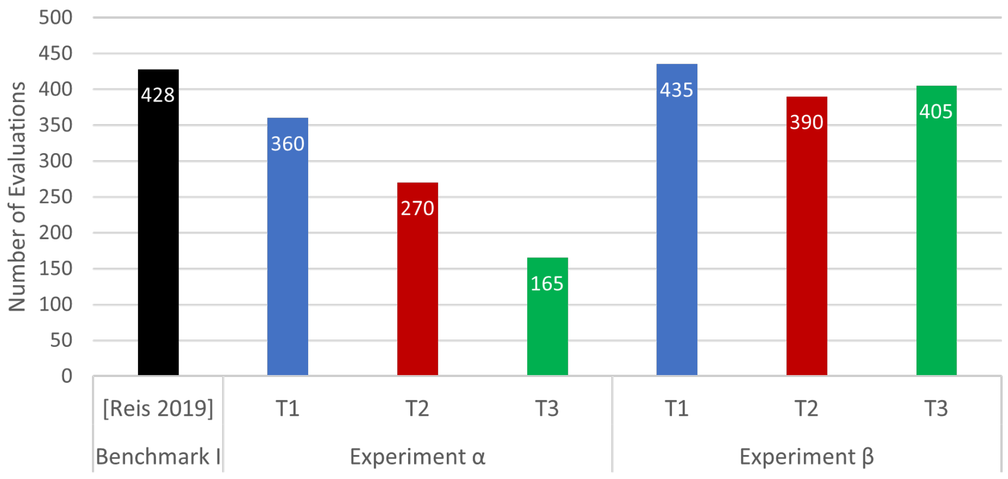


Figure 4.6: Comparison between the number of evaluations from each test.

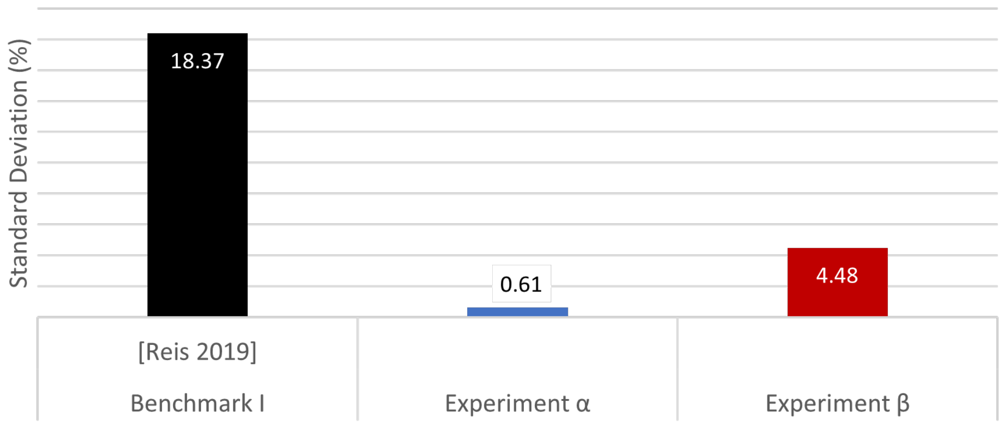


Figure 4.7: Comparison between the standard deviation from each experiment.

lations, Experiments A and B show results with lower standard deviation. The standard deviation calculated from Experiment β is higher than the one from Experiment α since, as a result of a higher cognitive parameter (c_1), the particles have the tendency to search possible solutions closer to their personal best location, while also searching towards the current global best position. The increase in the search space of solutions contributes to a higher dispersion between them, which results in an increased standard deviation.

Benchmark – Experiment Ω

Focusing on transitioning the model used by [Reis 2019], which is based on square plates of 140 mm of width and with the rear surface of the back-plate fixed, to the proposed model in Section 3.1, an Experiment Ω is created. To be able to compare the results from this experiment with the ones from previous experiments, the boundary conditions of this model maintain the same. The diagram of the experiment if shown in Figure 4.8. Adding this experiment provides an opportunity to evaluate the differences in computational times between the three of them. Furthermore, with results from this experiment it is also possible to check how the increased plate size affects the optimal solution of the problem.

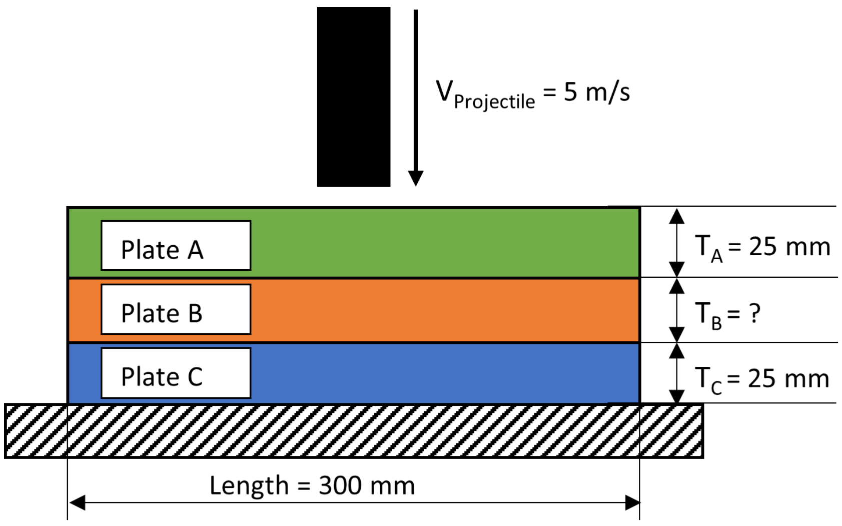


Figure 4.8: Diagram for Experiment Ω.

In this experiment, the size of the plate transitions from 140 mm to 300 mm, which as consequence slightly modifies the weight formula in Equation 4.9. The weight formula for this case can then be defined as

$$\begin{aligned} W(t_\mathrm{B}) &= 4L_{1,\mathrm{A}}L_{2,\mathrm{A}}\big(\rho_\mathrm{A}t_\mathrm{A} + \rho_\mathrm{B}t_\mathrm{B} + \rho_\mathrm{C}t_\mathrm{C}\big) \\ &= 4 \times 0.15^2 \times \big(67.5 + 1140t_\mathrm{B} + 196.25\big), \end{aligned} \tag{4.10}$$

maintaining the same set of materials used previously. The PSO operational parameters chosen for this experiment are the same as the ones from Experiment α, as it resulted in a lower number of evaluations in all three tests.

The more effective way of finding the optimal solution of this benchmark is favour the search towards the global best instead of the personal best, since the solution results in a model configuration at which the maximum stress analysed on the back-face of the rear plate is $\sim$ 20 MPa. It is then clear that a higher social parameter (c_2) and lower cognitive parameter (c_1) is more suitable for the setup parameters on this benchmark. Looking at the weight evolution in Figure 4.9 it is possible to see that, as expected, the algorithm is able to find the optimal solution in a small number of evaluations, with the fastest one converging after 225 evaluations.

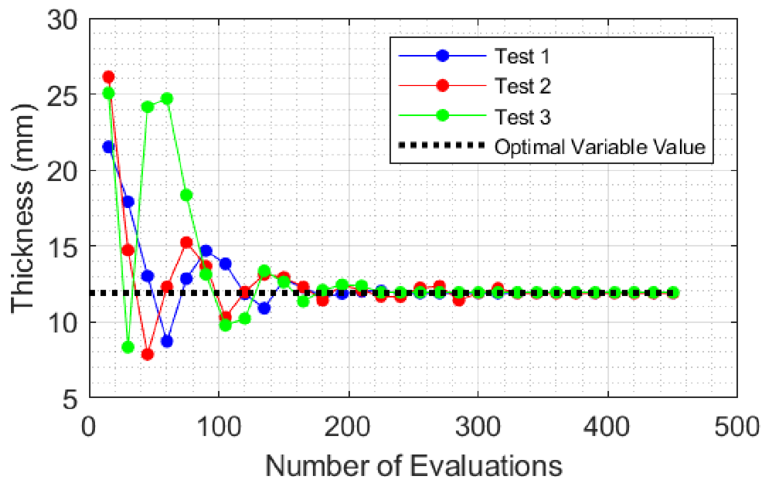


Figure 4.9: The evolution of the thickness for Experiment Ω.

4.4.3 Results and Discussion

The information regarding the number of evaluations and computational times of all tests of this benchmark are listed in Table 4.4, while the summary of the obtained results of the benchmark are listed in Table 4.5.

Table 4.4: Number of evaluations and computational times for the benchmark.

		Experiment		
		α	β	Ω
Number of evaluations	Test 1	360	435	240
	Test 2	270	390	330
	Test 3	165	405	225
Computational times (s)	Test 1	20854	22828	11715
	Test 2	17952	22880	17615
	Test 3	8819	19854	12142

Table 4.5: Results obtained for the benchmark.

	Experiment		
	α	β	Ω
Best solution (mm)	11.49943	11.50784	11.92474
Average (mm)	11.51718	11.54056	11.91241
Standard deviation	0.0061311	0.044825	0.021135
Resulting weight (kg)	5.43134	5.431538	24.98347

It is possible to see that the weight returned in Experiment Ω is higher than in the other two experiments as a result of the increased size of the plates. The standard deviation for all experiments is considerably low and every experiment is able to converge into the optimal solution. It is also interesting to verify that the increased size of the plates affects the optimal value for the thickness of Plate B. The reason for Experiment Ω returning a configuration with a higher interlayer thickness is due to the difference in

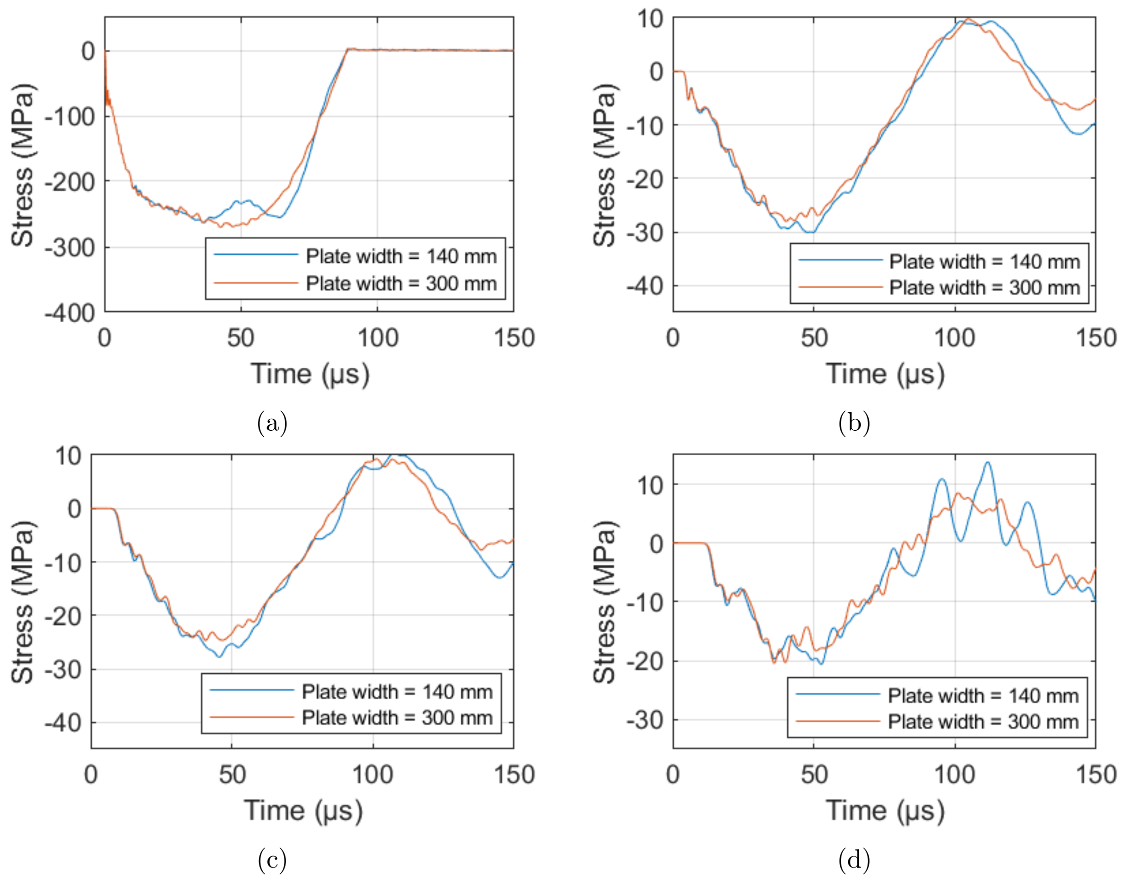


Figure 4.10: Average stress in OZ direction for the resulting configuration from Experiment α and β (plate width = 140 mm) and from Experiment Ω (plate width = 300 mm) measured at: (a) impact region in front surface of Plate A, (b) impact region in front surface of Plate B, (c) impact region in front surface of Plate C, (d) impact region in rear surface of Plate C.

the shape of the stress wave. In Figure 4.10 it is possible to see the shape of the wave for both plate sizes analysed in four different regions: impact area of the front surface of Plate A, B and C and the impact area of the rear surface of Plate C. In Figure 4.10a, it is noticeable that the curves start to separate around $\sim 40\ \mu$s, resulting in the propagation of a stress wave with different shape for the two configurations.

The shear stress in plane OXZ for the two configurations, measured in the impact area of the front surface of Plate A, is shown in Figure 4.11. The increase in plate size means that a larger area is clamped at the rear surface of Plate C. This affects the bending moment in the centre of the plate, hence a divergence between the curves that can be noticed at approximately $\sim 10\ \mu$s. As a result, the propagation of the transverse wave in plane OXZ may interfere with the propagation of the longitudinal wave in the OZ direction, thus resulting in a different wave shape for the two configurations. This leads to the stress difference analysed at the rear surface of Plate C, as shown in Figure 4.10d.

To analyse the transfer of energy from the projectile to the plates, Figure 4.12 show the evolution of the kinetic energy of the projectile and the strain energy of Plates A,

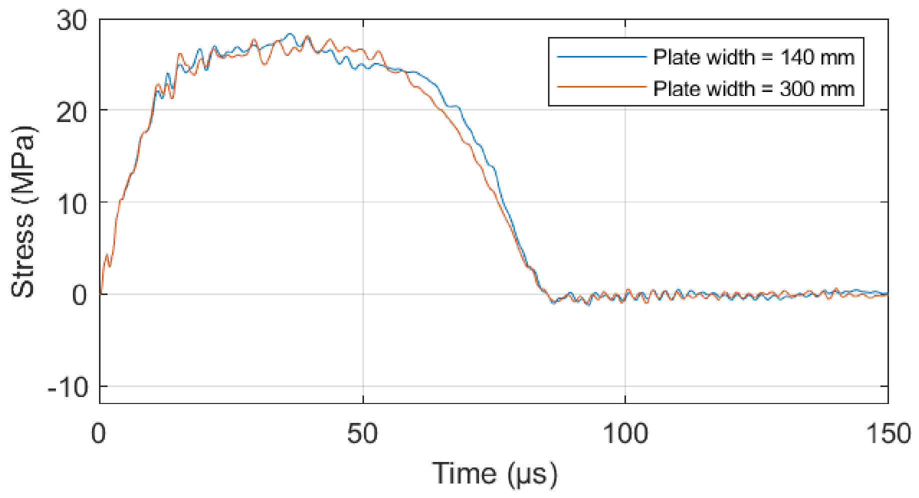
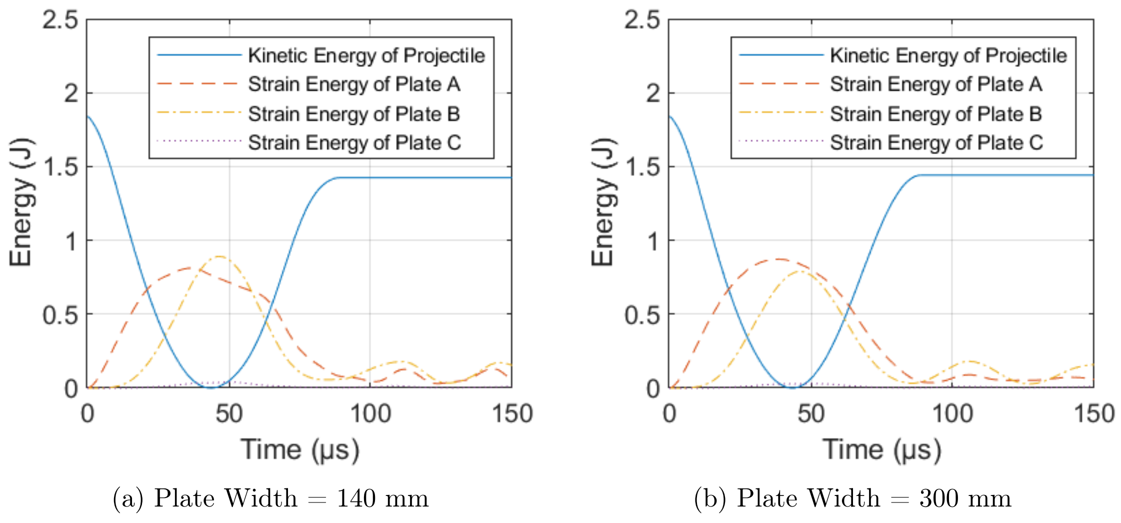


Figure 4.11: Average shear stress in OXZ plane for the resulting configuration from Experiment α and β (plate width = 140 mm) and from Experiment Ω (plate width = 300 mm) measured at the impact region in the front surface of Plate A.

B and C for the two configurations. The curves are extracted from the history output database in Abaqus. It is noticeable that the projectile's kinetic energy evolution is very similar in the two configurations. The strain energy, however, is distributed differently, as it is possible to see in Figure 4.12a that the maximum strain energy in Plate A is lower than in Plate B. In Figure 4.12b, it is seen that a higher amount of energy is transferred to Plate A, as its maximum strain energy is higher. From the graphs it can be concluded that for both configurations the amount of energy transferred into Plate C is very low, since the material is steel, resulting in low amount of longitudinal strain due to a higher Young's modulus than the other plates' materials.



(a) Plate Width = 140 mm (b) Plate Width = 300 mm

Figure 4.12: Evolution of the kinetic energy of the projectile and strain energy of the plates.

Chapter 5

Optimisation Procedure and Implementation

This dissertation targets the development of an optimisation tool that can be adapted to numerous designs of ballistic test models. The diversity in possible material configurations in layered armour protection systems needs to be considered when setting up the parameters for both the Particle Swarm Optimisation algorithm and also the time increment size and duration of event in Abaqus. Furthermore, as discussed in previous chapters, the chosen type of boundary condition on the sides of the plates resembles the types of backing material fixtures used in real experimental tests, allowing the plates to bend upon impact and resulting in different generated stress magnitudes and wave propagation.

This chapter aims to incorporate all the knowledge from the previous chapters along two distinct optimisation problems. At the end of the previous chapter, a benchmark was presented with intentions of comparing results obtained by [Reis 2019], serving as a validation problem for the two problems of the current chapter. The first problem focuses on the optimisation of the interlayer thickness of a 3-layer armour plate subjected to a projectile impact moving with a velocity of 25 m/s. The geometrical model used in this problem is detailed in Chapter 3. The optimisation process is focused on evaluating the performance effects of different operational parameters of the Particle Swarm Optimisation algorithm. The second problem consists in solving an optimisation problem involving a multi-layered plate subjected to a projectile impact with 40 m/s of velocity, although searching for the optimal thickness values of tree plates. Different parameters are tested in each Problem to evaluate the impact on computational times at the end of each Problem and how they are suitable to each type of problem. The goal is to maximise the overall performance of the PSO algorithm regardless of the type of problem in question. Furthermore, the solutions obtained are analysed in terms of generated stress and energy transfer.

5.1 Problem I — Multi-layer Armour Plate Impacted by Projectile with a Velocity of 25 m/s

This problem tests different PSO parameters for the designed model described in Chapter 3 in order to enhance the performance of the algorithm. The details of the model for this problem are shown in Figure 5.1 and can be summarised as:

- **Plate Geometry**: Square plates of 300 mm of width;
- **Number of plates**: 3 (A, B and C);
- **Plate Boundary Condition**: Side surfaces of each plate are clamped;
- **Projectile Velocity**: 25 m/s.

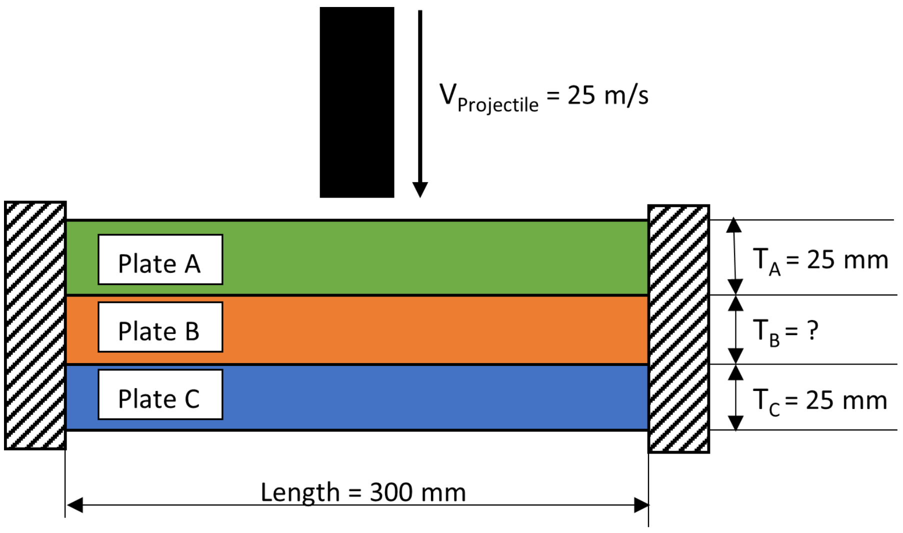


Figure 5.1: Diagram for Problem I.

With the intention of generating plastic deformation on the plate and evaluate how the calculations are affected, the chosen materials, extracted from Tables 3.1 and 3.2, are listed in the Table 5.1.

Table 5.1: Plate materials and respective mechanical properties.

Plate	Material	Density (kg/m^3)	Young's Modulus (GPa)	Yield Strength (MPa)	Poisson's Ratio
A	Aluminium 1100	2710	68.9	55	0.33
B	Nylon-6	1140	3	100	0.35
C	AISI 4340 Steel	7850	200	1355	0.29

5.1.1 Problem Formulation

This problem sets up the optimisation of a three layer armour system. The objective of this problem is to obtain the minimal interlayer thickness that enables the armour plate to withstand the stress generated by a projectile impact with a velocity of 25 m/s. The stress is analysed in the impact region of the front surface of Plate B, shown in Figure 5.2.

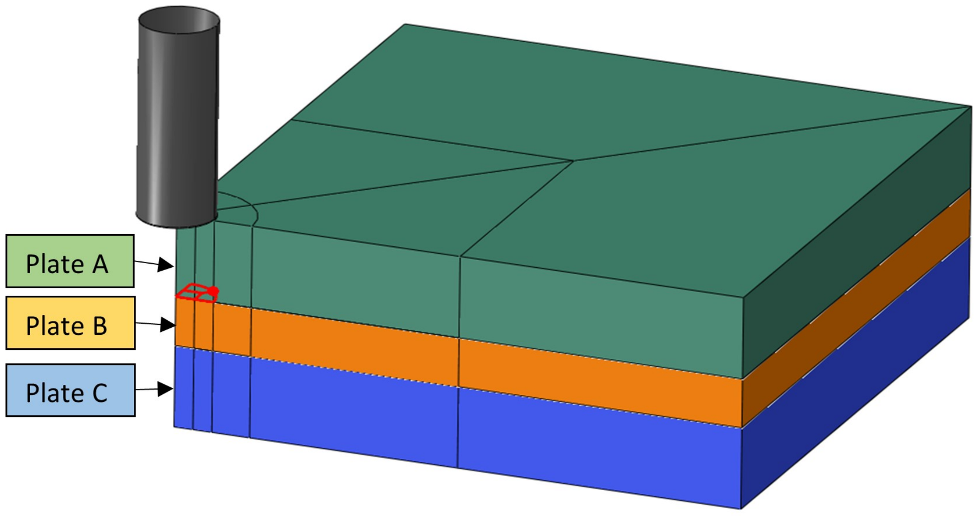


Figure 5.2: Representation of the region where the stress is measured for Problem I.

The scope of the procedure is based on generating plastic deformation exclusively on Plate A, ensuring that the Nylon-6 interlayer plate is exclusively subjected to the propagation of elastic stress waves. This enables the possibility to compare how the stress wave propagates throughout the three different plates. The results obtained regarding generated stresses on the plates and the energy transfers from the projectile to the plates are analysed in Section 5.1.3. Focusing on the weight minimisation of the armour plate, the problem can be defined as

$$\begin{aligned} &\text{minimise} && f(t_{\mathrm{B}}, r) = W(t_{\mathrm{B}}) + P(t_{\mathrm{B}}, r) \ , \\ &\text{subjected to} && t_{\mathrm{B,\,min}} \leq t_{\mathrm{B}} \leq t_{\mathrm{B,\,max}} \ , \\ & && g(x) = \sigma_{\mathrm{z}} - \sigma_{\mathrm{z,\,max}} \leq 0 \ , \end{aligned} \tag{5.1}$$

where

$$\begin{aligned} W(t_{\mathrm{B}}) &= 4 \times L_{1,\mathrm{A}} \times L_{2,\mathrm{A}} \times \left(\rho_{\mathrm{A}} t_{\mathrm{A}} + \rho_{\mathrm{B}} t_{\mathrm{B}} + \rho_{\mathrm{C}} t_{\mathrm{C}}\right) \\ &= 4 \times 0.15^2 \times \left(67.75 + 1140 t_{\mathrm{B}} + 196.25\right), \end{aligned} \tag{5.2}$$

and the maximum stress is defined as $\sigma_z = 90$ MPa. The minimum and maximum possible thickness values for Plate B are 1.0 mm and 50.0 mm, respectively. The stress-strain values for the plasticity model used are displayed in Figures 3.4 and 3.5. Setting up a 90 MPa stress constraint measured on the frontal impact region of Plate B means that the algorithm searches for solutions where the measured stress is sure to be lower than the yield strength of the material, which is 100 MPa.

It is important to state that, when trying to minimise the weight of the plate, naturally the solutions that the algorithm return result in model configurations which measured stress is approximately ~ 90 MPa, which is the defined maximum stress constraint. This happens because this optimisation problem deals with a single variable, the interlayer thickness.

5.1.2 Setup and Implementation

The benchmark in the previous chapter serves the purpose of understanding how the cognitive and social parameters of PSO affected the performance of the algorithm. For this problem, focusing on the values for the inertia weight ω and the penalty parameter r, it is possible to understand how the algorithm performance is affected. Understand how the algorithm performs when searching towards the optimal solution is important when processing more complex problems, such as Problem II in Section 5.2, which is an optimisation problem with three variables.

This problem is divided in four distinct experiments. The first experiment, Experiment α, is conducted using the same parameters as the first experiment of the benchmark, but reducing the number of particles used from 15 to 10 and increasing the penalty parameter r from 2 to 5. The number of particles is reduced to test how it affects the algorithm's performance, while an increase in the penalty parameter ensures that the algorithm is finding solutions that do not violate the stress constraint. The second and third experiments, Experiments β and ω, evaluate the inertia weight effects in the evolution of the particles. Finally, Experiment γ evaluates how reducing the penalty parameter to $r = 2$ changes the results returned from the algorithm. The parameters used in all the experiments are listed in Table 5.2.

Table 5.2: Problem I - PSO operational parameters for the experiments.

Experiment	Number of particles	Number of iterations	ω	c_1	c_2	r
α	10	30	0.5	1	2	5
β	10	30	0.75	1	2	5
Ω	10	30	0.25	1	2	5
γ	10	30	0.25	1	2	2

Problem I – Experiment α

In this experiment, the penalty parameter r was set to 5 for a smoother operating algorithm under the desired stress constraint.

Looking at the results in Figure 5.3a, it is possible to see that the algorithm evolves in a very similar way as in Experiments α and Ω from the benchmark in the previous chapter, as expected, since they share the same configuration of cognitive and social parameters. The stress plot from Figure 5.3b confirms that the particles converge close to a returned stress of 90 MPa.

With this configuration of parameters, the search space is well covered and the particles tend to search towards the global best location. It is noticeable that in Figure 5.3a the particles after thirty evaluations are close to the optimal solution and then fluctuate about it, finally starting to converge after $\sim$ 160 evaluations. The inertia weight of the algorithm affects the time necessary for the particles to converge, since the evolution of the particles' velocities are linked to it.

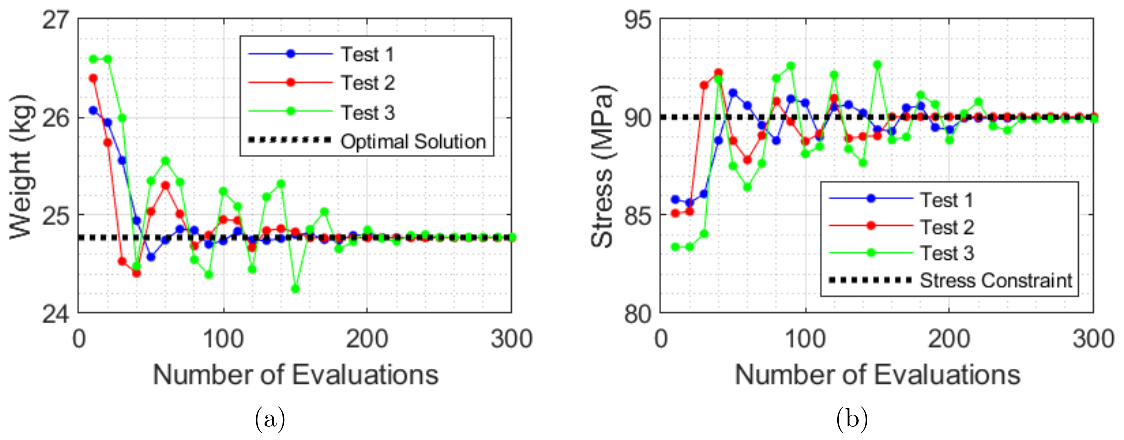


Figure 5.3: Problem I - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment α.

Problem I – Experiment β

The second experiment of Problem I tests the consequences on the algorithm progression upon a higher inertia weight (ω) value. Essentially, the higher the inertia the larger the search space of each particle. However, by covering a larger search space a slower convergence might be induced, due to a more intense shift in position between iterations. Nevertheless, the selected operational parameters for this experiment are listed in Table 5.2.

In Figure 5.3a it can be seen that the particles oscillate about the optimal solution, taking longer to converge towards it. This is the effect of the increased inertia weight, as the velocity magnitude of each particle decreases at a slower ratio when compared to the results obtained in Experiment α. Additionally, the stopping criteria for the algorithm is not satisfied for any of the three tests, resulting in it running the maximum amount of iterations. In Figure 5.4b it is plotted the average of the returned stress from the three tests along all iterations. It can also be noticed that the increased inertia weight is resulting in many solutions that do not respect the defined maximum stress constraint.

The evolution of the objective function is shown in Figure 5.5a, where it can be seen that the particles are not being guided towards the optimal solution, with Test 1 having the worst solutions. To understand how the particles are behaving in Test 1, Figure 5.5b shows the evolution of the best, average and worst solutions. It is possible to see that while the algorithm was able to return good solutions, it failed in guiding the majority of the particles towards them, as seen in the red curve.

The algorithm showed some difficulty in finding the optimal variable in Tests 1 and 2 with considerable shifts around the defined stress constraint, only starting to stabilise after ~ 200 evaluations. This means that in order for the algorithm to be able to converge into an actual solution, it needs to run additional iterations and thus this particular choice of PSO operational parameters is not ideal.

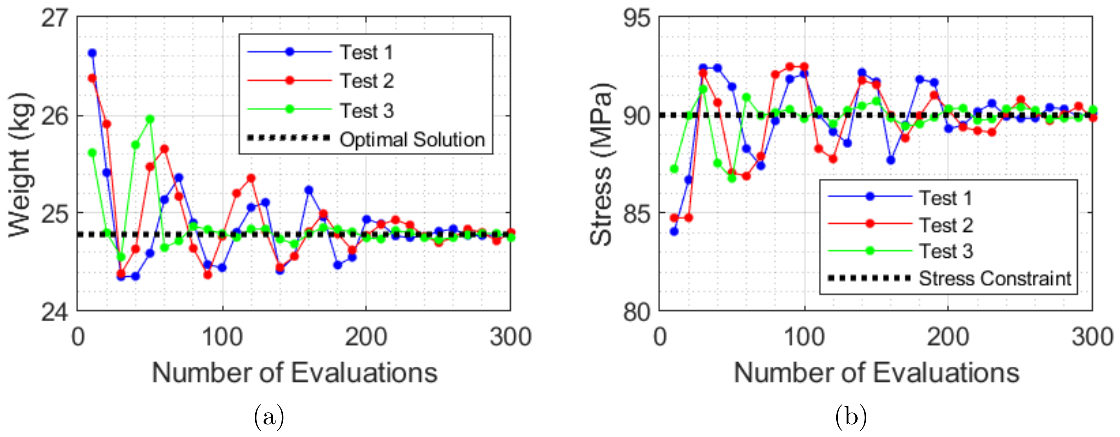


Figure 5.4: Problem I - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment β.

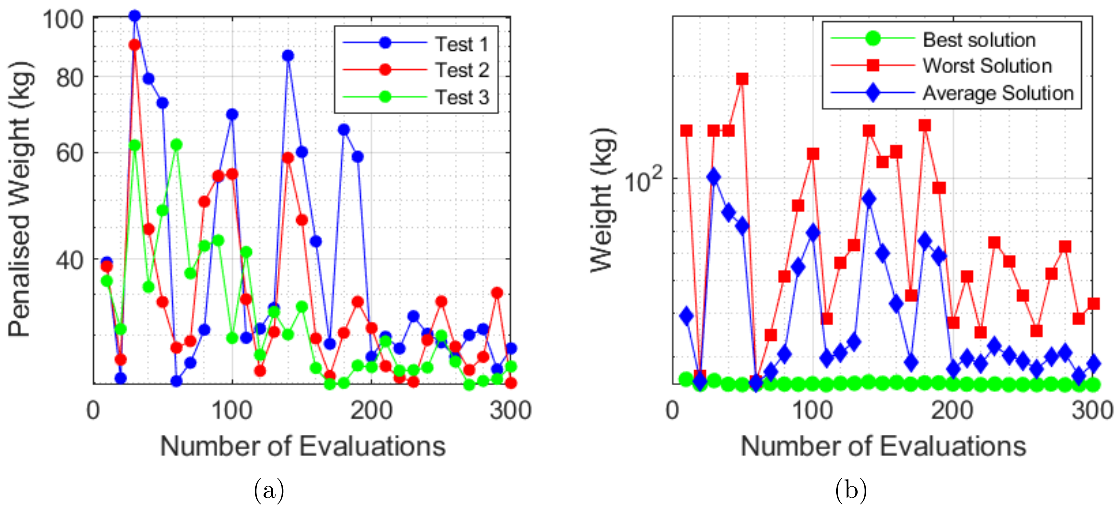


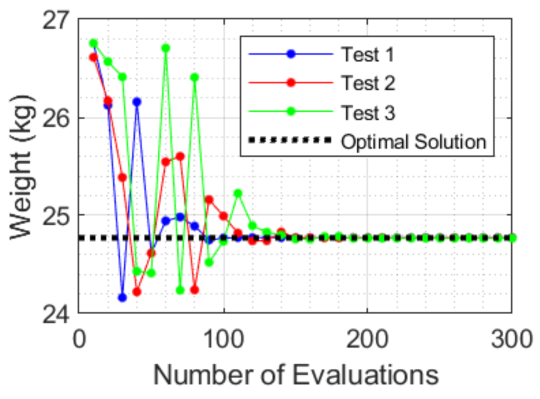
Figure 5.5: The evolution of: (a) the objective function, (b) best, average and worst values of the objective function returned for Experiment β.

Problem I – Experiment Ω

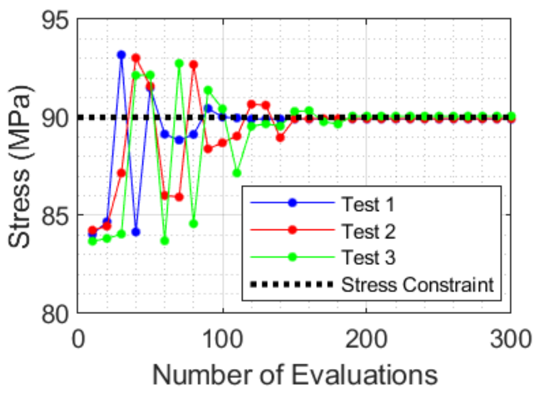
As seen in the previous experiment, due to a higher inertia weight (ω) parameter, no test resulted in an early convergence. Decreasing the inertia weight to $\omega = 0.25$ in the algorithm results in particle velocities which are not as affected by their assigned velocity in the precedent iteration, hence each particle has the tendency to decelerate towards the optimal solution.

It is interesting to notice that all tests in this experiment converged quickly, around ~ 140 evaluations. The reduced inertia guided all the particles towards the best know location. Comparing the evolution of the objective function in Figure 5.6a, or the returned stress in Figure 5.6b with the equivalents in Experiment β, it is seen that although the search space is more reduced, the particles achieve the desired solution faster.

The optimal search criteria would be covering a large search space in initial iterations and once evidences of the potential best particle location are found, all the other particles can be guided and search towards that location.



(a)



(b)

Figure 5.6: Problem I - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment Ω.

It is safe to assume that when optimising the system's weight with only one variable, a low inertia for the algorithm displays a high performance level for this problem. Nevertheless, there is one more parameter that needs to be tested in order to evaluate what consequences it might induce in the optimisation procedure. Furthermore, looking at Figure 5.7 it is possible to see that the solutions that violate the maximum stress constraint are being guided towards solutions that do not, hence starting to converge at after ~ 120 evaluations.

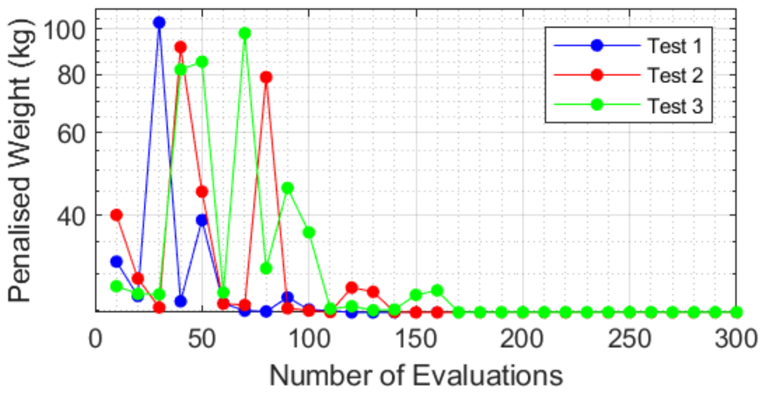


Figure 5.7: Problem I - The evolution of the objective function with applied penalties for Experiment Ω.

Problem I – Experiment γ

The objective of this experiment is to evaluate the consequences of a lower penalty parameter in the optimisation procedure, making $r = 2$. In Figure 5.8a is shown the evolution of the weight returned by the algorithm and it is possible to see that it successfully was able to converge into a solution. However, looking at Figure 5.8b, it is noticeable that the returned solutions from Test 2 and Test 3 violate the defined stress constraint. Effectively, this penalty parameter is not sufficiently high to force the algorithm to search solutions that do not violate the maximum allowed stress. Nevertheless, the algorithm managed to find solutions that result in configurations with measured stresses close to the defined stress limit, as a result of the values of stress being numerically superior to the ones of the objective function.

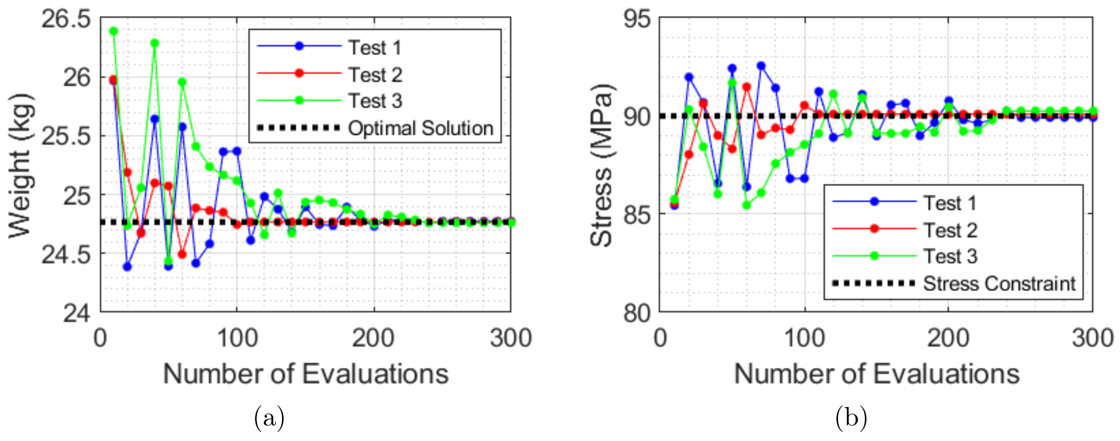


Figure 5.8: Problem I - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment γ.

5.1.3 Results and Discussion

Throughout the four experiments in this problem, Experiment Ω proved to be more efficient in returning optimal solutions with great precision, as the standard deviation for that experiment is significantly inferior to the other ones. Regardless, every experiment resulted in success since the algorithm was able to find the best solution in each one of them. The overall results can be found in Tables 5.3 and 5.4.

Table 5.3: Problem I results.

	Experiment			
	α	β	Ω	γ
Best solution (mm)	9.83706	9.83620	9.83739	9.84214
Average (mm)	9.86933	9.98270	9.85549	9.81015
Standard deviation	0.34084	0.31269	0.019926	0.14079
Resulting weight (kg)	24.76928	24.76919	24.76931	24.76980

Table 5.4: Number of evaluations and computational times for Problem I.

		Experiment			
		α	β	Ω	γ
Number of evaluations	Test 1	210	300	120	250
	Test 2	160	300	150	110
	Test 3	250	300	190	240
Computational times (s)	Test 1	10812	19160	6114	14707
	Test 2	8107	14369	7586	5562
	Test 3	12818	15335	11177	10943

Using the solution obtained from Experiment Ω, the model analysed features an interlayer thickness of $\sim$ 9.83739 mm. In Figure 5.9 the stress wave measured in the frontal impact region of each plate and the rear impact region of Plate C is plotted. Focusing on the orange curve, which resembles the stress wave measured in the frontal impact region of Plate B, it is noticeable that the maximum stress occurs approximately at $\sim$ 180 μs and its magnitude is $\sim$ 90 MPa. It is also seen that the magnitude of the maximum stress measured at the frontal impact region of Plate B is approximately double of the one measured in Plate B, which means that, with this configuration, Plate A dampens nearly half of the stress.

Analysing the model in terms of energy transfer, in Figure 5.10 it is shown the evolution of the strain energy in the three plates as well as the energy dissipated through plastic strain. It is important to state that, unlike the plastic strain energy, the strain energy is recoverable because it results from elastic deformation. In Figure 5.10a, it can be seen that the maximum strain energy generated is $\sim$ 2.5 J and originates from Plate A, which resembles approximately half of the strain energy in the system at $\sim$ 200 μs. The plastic dissipation energy, however, originates exclusively from Plate A, which means that there is no plastic deformation in Plates B and C, as seen in Figure 5.10b. It is also relevant to highlight that the amount of energy dissipated from plastic deformation is considerably higher than the energy generated from elastic strain.

The elastic strain, plastic dissipation and artificial strain energies are a part of the

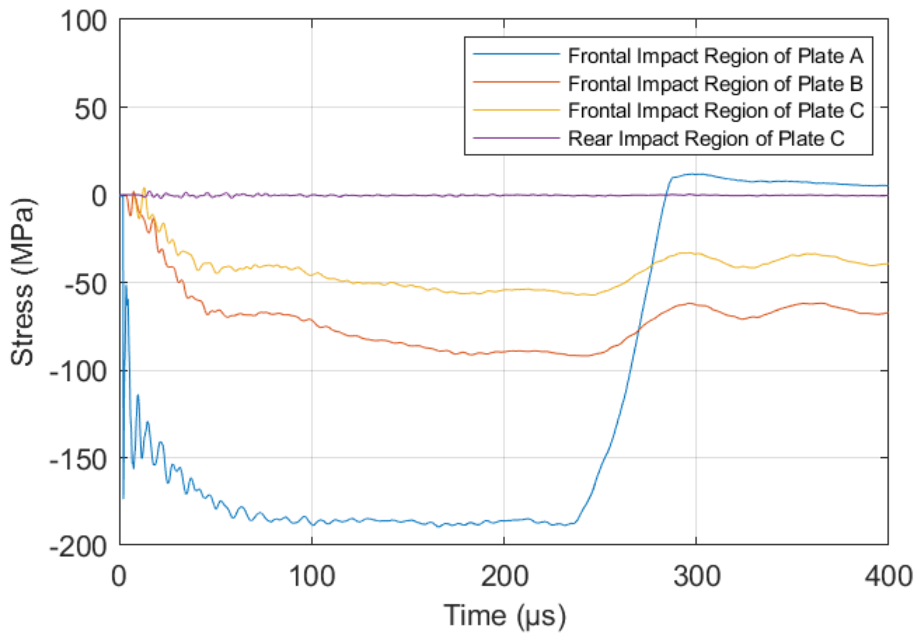
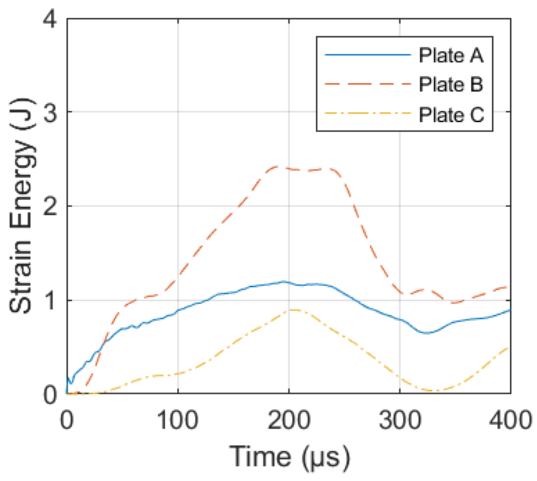


Figure 5.9: The evolution of the measured stress in the frontal impact region of Plates A, B and C and in the rear impact region of Plate C.

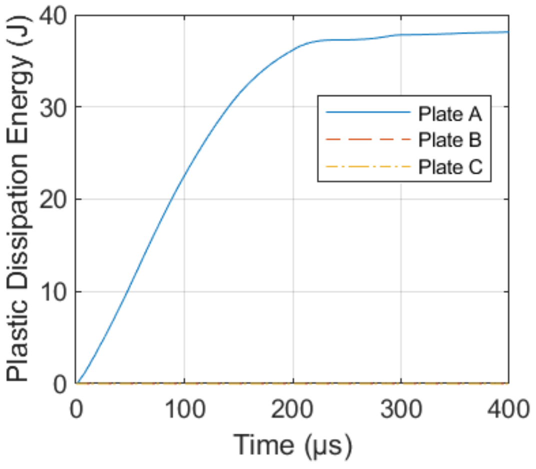
internal energy on the system. The artificial strain energy is primarily the energy dissipated to control hourglassing deformation [Smith 2009] in the finite elements. If it is excessive, too much strain energy may be going into controlling hourglassing deformation, hence it is important that the artificial strain energy is considerably lower than the internal energy of the system. In case it is excessive, the mesh of the model needs to be adjusted. In Figure 5.11 it is possible to see the evolution of the kinetic energy from the projectile, the internal energy of each plate and the generated artificial strain energy. It can be noticed that the majority of the kinetic energy of the projectile is transferred into Plate A. The artificial strain energy generated is not excessive so it does not represent an issue for the used mesh. In Table 5.5 it is possible to see the initial and final values of the kinetic energy of the projectile, as well as its respective final velocity magnitude.

Table 5.5: Variation of kinetic energy of the projectile.

Initial kinetic energy (J)	Final kinetic energy (J)	Fraction of energy transferred (%)	Final velocity (m/s)
45.94531	0.99994	97.82363	3.68813



(a) Elastic strain energy.



(b) Plastic Dissipation Energy.

Figure 5.10: Generated elastic and plastic strain energies at each plate.

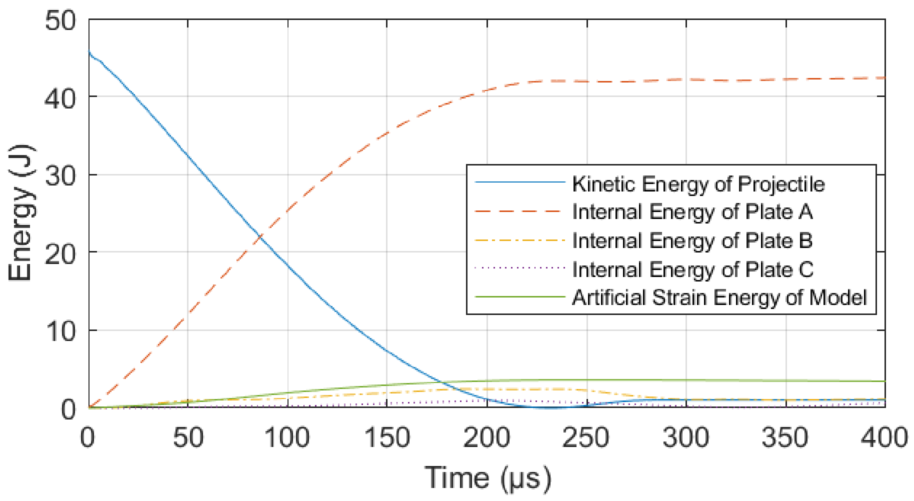


Figure 5.11: Evolution of the kinetic energy of the projectile, the internal energy of the plates and the artificial strain energy of the model.

5.2 Problem II — Multi-layer Armour Plate Impacted by Projectile with a Velocity of 40 m/s

The second problem is very similar to the previous one. Although revolving around the same objectives, this problem features the optimisation of a 3-plate model, focusing in finding the optimal values for three variables: the thickness of Plate A, B and C. Since the algorithm is searching for the optimal combination of the thickness of each plate, it is essential that the particles are able to find that optimal location, otherwise they are guided towards a local minimum instead of the desired global one.

Following the same structure as the previous problems, this one is broken down in three experiments. The impact velocity for this problem is raised to 40 m/s and the returned stress is evaluated in the back-plate (Plate C). This means that the main source of stress outputs is due to stress waves and depending on the conditions of simulation the results might differ. The solutions obtained by the algorithm are analysed in Section 5.2.3, where it is possible to study if there is propagation of elastic and plastic stress waves. Nevertheless, the optimal parameters are tested along the three experiments of this problem.

The setup is represented in Figure 5.12 and described in the items below:

- **Plate Geometry**: Square plates of 300 mm of width;
- **Number of plates**: 3 (A, B and C);
- **Plate Boundary Condition**: Side surfaces of each plate are clamped;
- **Projectile Velocity**: 40 m/s.

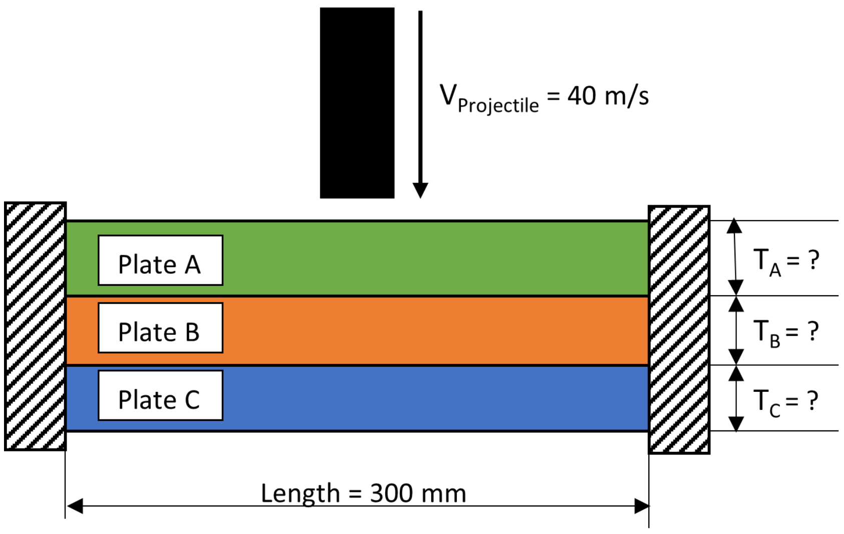


Figure 5.12: Diagram for Problem II.

The materials to be used for this problem are the same as the ones in Table 5.1.

5.2.1 Problem Formulation

The objective of this problem is to find the lightest 3-layer armour system able to limit the returned stress on the back-plate to the assigned amount. The material data for the plates is the same as in Problem I. The stress is analysed in the impact region of the rear surface of Plate C, which is shown in Figure 5.13.

This problem is formulated as

$$\begin{aligned} \text{minimise} \quad & f(t_\mathrm{A},\, t_\mathrm{B},\, t_\mathrm{C}, r) = W(t_\mathrm{A},\, t_\mathrm{B},\, t_\mathrm{C}) + P(t_\mathrm{A},\, t_\mathrm{B},\, t_\mathrm{C}, r)\ , \\ \text{subjected to} \quad & t_{\mathrm{A,\,min}} \le t_\mathrm{A} \le t_{\mathrm{A,\,max}}\ , \\ & t_{\mathrm{B,\,min}} \le t_\mathrm{B} \le t_{\mathrm{B,\,max}}\ , \\ & t_{\mathrm{C,\,min}} \le t_\mathrm{C} \le t_{\mathrm{C,\,max}}\ , \\ & g(x) = \sigma_\mathrm{z} - \sigma_{\mathrm{z,\,max}} \le 0\ , \end{aligned} \tag{5.3}$$

where the weight function is defined as

$$\begin{aligned} W(t_{\text{A}}, t_{\text{B}}, t_{\text{C}}) &= 4 \times L_{1,\text{A}} \times L_{2,\text{A}} \times \left(\rho_{\text{A}} t_{\text{A}} + \rho_{\text{B}} t_{\text{B}} + \rho_{\text{C}} t_{\text{C}}\right) \\ &= 4 \times 0.15^2 \times \left(2710 \times t_{\text{A}} + 1140 \times t_{\text{B}} + 7850 \times t_{\text{C}}\right) , \end{aligned} \tag{5.4}$$

and the variables t_{A}, t_{B} and t_{C} define the thickness between a minimum of 10 mm and a maximum of 50 mm for Plates A,B and C, respectively. The maximum stress allowed is measured in the rear impact region of Plate C and is set as $\sigma_{\text{z, max}} = 1$ MPa. This defined stress constraint value is numerically inferior to the values of the objective function and, thus, serves as an opportunity to understand how it affects algorithm's capabilities of finding an optimal solution.

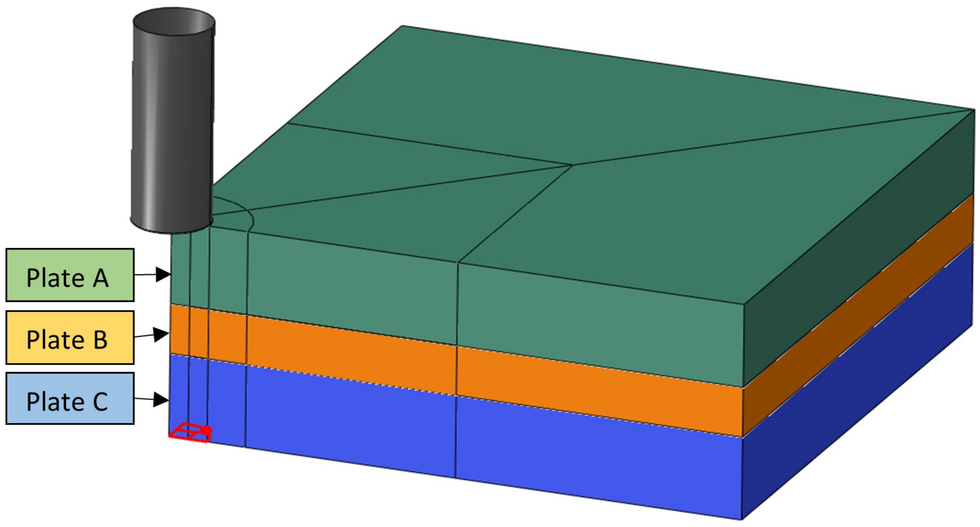


Figure 5.13: Representation of the region where the stress is measured for Problem II.

5.2.2 Setup and Implementation

This optimisation problem features three different variables, which means that there is a large amount of possible solutions and the algorithm may struggle to find the best one. Under this circumstances, it is important to apply the PSO operational parameter used in Experiment Ω of Problem I, which is the one that performed better. This problem is then divided into three different experiments. The first experiment analyses how the best operational parameters from the previous problem while the second one focuses on understanding, how increasing the inertia weight can lead to a larger search space and, consequently, enhance the performance of the algorithm. Furthermore, the number of particles is increased to 15, providing a larger number of solutions and a larger search space. Nevertheless, given that the defined stress constraint is 1 MPa, the penalty parameter needs to be considerably high. This leads to the development of Experiment Ω, making the penalty parameter $r = 1000$. The algorithm's operational parameters are listed in Table 5.6.

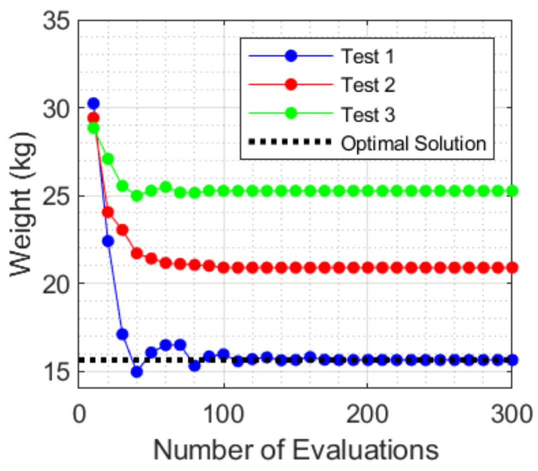
Table 5.6: Problem II - PSO operational parameters for the experiments

Experiment	Number of particles	Number of iterations	ω	c_1	c_2	r
α	10	30	0.25	1	2	5
β	15	30	0.5	1	2	5
Ω	15	30	0.5	1	2	1000

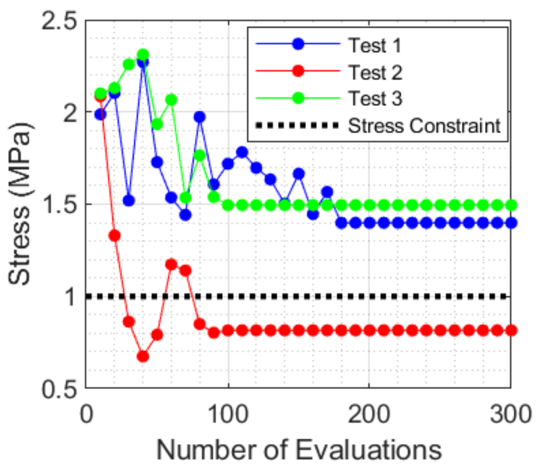
Problem II – Experiment α

The parameters for the first experiment of this problem are imported from Experiment Ω of the previous experiment, as it is the one with the best performance. The parameters are listed in Table 5.6.

With this selection of parameters, an early convergence was obtained for each of the three experiments, as seen in Figure 5.14a, however, the obtained solutions for Tests 1 and 3 do not agree with the established maximum stress limit. In Figure 5.14b the particles in the mentioned tests tend to converge to a returned stress around $\sigma_z = 1.5$ MPa. The reason behind the violation of the maximum allowed stress condition comes from an inadequate choice of the r penalty parameter. The poor choice of the mentioned parameter reflects in solutions that do not comply with the defined stress constraint. It can be seen in Figure 5.15 that the objective function for Test 1 is not penalised enough for violating the defined stress constraint, returning a solution that, although violating the defined stress constraint, the applied penalty to the weight is negligible. Furthermore, a low inertia weight ω value and the usage of a small number of particles can harm the overall search potential of the algorithm.



(a)



(b)

Figure 5.14: Problem II - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment α.

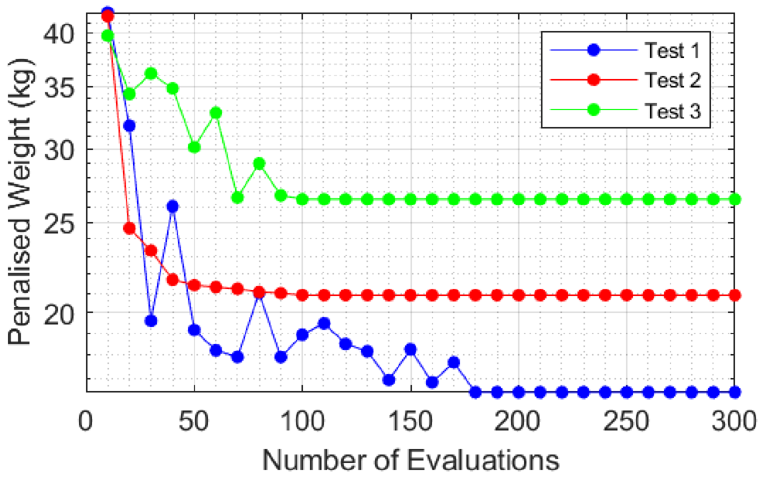


Figure 5.15: Problem II - The evolution of the objective function with applied penalties for Experiment α.

Problem II – Experiment β

The second experiment of Problem II serves the purpose of evaluating the impact of an increased search space on the algorithm's performance. The main difference is the addition of five more particles and an increased inertia weight, making $\omega = 0.5$.

The results in Figure 5.16a indicate an early convergence, however, taking a look at the plot in Figure 5.16b the convergence returned stress is close to $\sigma_z = 1.2$ MPa, violating the stress restrictions. While the same happened in the previous experiment, the returned convergence maximum stress is relatively inferior to the obtained in Experiment α, proving the importance of covering a larger search space. It is expected that having a penalty parameter of $r = 5$ is not ideal, however, using the objective of this experiment is to analyse how increasing the search space can provide a better algorithm performance, hence a correction to the penalty parameter is left for Experiment Ω. Looking at Figure 5.17 it can be seen that the algorithm does in fact converge towards better solutions than possible because the penalty parameter is not penalising sufficiently the objective function.

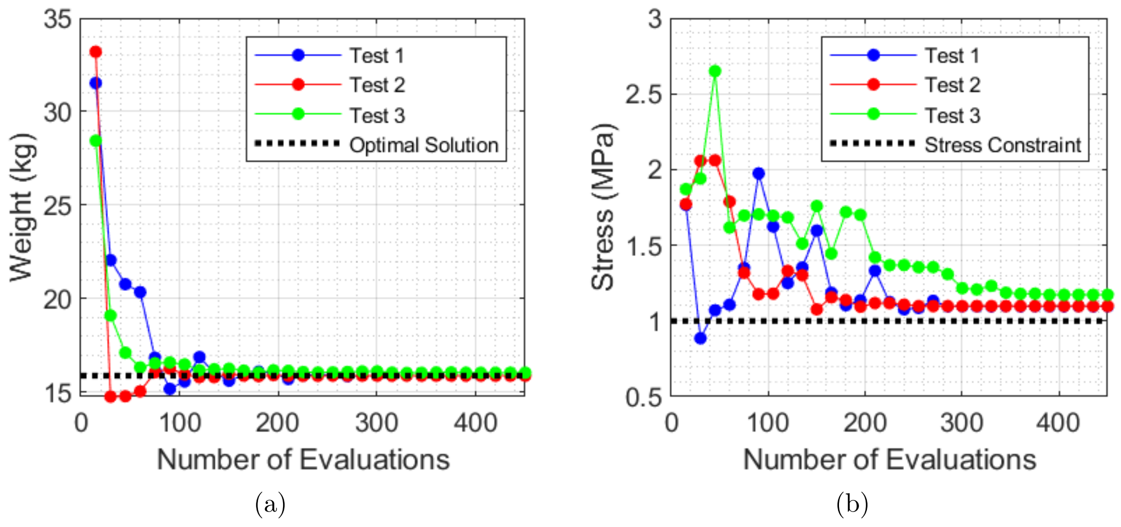


Figure 5.16: Problem II - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment β.

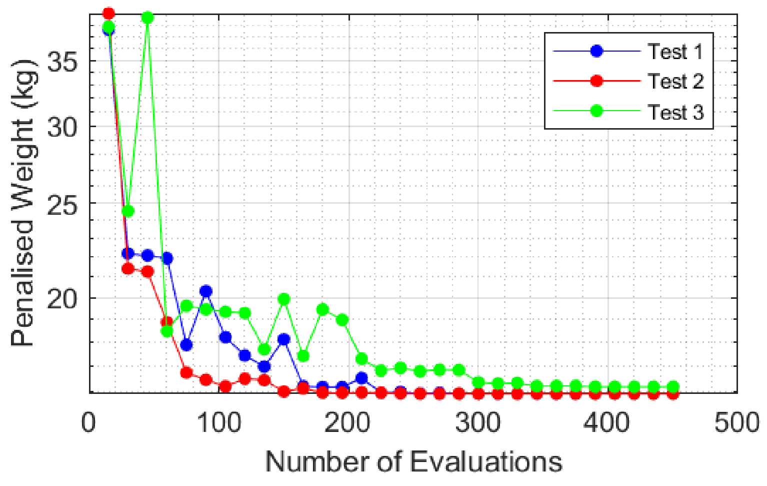
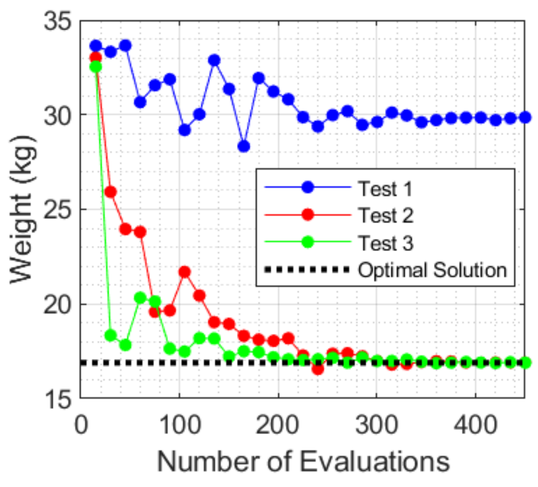


Figure 5.17: Problem II - The evolution of the objective function with applied penalties for Experiment β.

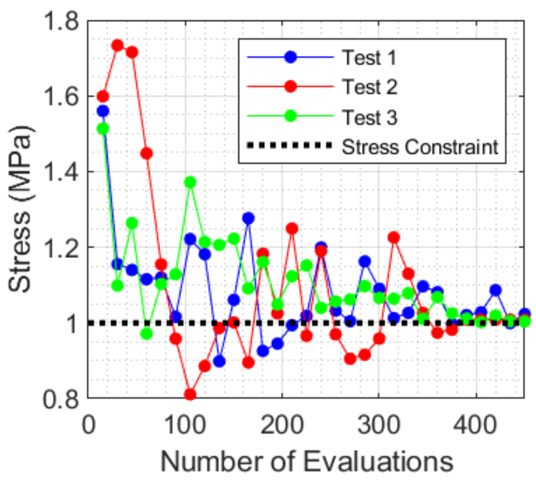
Problem II – Experiment Ω

In this experiment, the penalty parameter is greatly increased to ensure that the imposed maximum returned stress limit is respected, solving the issues in the previous experiments.

From Figure 5.18a it is possible to see that Tests 2 and 3 converged to the optimal solution as opposed to Test 1. The first test manages to converge into a solution that, while respecting the defined stress constraint, is not optimal because the focus is in minimising the plate's weight and this solution is $\sim$ 13 kg heavier than the other two. Regardless, Tests 2 and 3 were capable of finding two good solutions for this problem with the one provided from Test 3 being the better. The stress constraint is well implemented in this case as Tests 2 and 3 final results register a maximum returned stress of $\sigma_z = 1.0$ MPa as shown in Figure 5.18b.



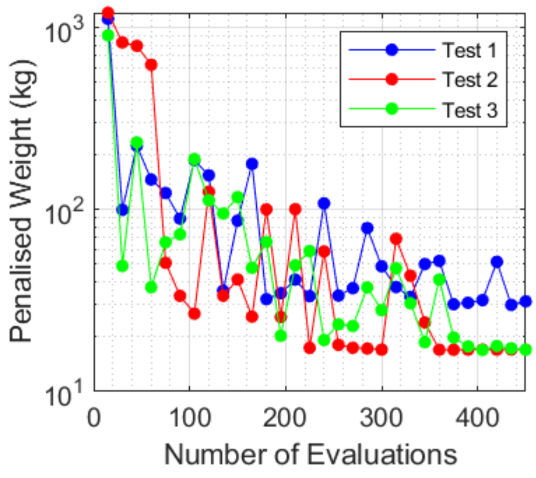
(a)



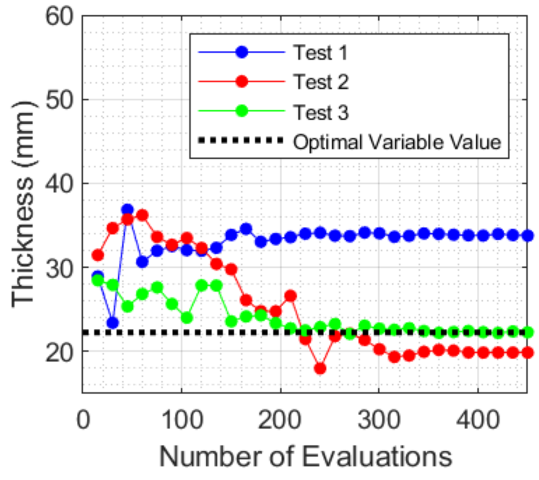
(b)

Figure 5.18: Problem II - The evolution of: (a) the weight function, (b) maximum stress returned for Experiment Ω.

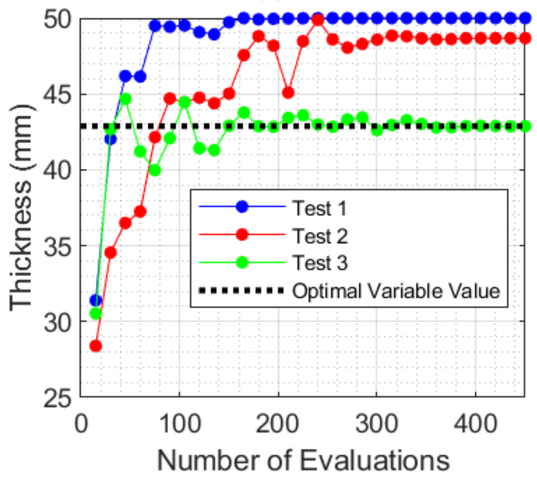
The evolution of the objective function is shown in Figure 5.19a. To visualise the optimal variables the algorithm used to find the solutions, Figures 5.19b, 5.19c and 5.19d display the evolution of the variable used for the thickness of each plate.



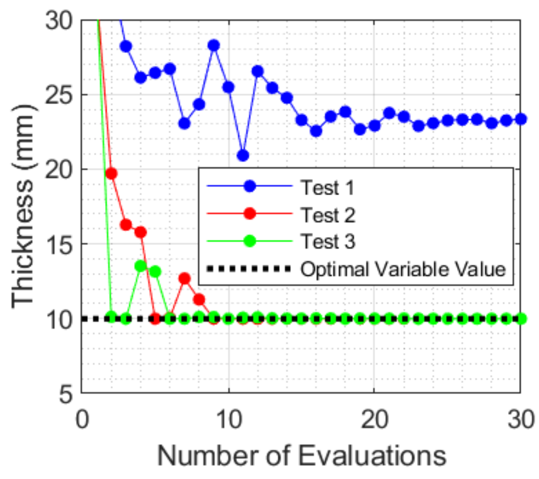
(a)



(b)



(c)



(d)

Figure 5.19: Problem II - The evolution of: (a) the objective function with applied penalties (b) the evolution of the thickness assigned to Plate A (c) the evolution of the thickness assigned to Plate B (d) the evolution of the thickness assigned to Plate C.

5.2.3 Results and Discussion

Overall the results for Problem II are listed in Table 5.7 and while Experiment Ω was the only one that was able to find the optimal solution, its first test was not able to find converge towards a solution at all. This explains why the average value of the final weight of the system is considerably different. Furthermore, the standard deviation calculations testify how the results are spread out. It is also interesting to check the two returned solutions in the Experiment Ω from Tests 2 and 3. Looking at Figure 5.18a it is possible to see that for these two experiments, the solution is very similar. However, when visualising Figures 5.19b and 5.19c it is possible to see that, in fact, the algorithm used two different values of thickness for Plate A and B. This means that these two possible configurations respect the imposed stress constraint while providing a light-weight solution.

Regardless, the results from Test 3 provide the lighter solution even if only just slightly and with the advantage of being a more compact system, as shown in the first figure of

Appendix A.5, where it can be seen that the total thickness of the armour system is ~ 3 mm thinner.

The computational times were longer along Experiment Ω mainly due to the termination criteria of the algorithm being satisfied towards latter iterations as opposed to the other Experiments. The first one was undoubtedly the faster due to an inferior amount of particles and early convergence towards the found solution.

Table 5.7: Problem II results.

	Experiment		
	α	β	Ω
Best solution for Plate A (mm)	21.44406	15.10821	22.28785
Average solution for Plate A (mm)	28.59180	16.92809	25.31797
Best solution for Plate B (mm)	32.50873	50.0	42.87540
Average solution for Plate B (mm)	25.35845	46.20260	47.18425
Best solution for Plate C (mm)	10.0	10.0	10.0
Average solution for Plate C (mm)	15.59538	10.0	14.44824
Standard deviation	4.00524	0.11271	6.06289
Resulting Weight (kg)	15.63060	15.87989	16.90003

Table 5.8: Number of evaluations and computational times for Problem II.

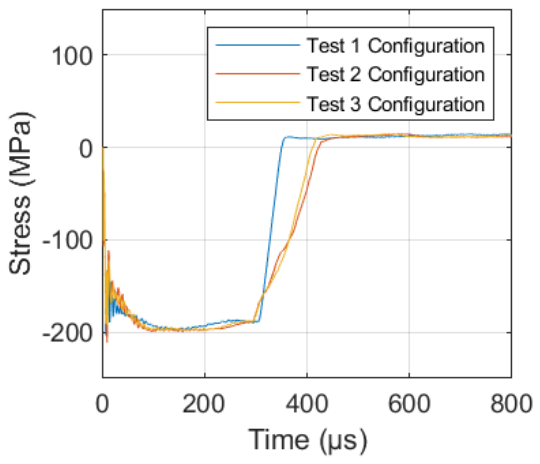
		Experiment		
		α	β	Ω
Number of evaluations	Test 1	180	285	450
	Test 2	100	255	390
	Test 3	100	390	450
Computational times (s)	Test 1	9245	20624	30658
	Test 2	7268	14555	21848
	Test 3	6294	24546	28323

In Experiment Ω it was possible to obtain three different solutions with the one from Test 3 being the best. Regardless, the three solutions are analysed in terms of stress wave propagation and energy transfers. The resulting thickness details for the configuration of the three solutions are listed in Table 5.9.

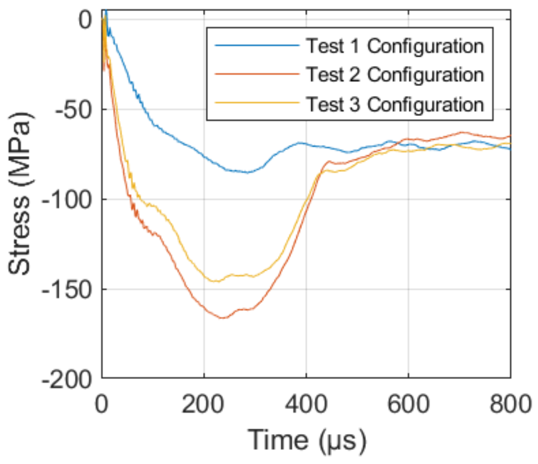
Table 5.9: Details of the thickness of the plates to analyse as different solutions to Problem II.

	Thickness of Plate (mm)			Weight (kg)
Configuration	A	B	C	
Test 1	33.85171	49.99991	23.25127	29.81345
Test 2	19.88531	48.67742	10.0	16.90933
Test 3	22.2878	42.87540	10.0	16.90001

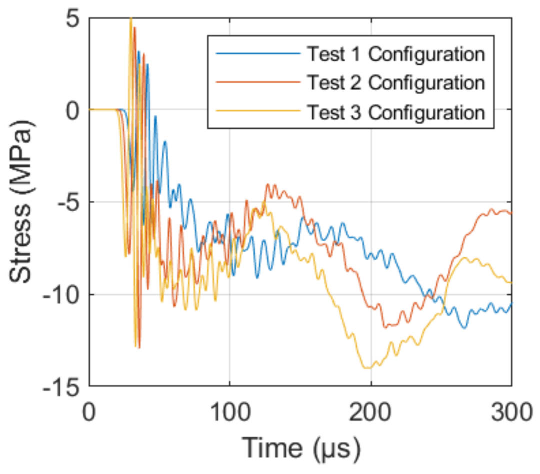
In order to analysing how the stress wave propagates in each one of the solutions obtained, Figure 5.20 shows the stress wave analysed in four different regions for the three resultant configurations. It is possible to see that from Figure 5.20a the shape of the stress wave for Tests 2 and 3 are identical, while for the configuration of Test 1 the duration of the pulse is relatively shorter. Since the configuration of Test 1 involves a thicker Plate A, the amount of plastic strain is smaller, thus resulting in shorter time of contact between projectile and plate, as explained in Section 3.7.1. In Figure 5.21 it is plotted the plastic equivalent strain along the elements in the centre of the plate, starting from the front surface of Plate A and ending in the rear surface of Plate C. This graph proves that the total plastic strain generated in the configuration of Test 1 is indeed smaller. It is also noticeable that there is no plastic equivalent strain beyond Plate A for any of the tests' configuration, which is expected due to the large amount of stress dampened throughout Plate A.



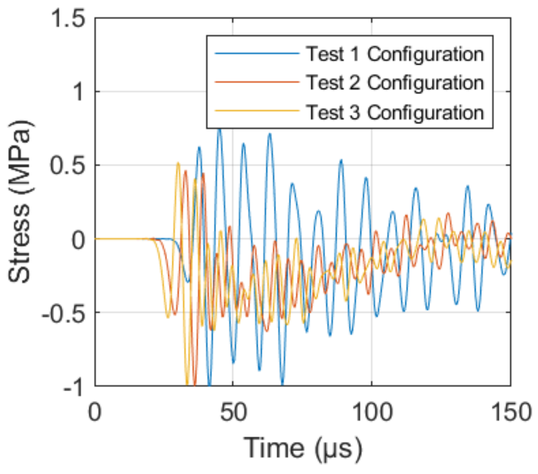
(a)



(b)



(c)



(d)

Figure 5.20: Average stress in OZ direction for the resulting configuration from Tests 1 to 3, measured at: (a) impact region in front surface of Plate A, (b) impact region in front surface of Plate B, (c) impact region in front surface of Plate C, (d) impact region in rear surface of Plate C.

Looking at Figure 5.22 it is possible to analyse and compare the energy transfers in the system between the three configurations. It is interesting to notice in Figure 5.22a that the maximum amount of energy dissipated through plastic strain is higher for the

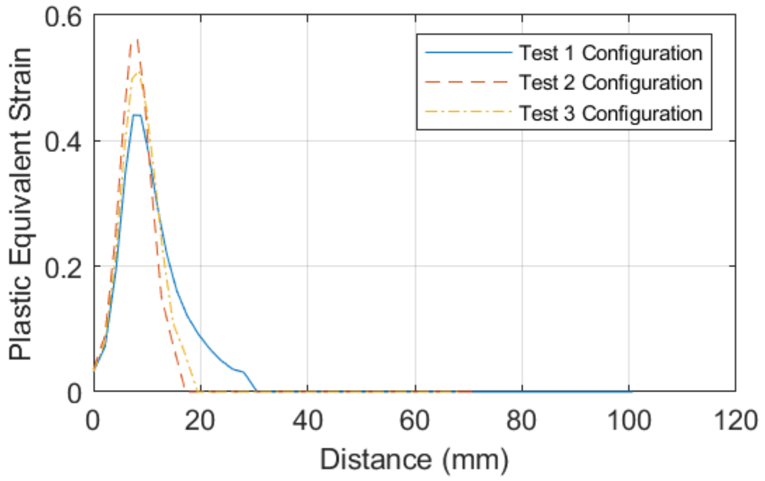


Figure 5.21: Plastic equivalent strain throughout the centre elements of the armour plate, starting from the front surface of Plate A and ending in the rear surface of Plate C.

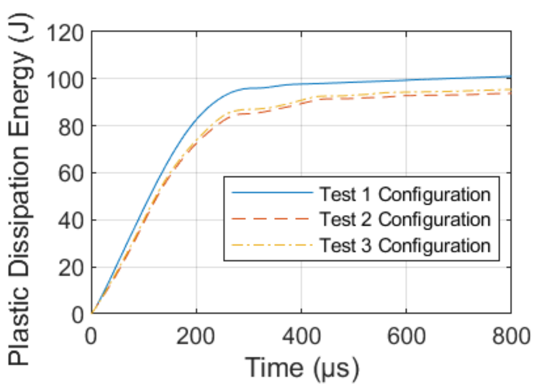
configuration of Test 1, even though it was shown that the maximum amount of equivalent plastic strain was smaller. This is a result of a smaller amount of strain energy transferred to Plate B, as seen in Figure 5.22c. For the configurations of Tests 2 and 3 it can be seen that a larger amount of elastic strain energy is transferred into Plate B. Regardless, it is clear that most of the energy transferred into the armour system is absorbed in Plate A, where there is energy dissipated under the form of plastic deformation.

The kinetic energy of the projectile and internal energy of each plate evolution is shown in Figure 5.23, as well as the artificial strain energy. It can be noticed that in Figure 5.23a, the kinetic energy transfer of the projectile for the configuration of Test 1 is the highest of all three, since its final kinetic energy is smaller. The artificial strain energy generated for all three solutions is much smaller than the sum of the internal energies of all plates, thus it does not represent an issue for the used mesh. Overall, it can be concluded that, regardless of the three obtained solutions, Plate A absorbs the highest amount of energy from the projectile while Plate C absorbs the least, and as a result, the thickness of Plate C has the lesser effect on the absorption of energy. However, the configuration from Test 1 features a higher thickness for Plate C when compared to the other experiments and is the configuration that absorbs the highest amount of energy from the projectile, at the cost of being the heaviest solution. The initial and final values of the kinetic energy of the projectile, as well as its respective final velocity magnitude are listed in Table 5.10.

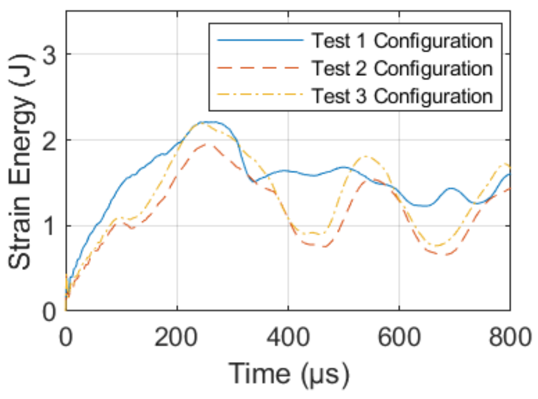
Table 5.10: Variation of kinetic energy of the projectile in Problem II.

Configuration	Initial kinetic energy (J)	Final kinetic energy (J)	Fraction of energy transferred (%)	Final velocity (m/s)
Test 1	117.620	1.48474	98.73768	4.49412
Test 2	117.620	4.86962	95.85987	8.13892
Test 3	117.620	4.10498	96.50996	7.47265

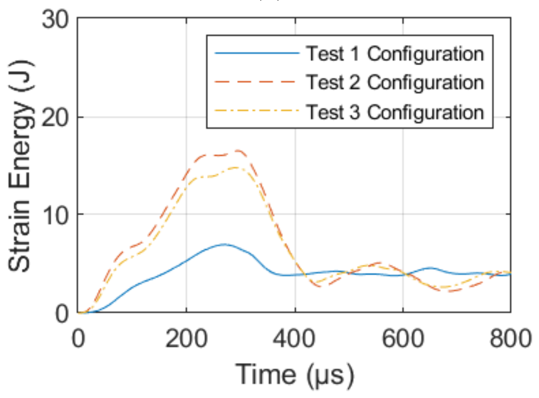
Finally, the last figure of Appendix A.5 shows the evolution of the vertical displace-



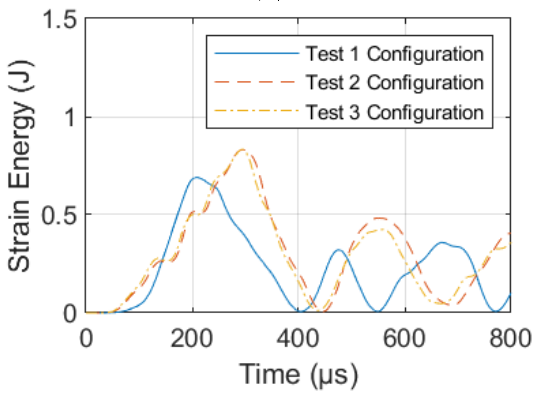
(a)



(b)



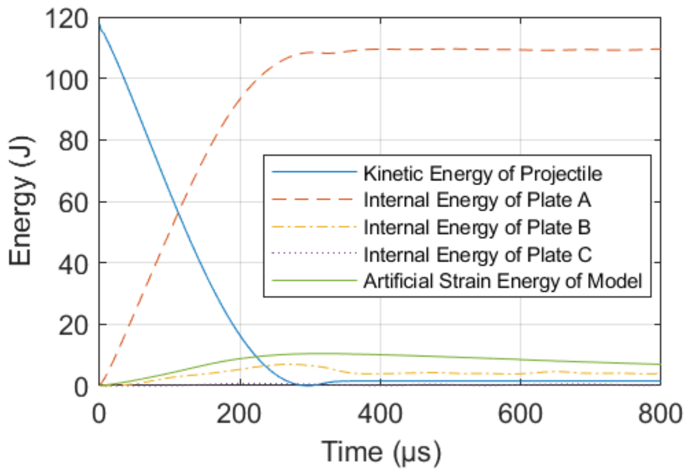
(c)



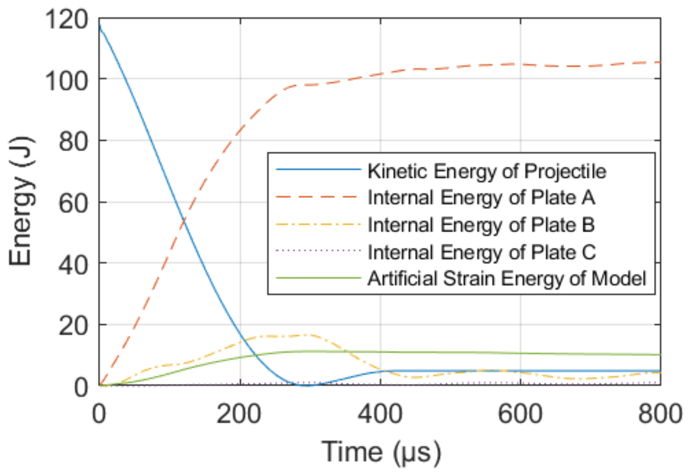
(d)

Figure 5.22: Analysis of the strain and plastic dissipation energy in the three plates for all the configurations: (a) plastic dissipation energy in the model, (b) elastic strain energy in Plate A, (c) elastic strain energy in Plate B, (d) elastic strain energy in Plate C.

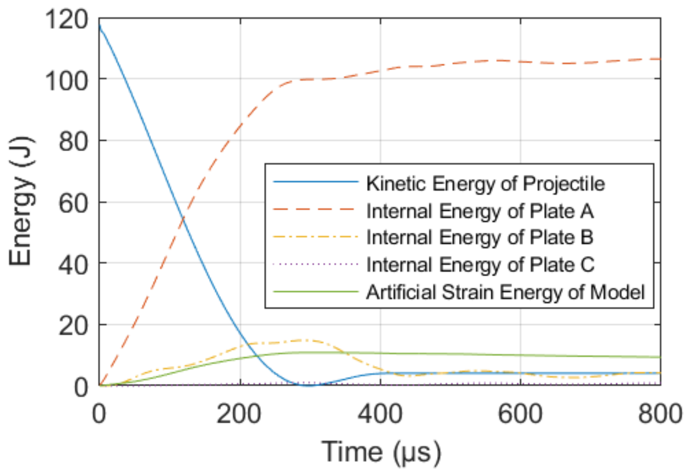
ment in the centre area of the rear surface of Plate C. While the configurations of Tests 2 an 3 have similar displacement magnitudes, the configuration of Test 1 shows a much smaller vertical displacement. This is expected since the configuration of Test 1 resembles the solution with the thickest Plate C, which is ~ 13 mm thicker than the Plate C of the other configurations.



(a) Configuration of Test 1



(b) Configuration of Test 2



(c) Configuration of Test 3

Figure 5.23: Evolution of the kinetic energy of the projectile, internal energy of the plates and generated artificial strain energy.

Intentionally blank page.

Chapter 6

Final Remarks

6.1 Main Conclusions

This dissertation analyses the propagation of stress waves in layered material structures, generated from an impact of a moving projectile. Based on a previous work, a Python script that is able to be interpreted by Abaqus was developed, resulting in an easy and fast method of building models to be simulated by the software. The development of the script facilitated numerous tasks during the development and testing of the model, providing a useful amount of possible parameters while also eliminating several time consuming tasks. Furthermore, it continuously expanded into a versatile utensil that could be used for post processing and monitoring while particularly accessible.

Chapter 3 played a vital role in developing the script as well as providing tools to incorporate the knowledge discussed in Chapter 2 and to test it in Abaqus, towards stress wave propagation validation purposes. In Section 3.6, the conducted analyses confirm the theoretical elastic and plastic stress wave speeds. Additionally, a study is conducted throughout Section 3.7 to further understand the effects of plastic deformation on the properties of the produced stress wave, such as the wavelength and magnitude. It is demonstrated that, with plastic deformation, the wavelength of the pulse increases, while the magnitude of the induced stress on the plate decreases. It is also shown throughout Chapter 3 that the generated stress waves generally display two characteristic peaks of stress. The magnitude of the measured stress in these two peaks evolve differently along the thickness of the plate, with the dampening of the first stress peak being higher as the velocity of the projectile increases, while the second one decreases with higher impact velocities. Moreover, for three different tested metals, it is shown that, despite differences in magnitude of stresses according to the material being used, the extracted stress peaks along the plate's thickness evolve in similar ways for the three tested metals.

In this work, the Particle Swarm Optimisation algorithm is successfully implemented and, despite its stochastic nature, is able to perform efficiently and find solutions to the assigned problems. In the latter part of Chapter 4, a simple optimisation problem aimed to find the optimal thickness for the interlayer plate in a 3-layer armour system is conducted. Serving as an opportunity for preliminary operating parameter adjustments of the algorithm, it is possible to find how the algorithm behaves and how it affects its capabilities in finding the optimal solution for the assigned problem. Furthermore, clamping the side surfaces of the plates resulted in lower generated stress magnitudes, when comparing to similar configurations with a clamped rear surface of the back plate.

Additionally, it is shown that, despite using the same boundary conditions, increasing the length of the plates slightly affects the protection capabilities of the armour, resulting in a need to increase the thickness of the plate in order to withstand the maximum defined amount of stress.

Taking into account a realistic approach to ballistic tests, Chapter 5 explores the effects of higher velocity impacts and, consequently, plastic deformation when searching for the ideal value for the thickness of the interlayer plate. Problem I serves the purpose of further enhance the computational efficiency of the algorithm. It is demonstrated that for an impact velocity of 25 m/s, a thickness of $\sim$ 9.8 mm for the Nylon-6 interlayer plate is needed to prevent plastic deformation in it. The stochastic nature of the algorithm is noticeable throughout Problem II, as it is solved through the search of a three variables. The problem aims to find the best combination of thickness of all three plates in order for the armour system to be able to withstand the generated stress on the centre region of the back plate, when impacted at a velocity of 40 m/s. The algorithm struggles to find solutions that do not violate the maximum stress restrictions, however, dividing the problem in various experiments helped setting up the ideal operating parameters in order to guide the algorithm towards the desired solution. In this problem, it is also possible to notice that the thickness of the steel back plate played the lesser role in weight minimisation, as the algorithm continually finds solutions where the thickness of the mentioned plate is set up to the minimum allowed value. Overall, a weight minimisation problem with three variables generates a spectrum of local minimums that the algorithm often converges into, without finding the global minimum, which is the best possible solution for this problem. Moreover, the importance of tuning the operational parameters of the algorithm to enhance its efficiency and accuracy is highlighted throughout this problem.

6.2 Further Work

This dissertation provides a useful tool through a flexible Python script, which facilitates the Abaqus model generation process. The script contains a large number of customisable parameters and the Particle Swarm Optimisation algorithm. It constitutes a powerful tool that can be used to find light-weight armour solutions efficiently and accurately, focusing on the minimisation of time consuming actions and inputs when developing a ballistic test model in Abaqus. Taking a more realistic approach to modern protection systems, incorporating new types of materials such as composites and fibres and exploring different designs can lead to interesting ballistic configurations. Furthermore, integrating damage models in the material's properties may provide scenarios where simulating high velocity impacts is possible. Applying the damage models can provide opportunities to study the behaviour of the material with penetration from the projectile, which can result in various modes of penetration depending on the type of projectile. Moreover, the Particle Swarm Optimisation can be improved to be versatile towards solving of complex and dynamic problems with different types of variables such as number of plates or different combinations of complex materials. Other interesting design variables may be incorporated into the algorithm to provide a light-weight, non expensive armour system that can be experimentally tested to compare results.

Bibliography

[Andrade-Campos *et al.* 2015] A. Andrade-Campos, J. Dias-de Oliveira and J. Pinho-da Cruz. Otimização Não-Linear em Engenharia. ETEP, 2015.

[Arora 2016] J. S. Arora. Introduction to Optimum Design. Elsevier Science, 2016.

[Ashby 2005] M. F. Ashby. Hybrids to fill holes in material property space. *Philosophical Magazine*, 85(26-27 SPEC. ISS.):3235–3257, 2005.

[Azevedo 2012] A. V. F. Azevedo. Desempenho balístico de sistemas de proteção com núcleos de MAC. Master's thesis, Universidade de Aveiro, Portugal, 2012.

[Chen 2001] Shen Yeh Chen. An approach for impact structure optimization using the robust genetic algorithm. *Finite Elements in Analysis and Design*, 37(5):431–446, 2001.

[Crouch 2019] I. G. Crouch. Body armour – New materials, new systems. *Defence Technology*, 15(3):241–253, 2019.

[DeHart *et al.* 2012] G. DeHart, L. Ellis, L. Givens, M. P. Hickey and J. Terp. Combat Tech [Episode 3], 2012.

[Dowling *et al.* 2013] N.E. Dowling, K.S. Prasad and R. Narayanasamy. Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue. Always learning. Pearson, 2013.

[García-González *et al.* 2015] D. García-González, M. Rodríguez-Millán, A. Vaz-Romero and A. Arias. High impact velocity on multi-layered composite of polyether ether ketone and aluminium. *Composite Interfaces*, 22(8):705–715, 2015.

[Gilson *et al.* 2020] L. Gilson, L. Rabet, A. Imad and F. Coghe. Experimental and numerical assessment of non-penetrating impacts on a composite protection and ballistic gelatine. *International Journal of Impact Engineering*, 136:103417, 2020.

[Grujicic *et al.* 2017] M. Grujicic, J. Snipes and S. Ramaswami. Ballistic-penetration resistance and flexural-stiffness optimization of a nacre-mimetic, B4C-reinforced, polyurea-matrix composite armor. *International Journal of Structural Integrity*, 8(3):341–372, 2017.

[Huang *et al.* 2012] Han Huang, Hu Qin, Zhifeng Hao and Andrew Lim. Example-based learning particle swarm optimization for continuous optimization. *Information Sciences*, 182(1):125–138, 2012.

[James 2018] D. James. Introduction to Machine Learning with Python: A Guide for Beginners in Data Science. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 1st edition, 2018.

[Jiang *et al.* 2020] W. Jiang, A. Bennett, N. Vlahopoulos and G. Zhang. A Reduced-Order Modeling Based Design and Optimization for a Lightweight Multilayer Armor Plate Against Blast and Impact. in R. Fangueiro and S. Rana, editores, *Advanced Materials for Defense*, pp. 79–93, Cham, 2020. Springer International Publishing.

[Kochenderfer and Wheeler 2019] M. J. Kochenderfer and T. A. Wheeler. Algorithms for Optimization. The MIT Press. MIT Press, 2019.

[Kędzierski *et al.* 2015] P. Kędzierski, A. Morka, G. Sławiński and T. Niezgoda. Optimization of two-component armour. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 63(No 1):173–179, 2015.

[Li and You 2019] Y. Li and Z. You. Open-section origami beams for energy absorption. *International Journal of Mechanical Sciences*, 157-158:741 – 757, 2019.

[Li *et al.* 2019] H. Li, Q. Duan, P. Zhang, R. Qu and Z. Zhang. A new method to estimate the plane strain fracture toughness of materials. *Fatigue & Fracture of Engineering Materials & Structures*, 42(2):415–424, 2019.

[Lin 1996] X. Lin. Numerical Computation of Stress Waves in Solids. John Wiley & Sons, Incorporated, 1996.

[Liu *et al.* 2003] L. Liu, Q. Zhang and P. Zhai. The Optimization Design on Metal/Ceramic FGM Armor with Neural Net and Conjugate Gradient Method. *Materials Science Forum*, 423-425:791 – 796, 2003.

[Luke 2009] S. Luke. Essentials of Metaheuristics. Lulu, 2009.

[Macaulay 2012] M. Macaulay. Introduction to Impact Engineering. Springer Netherlands, 2012.

[Martini and Barthelat 2016] R. Martini and F. Barthelat. Stretch-and-release fabrication, testing and optimization of a flexible ceramic armor inspired from fish scales. *Bioinspiration & Biomimetics*, 11(6):066001, oct 2016.

[Matusevich *et al.* 2012] A. E. Matusevich, J. C. Massa and R. A. Mancini. Computation of tensile strain-hardening exponents through the power-law relationship. *Journal of Testing and Evaluation*, 40(4), jul 2012.

[Medvedovski 2006] E. Medvedovski. Lightweight ceramic composite armour system. *Advances in Applied Ceramics*, 105(5):241–245, 2006.

[Meyers 1994] M. A. Meyers. Dynamic Behavior of Materials. Wiley-Interscience publication. Wiley, 1994.

[NIJ Standard-0101.062008] Ballistic Resistance of Body Armor. Standard, U.S. Department of Justice Office of Justice Programs, 810 Seventh Street N.W. Washington, DC 20531, July 2008.

[Park *et al.* 2005] M. Park, J. Yoo and D. Chung. An optimization of a multi-layered plate under ballistic impact. *International Journal of Solids and Structures*, 42(1):123–137, 2005.

[Pittman 2017] S. Pittman. Layered Armour Systems: the Influence of a Material's Properties on its Ability to be an Effective Interlayer. Technical report, The University of Edinburgh, 2017.

[Rahul *et al.* 2005] Rahul, D. Chakraborty and A. Dutta. Optimization of FRP composites against impact induced failure using island model parallel genetic algorithm. *Composites Science and Technology*, 65(13):2003–2013, 2005.

[Reis 2019] I. B. C. Reis. Optimization of Layer Configurations for Ballistic Impact on Light-Weight Armour Plates. Master's thesis, Universidade de Aveiro, Portugal, 2019.

[Saleh *et al.* 2020] M. Saleh, V. Luzin, M. A. Kariem, K. Thorogood and D. Ruan. Experimental Measurements of Residual Stress in ARMOX 500T and Evaluation of the Resultant Ballistic Performance. *Journal of Dynamic Behavior of Materials*, 6:78–95, 2020.

[Sangamesh *et al.* 2018] Sangamesh, K. S. Ravishankar and S. M. Kulkarni. Ballistic Impact Study on Jute-Epoxy and Natural Rubber Sandwich Composites. *Materials Today: Proceedings*, 5(2):6916–6923, 2018.

[Smith 2009] M. Smith. ABAQUS/Standard User's Manual, Version 6.9. Dassault Systèmes Simulia Corp, United States, 2009.

[Survey 2004] U.S. Geological Survey. Types of seismic waves, 2004.

[Svensson and Tell 2015] T. Svensson and F. Tell. Stress Wave Propagation Between Different Materials. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2015.

[Systèmes 2011] Dassault Systèmes. Abaqus Scripting User's Manual, Version 6.11. 2011.

[Tasdemirci and Hall 2007] A. Tasdemirci and I.W. Hall. The effects of plastic deformation on stress wave propagation in multi-layer materials. *International Journal of Impact Engineering*, 34(11):1797 – 1813, 2007.

[Wang and Zheng 2012] K. Wang and Y. Zheng. A new particle swarm optimization algorithm for fuzzy optimization of armored vehicle scheme design. *Applied Intelligence*, 37(4):520–526, 2012.

[Yadav *et al.* 2016] R. Yadav, M. Naebe, X. Wang and B. Kandasubramanian. Body armour materials: from steel to contemporary biomimetic systems. *RSC Advances*, 6(116):115145–115174, 2016.

[Yong *et al.* 2008] M. Yong, B. G. Falzon and L. Iannucci. On the application of genetic algorithms for optimising composites against impact loading. *International Journal of Impact Engineering*, 35(11):1293–1302, 2008.

[Zahir *et al.* 2019] N. Zahir, A. Ghani, M. Daud, D. Malingam and R. Mansur. Optimisation of Hybrid Composite Reinforced Carbon and Glass Using AHP Method. *Defence S&T Technical Bulletin*, 12(1):101–112, 2019.

[Zheng-Dong *et al.* 2006] M. Zheng-Dong, W. Hui, C. Yushun, D. Rose, A. Socks and D. Ostberg. Designing an Innovative Composite Armor System for Affordable Ballistic Protection. *25th Army Science Conference*, 2006.

Appendix A

Complementary Topics

This appendix intends to complement Chapters 3, 4 and 5 with additional plots and figures.

A.1 Wave Progression Graphics

This section contain the plots with the evolution of the first and second stress peaks, measured for different tests using different impact velocities and compare the generated stresses for the three different materials: steel, aluminium and titanium. The plots in Figures A.1 to A.6 provide additional data from Section 3.7.3.

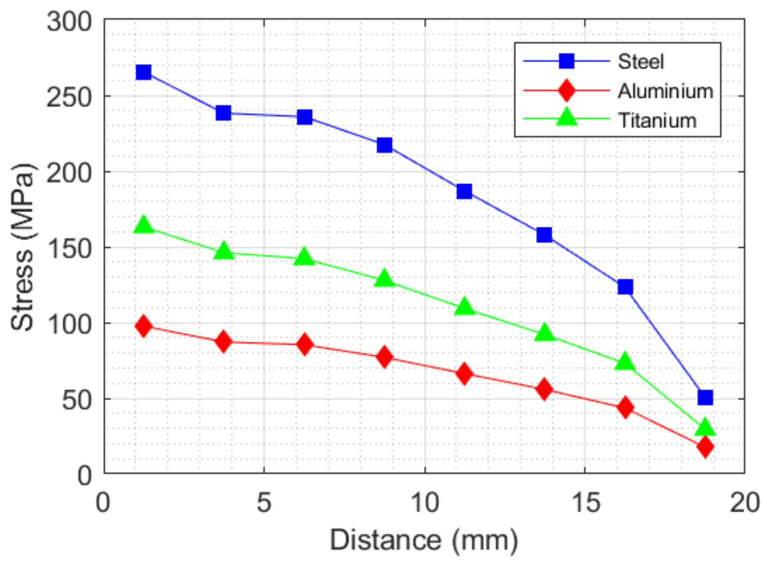


Figure A.1: First stress peak progression along the thickness of the plate, measured for an impact velocity of 5 m/s.

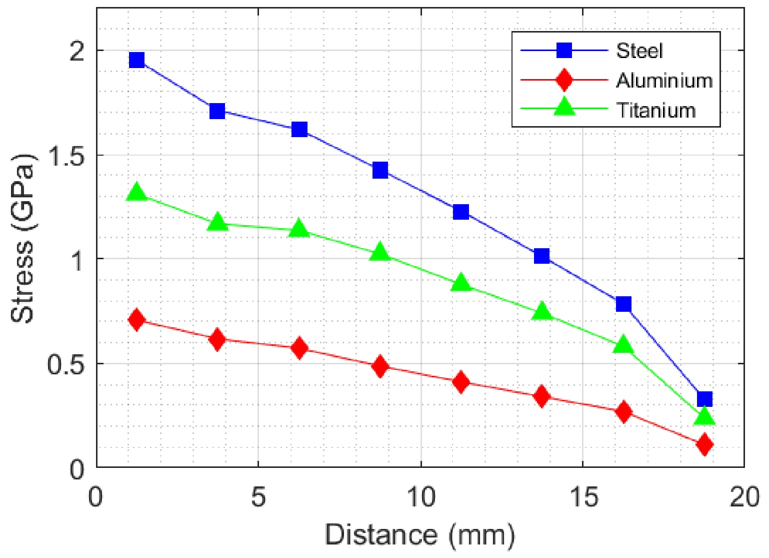


Figure A.2: First stress peak progression along thickness of the plate, measured for an impact velocity of 40 m/s.

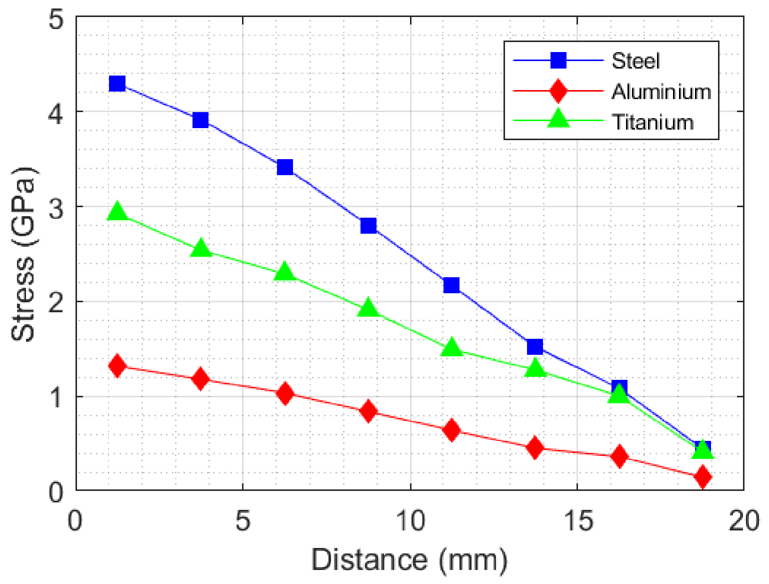


Figure A.3: First stress peak progression along thickness of the plate, measured for an impact velocity of 100 m/s and 80 m/s specifically in the case of Aluminium.

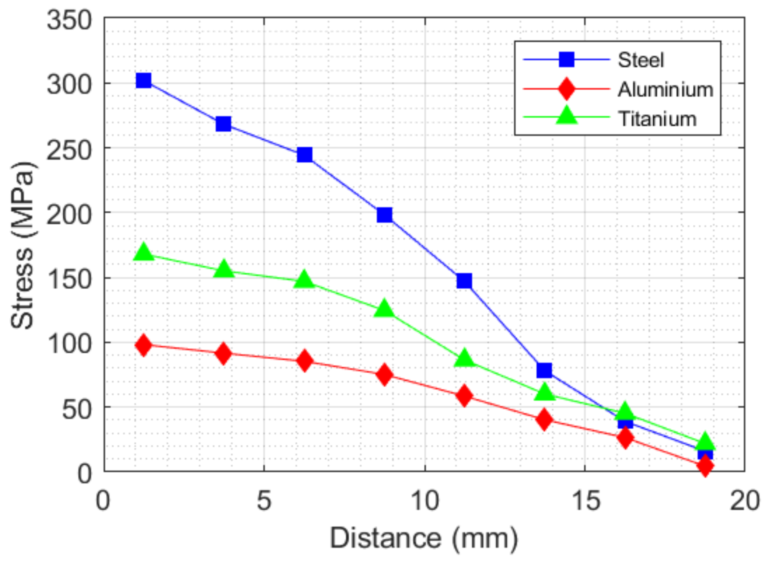


Figure A.4: Second stress peak progression along thickness measured for an impact velocity of 5 m/s.

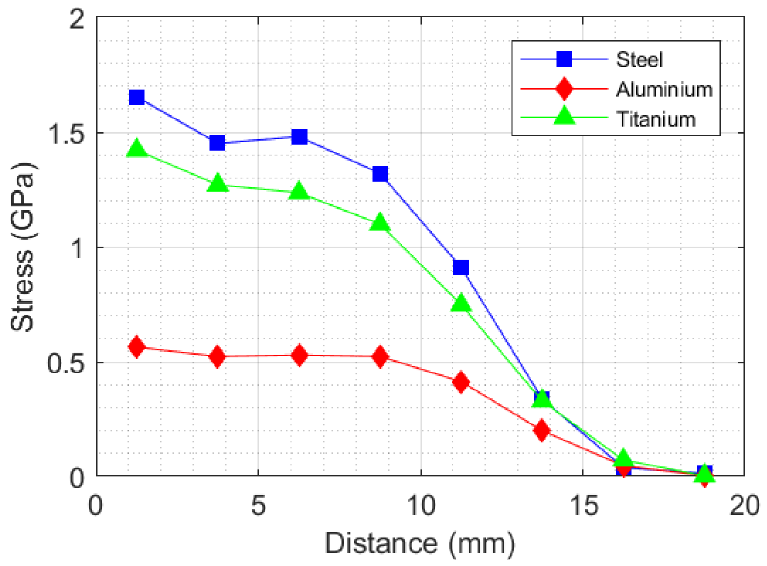


Figure A.5: Second stress peak progression along thickness measured for an impact velocity of 40 m/s.

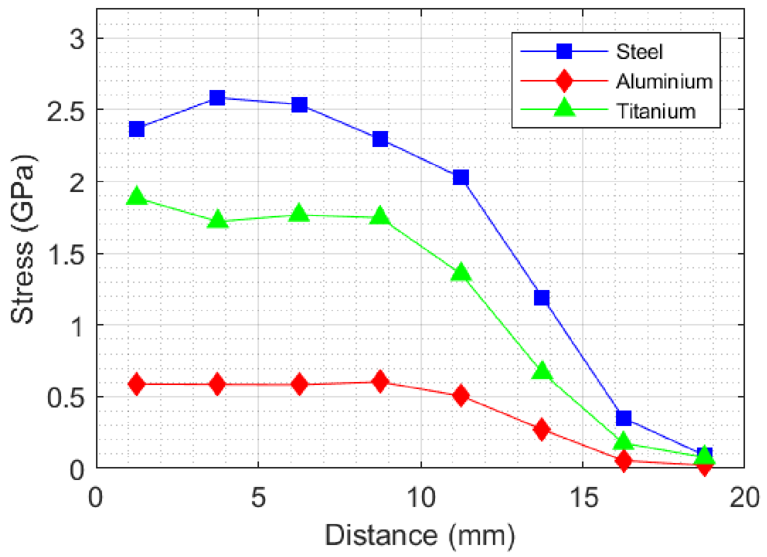


Figure A.6: Second stress peak progression along thickness measured for an impact velocity of 100 m/s and 80 m/s specifically in the case of Aluminium.

A.2 Flowchart of the Particle Swarm Optimisation algorithm

This section contains the flowchart of the Particle Swarm Optimisation algorithm implemented in the optimisation procedures of Chapters 4 and 5.

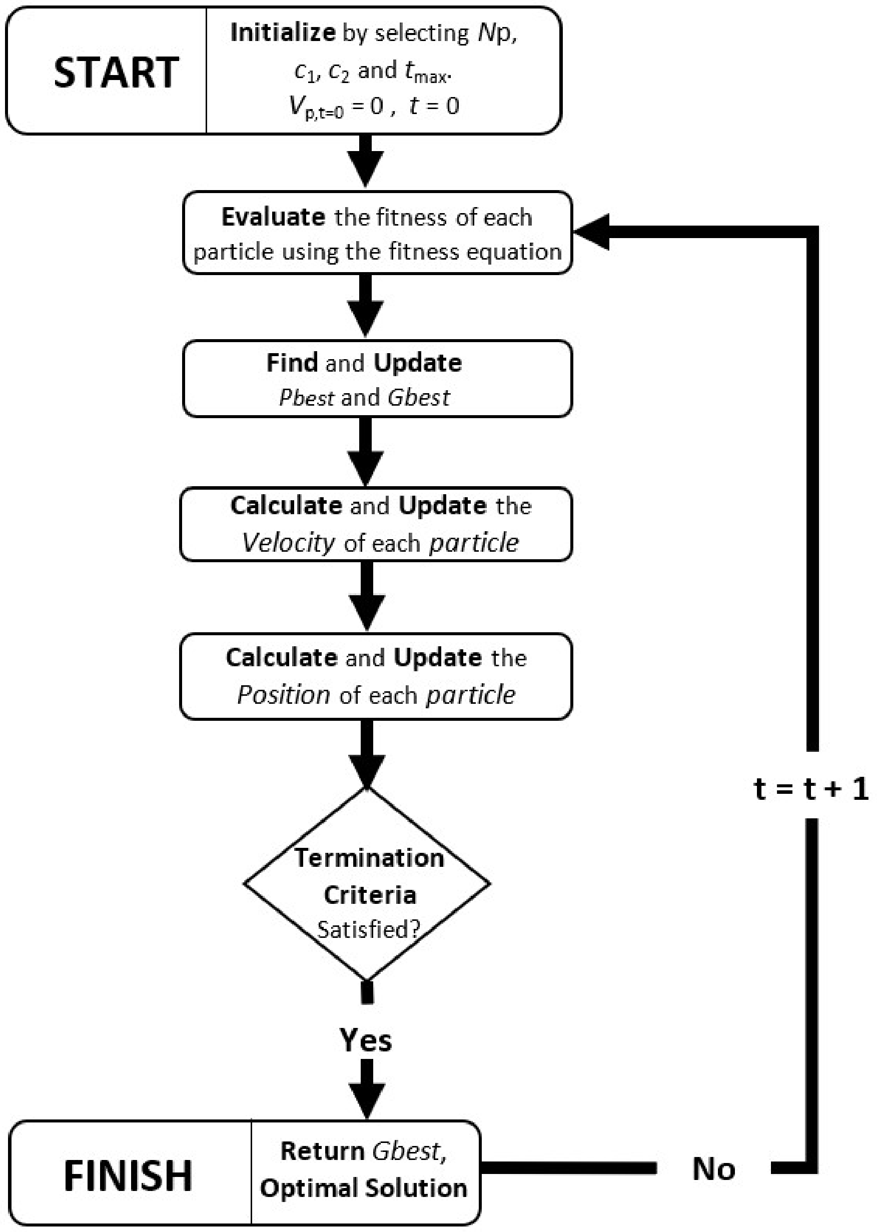

START
Initialize by selecting *N*p, c_1, c_2 and t_{max}. $V_{p,t=0} = 0$, $t = 0$
Evaluate the fitness of each particle using the fitness equation
Find and **Update** *Pbest* and *Gbest*
Calculate and **Update** the *Velocity* of each *particle*
Calculate and **Update** the *Position* of each *particle*
Termination Criteria Satisfied?
Yes
No
t = t + 1
FINISH
Return *Gbest*, **Optimal Solution**

Figure A.7: A basic flowchart of the PSO algorithm.

A.3 Additional Remarks on the Model Generation

The upcoming optimisation analyses are established through a model generation script that is flexible enough to provide a large number of possible different models in Abaqus. In the context of optimisation studies, the possibility to easily change some parameters and generate the model without the surge of unexpected problems and errors is very convenient.

To better understand the flexibility of the script to alterations for a number of models, a summary of the steps on generating a model in Abaqus can be stated as:

1. Geometry (Plate)
    - Builds a rectangular plate
2. Geometry (Projectile)
    - Builds a cylindrical projectile
3. Material Database
4. Part Section
5. Create Assembly
6. Assign Boundary Conditions
7. Create Step (Dynamic Explicit for Transient Analysis)
8. Create Interactions and Constraints
9. Create Mesh
10. Create and Submit Job

A.3.1 Geometry (Plate) Parameters

Following the ten steps above, the script provides a very useful number of parameter to quickly adjust and edit the model.

The plate geometry step has the following parameters:

- Plate Length (longest side of the plate in mm);
- Plate Width (second longest side of the plate in mm);
- Number of Plates: defines the number of plates the assembly has;
- Plate Thickness: sets the thickness for each plate.

A.3.2 Geometry (Projectile) Parameters

The projectile geometry step presents a similar structure of parameters:

- Projectile Length (in mm);
- Projectile Diameter (in mm);
- Fillet Radius (in mm);
- Projectile Mass (in kg).

It is important to note that plate partition dimensions adapt according to projectile dimensions, appropriately setting up the impact region's diameter.

A.3.3 Material Database Parameters

A database of materials is stored along matrices and vectors with the objective of having a constant list of available materials to choose from when defining the model, each one of them with its own index. The parameters are:

- Density (in kg/m^3);
- Young's Modulus (in Pa);
- Poisson's Ratio;
- Yield Stress (in Pa): A vector containing the yield stress data for the plasticity stress-strain curve of the respective material;
- Plastic Strain: A vector containing the plastic strain data for the plasticity stress-strain curve of the respective material.

A.3.4 Part Section Parameters

In this step, each plate has its own material section assignment. In the script each plate has a section parameter pointing towards the material database. The value of the parameter is the index on the material database.

A.3.5 Creating the Assembly

While there is no parameters assigned onto this step, the script is assembling all the created parts together while maintaining a distance of 0.0011 mm between the projectile and plate contact surfaces.

A.3.6 Boundary Conditions Parameters

The boundary conditions applied to the model are mostly constant, except for the velocity of the projectile, which can be controlled in the script and is defined in m/s.

A.3.7 Create Step Parameters

In this step, the conditions of analysis are set. Naturally creating a transient analysis, controlling the simulation settings is very important and may change depending on the type of study and situation itself. The parameters in this step are:

- Duration of Event (in s): Controls the time length of the analysis;
- Number of Frames: This number sets up the number of calculations during the analysis, between time steps.

A.3.8 Interactions and Constraints Parameters

The only parameter assigned to this step is the value for the Friction Penalty, which is normally equal to 0.2 .

A.3.9 Mesh Parameters

These parameters ease the meshing process while also simplifying it. It is possible to control the essential parameters of the mesh such as:

- Thickness of the elements;
- Number of Elements in each cell (from Number 1 to Number 9);
- Element Type Choice.

A.3.10 Job Parameters

A very simple parameter to setup a job submission automatically or manually as well as the name of the job in particular.

A.3.11 Parameters Overview

The combination of all the parameters described above result in a versatile script that provides plenty of possible different models and setups while also saving the user valuable time. A flowchart in Figure A.8 summarise the information detailed above.

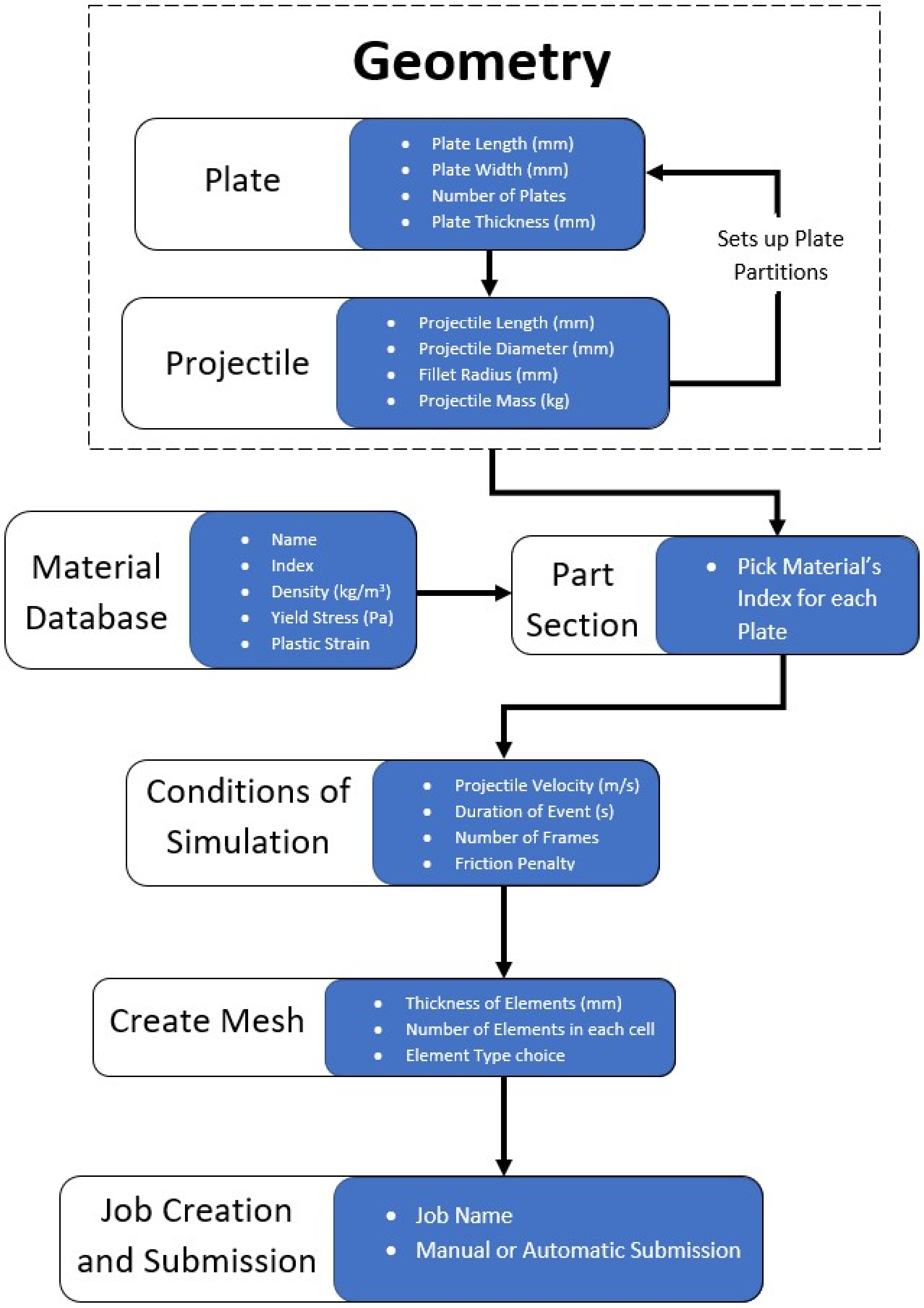


Figure A.8: Parameters in the script for effortless model editing purposes, in the form of a flowchart.

A.4 Benchmark

This section aims to provide additional plots to further understand the optimisation analyses undergoing in Chapter 4.

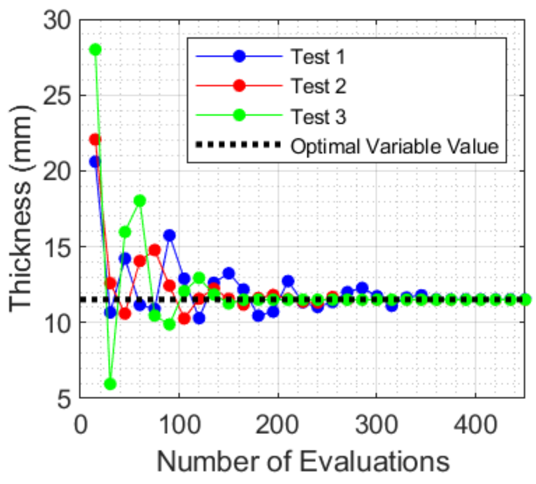


Figure A.9: The evolution of the thickness assigned for Experiment α.

A.5 Problem II

To support the discussion of results in Chapter 5, this section contains supplementary plots.

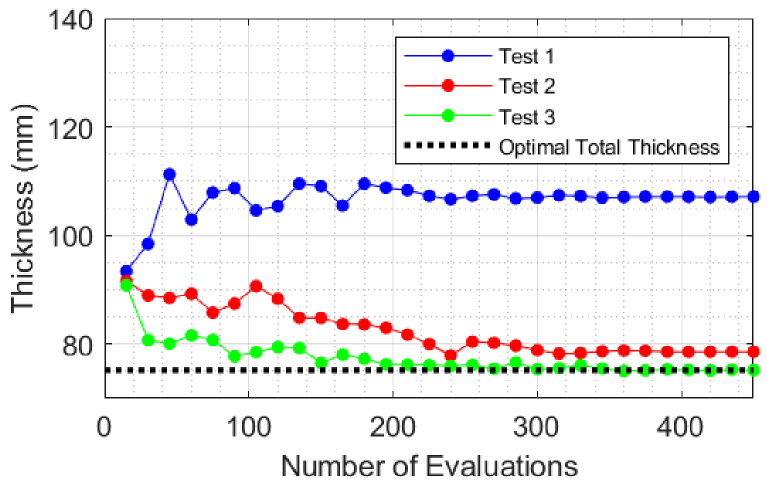


Figure A.10: Problem II - The evolution of the total thickness of the armour system for Experiment Ω.

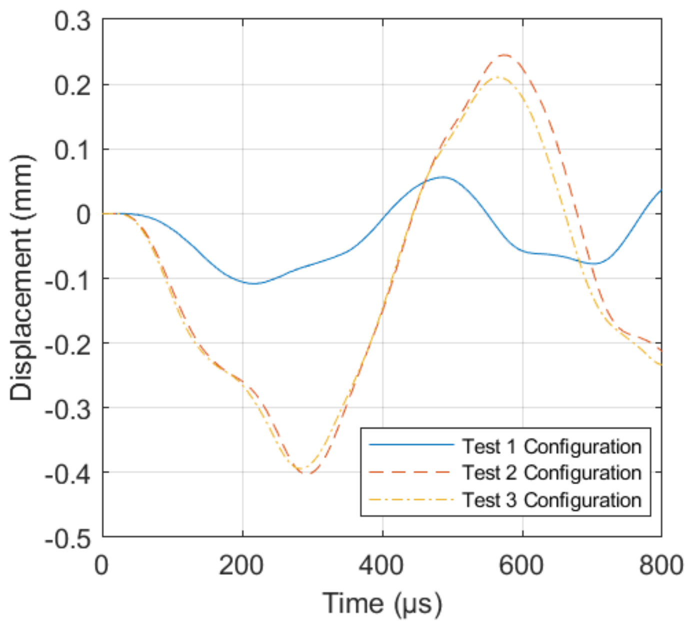


Figure A.11: Maximum vertical displacement of the centre area in the rear surface of Plate C.

Intentionally blank page.

Appendix B

Optimisation Python Code

This appendix contains the Python code used for the optimisation procedures present in Chapters 4 and 5. The script contains four essential functions, which are detailed in Section 4.3 and each one of them can be run independently.

The particular code in this appendix is used for the optimisation procedure used for the Problem II, in Section 5.2. Accordingly, the contained parameters are set to generate and process the model for Problem II. Nevertheless, all the parameters contained can be edited. To run the code, the user must launch the Abaqus CAE application and click “File”, “Run Script” and then select the script to run, which is named “PSO_Opti.py”.

```
# File Name: PSO_Opti.py -----------------------------------------
# ----------------------------------------
#
# Pedro Rocha
# pedro.miguel.rocha97@ua.pt
# Master's Dissertation
# MSc in Mechanical Engineering
# University of Aveiro
#
# ----------------------------------------

'''
Description:
This file contains a Particle Swarm Optimisation algorithm
used to optimise a 3-layer armour plate, while also containing
all the functions used for the model generation in Abaqus.
'''

# IMPORT PACKAGES
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
```

```python
from caeModules import *
from driverUtils import executeOnCaeStartup

import numpy as np
import random
import time
import string

LETTERS = string.ascii_uppercase

#-----------------------------------------
# Model Name is Rocha Ballistic Test
mdb.models.changeKey(fromName='Model-1', toName='RochaBallisticTest')

# Define Materials---------------------------

#Material Matrix (Lines -> Materials | Column 1 -> Density | Column 2 ->
    Young's Modulus | Column 3 -> Poisson's Modulus)
IsaacMat=[[7850.0, 200000000000.0, 0.33], #Steel
          [2700.0, 70000000000.0, 0.33],  #Aluminium
          [1140.0, 3000e6, 0.35],         #Nylon-6
          [960.0, 2.5e6, 0.499],          #EPDM
          [293.0, 900e6, 0.3],            #Cork
          [410.0, 103.08e6, 0.05]]        #AlFoam

RochaMat=[[7850.0, 200000000000.0, 0.29], #AISI 4340 Steel
          [2710.0, 68900000000.0, 0.33],  #Al 1100 Aluminium
          [1140.0, 3000e6, 0.35],         #Nylon-6
          [960.0, 2.5e6, 0.499],          #EPDM
          [293.0, 900e6, 0.3],            #Cork
          [410.0, 103.08e6, 0.05]]        #AlFoam

# Isaac Steel-----------------------------------------------------
mdb.models['RochaBallisticTest'].Material(description=
  'Generic Steel used by Isaac (for validation purposes)', name='Steel')
mdb.models['RochaBallisticTest'].materials['Steel'].Density(table=((
    IsaacMat[0][0], ),  ))
mdb.models['RochaBallisticTest'].materials['Steel'].Elastic(table=((
  IsaacMat[0][1], IsaacMat[0][2]), ))

# Isaac Aluminium-------------------------------------------------
mdb.models['RochaBallisticTest'].Material(description=
  'Generic Aluminium used by Isaac (for validation purposes)', name='
    Aluminium')
mdb.models['RochaBallisticTest'].materials['Aluminium'].Density(table=((
    IsaacMat[1][0], ),  ))
mdb.models['RochaBallisticTest'].materials['Aluminium'].Elastic(table=((
  IsaacMat[1][1], IsaacMat[1][2]), ))

# Isaac Nylon-6
mdb.models['RochaBallisticTest'].Material(description=
  'Generic Nylon-6 used by Isaac (for validation purposes)', name='Nylon
    -6')
mdb.models['RochaBallisticTest'].materials['Nylon-6'].Density(table=((
    IsaacMat[2][0], ),
  ))
mdb.models['RochaBallisticTest'].materials['Nylon-6'].Elastic(table=((
  IsaacMat[2][1], IsaacMat[2][2]), ))
```

```

# Isaac EPDM
mdb.models['RochaBallisticTest'].Material(description=
  'Generic EPDM used by Isaac (for validation purposes)', name='EPDM')
mdb.models['RochaBallisticTest'].materials['EPDM'].Density(table=((
    IsaacMat[3][0], ),
  ))
mdb.models['RochaBallisticTest'].materials['EPDM'].Elastic(table=((
  IsaacMat[3][1], IsaacMat[3][2]), ))

# Isaac Cork
mdb.models['RochaBallisticTest'].Material(description=
  'Generic Cork used by Isaac (for validation purposes)', name='Cork')
mdb.models['RochaBallisticTest'].materials['Cork'].Density(table=((
    IsaacMat[4][0], ),
  ))
mdb.models['RochaBallisticTest'].materials['Cork'].Elastic(table=((
   IsaacMat[4][1], IsaacMat[4][2]), ))

# Isaac AlFoam
mdb.models['RochaBallisticTest'].Material(description=
  'Generic AlFoam used by Isaac (for validation purposes)', name='AlFoam'
    )
mdb.models['RochaBallisticTest'].materials['AlFoam'].Density(table=((
    IsaacMat[5][0], ),
  ))
mdb.models['RochaBallisticTest'].materials['AlFoam'].Elastic(table=((
   IsaacMat[5][1], IsaacMat[5][2]), ))

# Rocha Steel (Plastic)
mdb.models['RochaBallisticTest'].Material(description=
  "", name='AISI 4340')
mdb.models['RochaBallisticTest'].materials['AISI 4340'].Density(table=((
    RochaMat[0][0], ), ))
mdb.models['RochaBallisticTest'].materials['AISI 4340'].Elastic(table=((
    RochaMat[0][1],
  RochaMat[0][2]), ))
mdb.models['RochaBallisticTest'].materials['AISI 4340'].Plastic(table
    =((1422388759.0, 0.0), (1506589975.0, 0.001552465), (1602274025.0,
0.003704129), (1656677772.0, 0.00554614), (1733470785.0, 0.008297903), (
1789161620.0, 0.010591797), (1869211139.0, 0.01469589), (1942047190.0,
0.020132245), (2004175089.0, 0.025095371), (2059022278.0, 0.031367939), (
2114397671.0, 0.037601408), (2139159494.0, 0.044231875), (2172926785.0,
0.04994356), (2185282025.0, 0.055613446), (2204326278.0, 0.064290472)))

# Rocha Aluminium (Plastic)
mdb.models['RochaBallisticTest'].Material(description=
  "", name='Al 1100 Aluminium')
mdb.models['RochaBallisticTest'].materials['Al 1100 Aluminium'].Density(
    table=((RochaMat[1][0], ),
  ))
mdb.models['RochaBallisticTest'].materials['Al 1100 Aluminium'].Elastic(
    table=((
  RochaMat[1][1], RochaMat[1][2]), ))
mdb.models['RochaBallisticTest'].materials['Al 1100 Aluminium'].Plastic(
    table=((
  36097600.0, 0.0), (38048800.0, 0.0014), (39512200.0, 0.003),
    (42195100.0,
  0.0059), (48048800.0, 0.0137), (52926800.0, 0.0228), (57317100.0,
    0.0333),
  (62439000.0, 0.0502), (66097600.0, 0.0661), (68048800.0, 0.0798), (
  69268300.0, 0.0897), (69268300.0, 0.0975), (68292700.0, 0.1008), (
  67317100.0, 0.1036), (64634100.0, 0.1062)))

# Rocha Nylon (Plastic)
mdb.models['RochaBallisticTest'].Material(description='',
  name='Nylon-6P')
mdb.models['RochaBallisticTest'].materials['Nylon-6P'].Density(table
    =((1200.0, ), ))
mdb.models['RochaBallisticTest'].materials['Nylon-6P'].Elastic(table
    =((2330000000.0, 0.39),
  ))
mdb.models['RochaBallisticTest'].materials['Nylon-6P'].Plastic(table
    =((13710290.98, 0.0), (
18424585.23, 0.00787), (22498356.87, 0.01685), (25288294.93, 0.02544), (
27011491.96, 0.03663), (28091776.37, 0.04669), (28316835.63, 0.05712), (
27685873.05, 0.06717), (26628692.07, 0.07871), (25999322.83, 0.09025), (
25582664.46, 0.10067), (25595012.85, 0.11222), (25607361.23, 0.12377), (
25845565.54, 0.14649), (25865482.28, 0.16511), (25881814.02, 0.18038), (
25753424.66, 0.18212), (25753424.66, 0.34433), (26301369.86, 0.51612), (
27397260.27, 0.83105), (27397260.27, 1.08869), (29041095.89, 1.3464), (
30684931.51, 1.64228), (34520547.95, 1.97643), (37808219.18, 2.26285), (
40000000.0, 2.47288)))

MaterialsEl=("Steel", "Aluminium", "Nylon-6", "EPDM", "Cork", "AlFoam")
MaterialsPl=("AISI 4340", "Al 1100 Aluminium", "Nylon-6P")

# Number of Frames---------------------------------------------------
Frames=150

#--------------------------------------------------------------------
# Selecting Parameters

# Plate Parameters (Dimentions in mm)--------------------------------

LpA=300.0 #Length
WpA=300.0 #Width

# Plate Thickness parameters exclusively for model generation
# TpA=0 #Thickness Plate A
# TpB=0 #Thickness Plate B
# TpC=0 #Thickness Plate C
# TpD=0 #Thickness Plate D
# TpE=0 #Thickness Plate E
# TpF=0 #Thickness Plate F
# TpG=0 #Thickness Plate G
# TpH=0 #Thickness Plate H

# TpALL=(TpA,TpB,TpC,TpD,TpE,TpF,TpG,TpH)

#------------------------------------------------------------------
# Projectile Parameters (Dimentions in mm)

Lproj=60.0 #Length
Dproj=20.0 #Diameter
FilletRadius=0.5 #Radius of Contact Edge Fillet
```

```
ProjMass=0.147025 #Mass (kg)


#-------------------------------------------------------------------
# Number of Plates

NPlates = 3

#-------------------------------------------------------------------
# Plate Materials
NumMatA=1 #Index Materials
NumMatB=2
NumMatC=0
NumMatD=2
NumMatE=0
NumMatF=0
NumMatG=0
NumMatH=0

PlateAmat= MaterialsPl[NumMatA] #Elastic List / Plastic List
PlateBmat= MaterialsEl[NumMatB]
PlateCmat= MaterialsPl[NumMatC]
PlateDmat= MaterialsEl[NumMatD]
PlateEmat= MaterialsPl[NumMatE]
PlateFmat= MaterialsPl[NumMatF]
PlateGmat= MaterialsPl[NumMatG]
PlateHmat= MaterialsPl[NumMatH]

AllPlateMat=(PlateAmat,PlateBmat,PlateCmat,PlateDmat,PlateEmat,PlateFmat,
    PlateGmat,PlateHmat)

# Immediately run simulation if SimulON = 1 / if SimulON = 0 manual job
    subimission is required
SimulON = 1

#--------------------------------------------------------------

def Rocha3PlateModel(thicknessA,thicknessB,thicknessC): # Model Generator
     Function (3-layer plate model), number of inputs according to number
    of plates

  # Plate Parameters (Dimentions in mm)

  LpA=300.0 #Length
  WpA=300.0 #Width

  # Plate Thickness Parameters------------------

  TpA = thicknessA #Thickness Plate A
  TpB = thicknessB #Thickness Plate B
  TpC = thicknessC #Thickness Plate C
  TpD=0            #Thickness Plate D
  TpE=0            #Thickness Plate E
  TpF=0            #Thickness Plate F
  TpG=0            #Thickness Plate G
  TpH=0            #Thickness Plate H

  #Plate Thickness Vector
```

```

TpALL=(TpA,TpB,TpC,TpD,TpE,TpF,TpG,TpH)

# Assembly Details--------------------------------------------------

DistProj=0.0011 # Distance between Projectile and Plate A in mm

#-------------------------------------------------------------------
# Conditions of Simulation

VelProj=40.0         # Velocity of the projectile in m/s
EventTime=40.0E-5    # Duration of the event analysis in s
NumFrames=Frames     # Number of frames calculated
NumHOutput=50        # Number of History Output frames calculated
FrictionPenalty=0.2 # Friction Penalty used in the Projectile-Plate
  Interaction

#--------------------------------------------------------------------
# Mesh Parameters

NElemThickness=int(round(TpA/2.5,0)+1) # Number of elements along the
  thickness of the plate
NElemCenter=5  # Number of Elements of the sides of the Center Square
  Partition
NElemD1=2    # Number of Elements along the Radius of the First
  Diameter ( First Diameter = Projectile Diameter)
NElemD2=4      # Number of Elements between the Radius of the First
  Diameter and Second Diameter ( Second Diameter = 2* Projectile
  Diameter)
NElemSQ1=10      # Number of Elements between the Radius of the Second
  Diameter and the First Large Square Partition
NElemSQ2=10      # Number of Elements between the First Large Square
  Partition and the Second One
NElemSQ3=10      # Number of Elements between the Second Large Square
  Partition and the Third One
NElemSQ4=5       # Number of Elements on the last partition in X
  direction (Y direction is influenced by the center square's number of
  elements)

NElemThicknessPlateB=int(round(TpB/2.5,0)+1)
NElemThicknessPlateC=int(round(TpC/2.5,0)+1)
NElemThicknessPlateD=int(round(TpD/2.5,0)+1)
NElemThicknessPlateE=int(round(TpE/2.5,0)+1)
NElemThicknessPlateF=int(round(TpF/2.5,0)+1)
NElemThicknessPlateG=int(round(TpG/2.5,0)+1)
NElemThicknessPlateH=int(round(TpH/2.5,0)+1)

# Element Thickness Vector---------------------------------------
NElemThicknessALL=(NElemThickness, NElemThicknessPlateB,
  NElemThicknessPlateC, NElemThicknessPlateD, NElemThicknessPlateE,
  NElemThicknessPlateF, NElemThicknessPlateG, NElemThicknessPlateH)

#Finite Element Type Choice

ElemP1=C3D8R
ElemP23=C3D8R
ElemP45=C3D8R

```

```
#-----------------------------------------------------------------
  #-------------------------------------------------------------
  #Create PlateA-------------------------------------------------

  mdb.models['RochaBallisticTest'].ConstrainedSketch(name='__profile__',
    sheetSize=0.5)
  mdb.models['RochaBallisticTest'].sketches['__profile__'].rectangle(
   point1=(0.0,
    0.0), point2=(LpA/2000, WpA/2000))
  mdb.models['RochaBallisticTest'].sketches['__profile__'].
   FixedConstraint(
    entity=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].vertices[0])
  mdb.models['RochaBallisticTest'].Part(dimensionality=THREE_D, name='
   PlateA',
    type=DEFORMABLE_BODY)
  mdb.models['RochaBallisticTest'].parts['PlateA'].BaseSolidExtrude(depth
   =TpA/1000,
    sketch=mdb.models['RochaBallisticTest'].sketches['__profile__'])
  del mdb.models['RochaBallisticTest'].sketches['__profile__']


  #----------------------------------------------------------
  #Create Partition
  #Partition Sketch

  mdb.models['RochaBallisticTest'].ConstrainedSketch(gridSpacing=0.005,
   name=
    '__profile__', sheetSize=0.250, transform=
    mdb.models['RochaBallisticTest'].parts['PlateA'].MakeSketchTransform(
    sketchPlane=mdb.models['RochaBallisticTest'].parts['PlateA'].faces
   [4],
    sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['RochaBallisticTest'].parts['PlateA'].edges
   [7],
    sketchOrientation=RIGHT, origin=(0.0, 0.0, LpA/2000)))
  mdb.models['RochaBallisticTest'].sketches['__profile__'].sketchOptions.
   setValues(
    decimalPlaces=3)
  mdb.models['RochaBallisticTest'].parts['PlateA'].
   projectReferencesOntoSketch(
    filter=COPLANAR_EDGES, sketch=
    mdb.models['RochaBallisticTest'].sketches['__profile__'])
  mdb.models['RochaBallisticTest'].sketches['__profile__'].rectangle(
   point1=(
    0.0, 0.0), point2=(0.005, 0.005))
  mdb.models['RochaBallisticTest'].parts['PlateA'].PartitionFaceBySketch(
   faces=
    mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
   getSequenceFromMask(
    ('[#10 ]', ), ), sketch=
    mdb.models['RochaBallisticTest'].sketches['__profile__'],
   sketchUpEdge=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[7])
  del mdb.models['RochaBallisticTest'].sketches['__profile__']

  mdb.models['RochaBallisticTest'].ConstrainedSketch(name='__edit__',
    objectToCopy=
    mdb.models['RochaBallisticTest'].parts['PlateA'].features['Partition
   face-1'].sketch)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   projectReferencesOntoSketch(
    filter=COPLANAR_EDGES, sketch=
    mdb.models['RochaBallisticTest'].sketches['__edit__'], upToFeature=
    mdb.models['RochaBallisticTest'].parts['PlateA'].features['Partition
   face-1'])
mdb.models['RochaBallisticTest'].sketches['__edit__'].Line(point1
   =(0.005,
    0.005), point2=(LpA/2000, WpA/2000))
mdb.models['RochaBallisticTest'].sketches['__edit__'].ArcByCenterEnds(
   center=(
    0.0, 0.0), direction=CLOCKWISE, point1=(0.0, 0.01), point2=(0.01,
   0.0))
mdb.models['RochaBallisticTest'].sketches['__edit__'].ArcByCenterEnds(
   center=(
    0.0, 0.0), direction=CLOCKWISE, point1=(0.0, 0.02), point2=(0.02,
   0.0))
mdb.models['RochaBallisticTest'].sketches['__edit__'].rectangle(point1
   =(0.0,
    0.0), point2=(LpA/4000, WpA/4000))

mdb.models['RochaBallisticTest'].parts['PlateA'].features['Partition
   face-1'].setValues(
    sketch=mdb.models['RochaBallisticTest'].sketches['__edit__'])
del mdb.models['RochaBallisticTest'].sketches['__edit__']
mdb.models['RochaBallisticTest'].parts['PlateA'].regenerate()


#-----------------------------------------------------------------
#Create Partition Cells

mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#1 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[15],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[20]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[25], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#2 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[17],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[21],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[26],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[29]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[16], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#4 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[17],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[31]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[24], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#8 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[4],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[33]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[30], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#1 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[4],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[38]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[36], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#8 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[49],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[54]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[42], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#10 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[52],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[54]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[47], sense=
   REVERSE)
mdb.models['RochaBallisticTest'].parts['PlateA'].
   PartitionCellByExtrudeEdge(
    cells=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
   getSequenceFromMask(
    ('[#20 ]', ), ), edges=(
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[54],
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[57]), line=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges[52], sense=
   REVERSE)

```

```

#-------------------------------------------------------------------
#Create Projectile

mdb.models['RochaBallisticTest'].ConstrainedSketch(name='__profile__',
    sheetSize=0.04)
mdb.models['RochaBallisticTest'].sketches['__profile__'].sketchOptions.
   setValues(
    decimalPlaces=3)
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   ConstructionLine(
    point1=(0.0, -0.02), point2=(0.0, 0.02))
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   FixedConstraint(
    entity=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].geometry[2])
mdb.models['RochaBallisticTest'].sketches['__profile__'].Line(point1
   =(0.0, 0.0)
    , point2=(Dproj/2000, 0.0))
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].geometry[3])
mdb.models['RochaBallisticTest'].sketches['__profile__'].Line(point1=(
   Dproj/2000,
    0.0), point2=(Dproj/2000, Lproj/1000))
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   VerticalConstraint(
    addUndoState=False, entity=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].geometry[4])
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].geometry[3],
    entity2=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].geometry[4])
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   ConstructionLine(
    point1=(0.0, 0.0117499999998327), point2=(0.0, -0.00779504235833883))
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   VerticalConstraint(
    addUndoState=False, entity=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].geometry[5])
mdb.models['RochaBallisticTest'].sketches['__profile__'].
   FixedConstraint(
    entity=
    mdb.models['RochaBallisticTest'].sketches['__profile__'].vertices[0])
mdb.models['RochaBallisticTest'].Part(dimensionality=THREE_D, name='
   Projectile',
    type=ANALYTIC_RIGID_SURFACE)
mdb.models['RochaBallisticTest'].parts['Projectile'].
   AnalyticRigidSurfRevolve(
    sketch=mdb.models['RochaBallisticTest'].sketches['__profile__'])
del mdb.models['RochaBallisticTest'].sketches['__profile__']
mdb.models['RochaBallisticTest'].ConstrainedSketch(name='__edit__',
    objectToCopy=
    mdb.models['RochaBallisticTest'].parts['Projectile'].features['3D
   Analytic rigid shell-1'].sketch)
```

```

# Projectile's Fillet--------------------------------------------

mdb.models['RochaBallisticTest'].parts['Projectile'].projectReferencesOntoSketch(
 filter=COPLANAR_EDGES, sketch=
 mdb.models['RochaBallisticTest'].sketches['__edit__'], upToFeature=
 mdb.models['RochaBallisticTest'].parts['Projectile'].features['3D Analytic rigid shell-1'])
mdb.models['RochaBallisticTest'].sketches['__edit__'].FilletByRadius(curve1=
 mdb.models['RochaBallisticTest'].sketches['__edit__'].geometry[3], curve2=
 mdb.models['RochaBallisticTest'].sketches['__edit__'].geometry[4],
 nearPoint1=(0.00743559375405312, 4.74941916763783e-05), nearPoint2=(
 0.0100798755884171, 0.0023565455339849), radius=FilletRadius/1000)
mdb.models['RochaBallisticTest'].parts['Projectile'].features['3D Analytic rigid shell-1'].setValues(
 sketch=mdb.models['RochaBallisticTest'].sketches['__edit__'])
del mdb.models['RochaBallisticTest'].sketches['__edit__']
mdb.models['RochaBallisticTest'].parts['Projectile'].regenerate()

#----------------------------------------------------------
# Assign Mass to projectile

mdb.models['RochaBallisticTest'].parts['Projectile'].ReferencePoint(point=
 mdb.models['RochaBallisticTest'].parts['Projectile'].InterestingPoint(
 mdb.models['RochaBallisticTest'].parts['Projectile'].edges[3], CENTER))
mdb.models['RochaBallisticTest'].parts['Projectile'].Set(name='RPProjInert',
 referencePoints=(
 mdb.models['RochaBallisticTest'].parts['Projectile'].referencePoints[2], ))
mdb.models['RochaBallisticTest'].parts['Projectile'].engineeringFeatures.PointMassInertia(
 alpha=0.0, composite=0.0, mass=ProjMass, name='ProjInertia', region=
 mdb.models['RochaBallisticTest'].parts['Projectile'].sets['RPProjInert'])

#---------------------------------------------------------------
# Assembly 1 Plate

mdb.models['RochaBallisticTest'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['RochaBallisticTest'].rootAssembly.Instance(dependent=ON, name=
 'PlateA-1', part=mdb.models['RochaBallisticTest'].parts['PlateA'])
mdb.models['RochaBallisticTest'].rootAssembly.Instance(dependent=ON, name=
 'Projectile-1', part=mdb.models['RochaBallisticTest'].parts['Projectile'])

#--------------------------------------------------------------
# Rotate Projectile
mdb.models['RochaBallisticTest'].rootAssembly.rotate(angle=90.0,
    axisDirection=
    (0.01, 0.0, 0.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('
   Projectile-1',
    ))

#-------------------------------------------------------------
# Translate Projectile

mdb.models['RochaBallisticTest'].rootAssembly.translate(instanceList=(
    'Projectile-1', ), vector=(0.0, 0.0, (TpA/1000)+(DistProj/1000)))


#-------------------------------------------------------------
# Create Step

# Boundary Conditions--------------------------------------------

mdb.models['RochaBallisticTest'].rootAssembly.Set(faces=
    mdb.models['RochaBallisticTest'].rootAssembly.instances['PlateA-1'].
   faces.getSequenceFromMask(
    ('[#4044400 #10 ]', ), ), name='SurfaceXX')
mdb.models['RochaBallisticTest'].XsymmBC(createStepName='Initial',
   localCsys=
    None, name='BC-SymmXX', region=
    mdb.models['RochaBallisticTest'].rootAssembly.sets['SurfaceXX'])
mdb.models['RochaBallisticTest'].rootAssembly.Set(faces=
    mdb.models['RochaBallisticTest'].rootAssembly.instances['PlateA-1'].
   faces.getSequenceFromMask(
    ('[#8000124 #80 ]', ), ), name='SurfaceYY')
mdb.models['RochaBallisticTest'].YsymmBC(createStepName='Initial',
   localCsys=
    None, name='BC-SymmYY', region=
    mdb.models['RochaBallisticTest'].rootAssembly.sets['SurfaceYY'])
mdb.models['RochaBallisticTest'].rootAssembly.Set(faces=
    mdb.models['RochaBallisticTest'].rootAssembly.instances['PlateA-1'].
   faces.getSequenceFromMask(
    ('[#44484492 #800012 ]', ), ), name='SurfaceRear')
mdb.models['RochaBallisticTest'].rootAssembly.Set(name='ProjRP',
    referencePoints=(
    mdb.models['RochaBallisticTest'].rootAssembly.instances['Projectile-1
   '].referencePoints[2],
    ))
mdb.models['RochaBallisticTest'].VelocityBC(amplitude=UNSET,
   createStepName=
    'Initial', distributionType=UNIFORM, fieldName='', localCsys=None,
   name=
    'BC-Velocity', region=
    mdb.models['RochaBallisticTest'].rootAssembly.sets['ProjRP'], v1=0.0,
     v2=
    0.0, v3=UNSET, vr1=0.0, vr2=0.0, vr3=0.0)
mdb.models['RochaBallisticTest'].rootAssembly.Set(name='ProjRPVel',
    referencePoints=(
    mdb.models['RochaBallisticTest'].rootAssembly.instances['Projectile-1
   '].referencePoints[2],
    ))
mdb.models['RochaBallisticTest'].Velocity(distributionType=MAGNITUDE,
   field='',
    name='PredField-VelocityProj', omega=0.0, region=
    mdb.models['RochaBallisticTest'].rootAssembly.sets['ProjRPVel'],
   velocity1=
    0.0, velocity2=0.0, velocity3=-VelProj)

#-------------------------------------------------------------
# Create Transient Step (Explicit Dynamic)

mdb.models['RochaBallisticTest'].ExplicitDynamicsStep(description=
    'Simula x segundos do evento (impacto de um projetil na placa
   encastrada)',
    name='TransientBulletImpact', previous='Initial', timePeriod=
   EventTime)
mdb.models['RochaBallisticTest'].fieldOutputRequests.changeKey(fromName
   =
    'F-Output-1', toName='Requested Field Outputs')
mdb.models['RochaBallisticTest'].fieldOutputRequests['Requested Field
   Outputs'].setValues(
    numIntervals=NumFrames, timeMarks=ON)

#-------------------------------------------------------------
# Create Interactions

mdb.models['RochaBallisticTest'].ContactProperty('IntProp-
   FrictionBetween')
mdb.models['RochaBallisticTest'].interactionProperties['IntProp-
   FrictionBetween'].TangentialBehavior(
    dependencies=0, directionality=ISOTROPIC, elasticSlipStiffness=None,
    formulation=PENALTY, fraction=0.005, maximumElasticSlip=FRACTION,
    pressureDependency=OFF, shearStressLimit=None, slipRateDependency=OFF
   ,
    table=((FrictionPenalty, ), ), temperatureDependency=OFF)
mdb.models['RochaBallisticTest'].rootAssembly.Surface(name='ProjSurf',
    side1Faces=
    mdb.models['RochaBallisticTest'].rootAssembly.instances['Projectile-1
   '].faces.getSequenceFromMask(
    ('[#3 ]', ), ))
mdb.models['RochaBallisticTest'].rootAssembly.Surface(name='
   FrontSurface',
    side1Faces=
    mdb.models['RochaBallisticTest'].rootAssembly.instances['PlateA-1'].
   faces.getSequenceFromMask(
    ('[#f0000000 #10f ]', ), ))
mdb.models['RochaBallisticTest'].SurfaceToSurfaceContactExp(
   clearanceRegion=
    None, createStepName='TransientBulletImpact', datumAxis=None,
    initialClearance=OMIT, interactionProperty='IntProp-FrictionBetween',
    master=mdb.models['RochaBallisticTest'].rootAssembly.surfaces['
   ProjSurf'],
    mechanicalConstraint=PENALTY, name='Int-BulletPlate', slave=
    mdb.models['RochaBallisticTest'].rootAssembly.surfaces['FrontSurface'
   ],
    sliding=FINITE)

# Create History Output---------------------------------------

mdb.models['RochaBallisticTest'].rootAssembly.regenerate()
mdb.models['RochaBallisticTest'].HistoryOutputRequest(createStepName=
    'TransientBulletImpact', interactions=('Int-BulletPlate', ), name=
    'RequestedAddHOutput', rebar=EXCLUDE, sectionPoints=DEFAULT,
   variables=(
    'CFNM', 'CFN1', 'CFN2', 'CFN3'))

mdb.models['RochaBallisticTest'].historyOutputRequests['
   RequestedAddHOutput'].setValues(
    numIntervals=NumHOutput)

#-------------------------------------------------------------------
# Create Mesh

# Seeding-----------------------------------------------------------

# Thickness Seeds--------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].seedEdgeByNumber(
   constraint=
    FINER, edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
   getSequenceFromMask(
    ('[#1244830a #8208c082 ]', ), ), number=NElemThickness)
mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
   getSequenceFromMask(
    ('[#1244830a #8208c082 ]', ), ), name='ThicknessSeeds')

# Cell N1-----------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].seedEdgeByNumber(
   constraint=
    FINER, edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
   getSequenceFromMask(
    ('[#0 #21760800 ]', ), ), number=NElemCenter)
mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
   getSequenceFromMask(
    ('[#0 #21760800 ]', ), ), name='SeedsCell1')

# Cell N2N3-----------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].seedEdgeByNumber(
   constraint=
    FINER, edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
   getSequenceFromMask(
    ('[#a00000 #803400 ]', ), ), number=NElemD1)
mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
   getSequenceFromMask(
    ('[#a00000 #803400 ]', ), ), name='SeedsCell23')

# Cell N4N5----------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].seedEdgeByNumber(
   constraint=
    FINER, edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
```

```
  getSequenceFromMask(
    ('[#114000 #260 ]', ), ), number=NElemD2)
mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
  getSequenceFromMask(
    ('[#114000 #260 ]', ), ), name='SeedsCell45')

# Cell N6N7------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].seedEdgeByNumber(
  constraint=
    FINER, edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
  getSequenceFromMask(
    ('[#800000d0 #9 ]', ), ), number=NElemSQ1)
mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
  getSequenceFromMask(
    ('[#800000d0 #9 ]', ), ), name='SeedsCell67')

# Cell N8N9--------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].seedEdgeByNumber(
  constraint=
    FINER, edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
  getSequenceFromMask(
    ('[#28002800 #8010000 ]', ), ), number=NElemSQ2)
mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
    mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
  getSequenceFromMask(
    ('[#28002800 #8010000 ]', ), ), name='SeedsCell89')

#-------------------------------------------------------------------
# Meshing

mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
    ('[#80 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
    ('[#20 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
    ('[#4 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
    ('[#100 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
    ('[#2 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#8 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#1 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#40 ]', ), ))
mdb.models['RochaBallisticTest'].parts['PlateA'].generateMesh(regions=
    mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#10 ]', ), ))

# Assign Element Type----------------------------------------

#Partition 1---------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].setElementType(
  elemTypes=(
   ElemType(elemCode=ElemP1, elemLibrary=EXPLICIT, secondOrderAccuracy=
  OFF,
   distortionControl=DEFAULT), ElemType(elemCode=C3D6, elemLibrary=
  EXPLICIT),
   ElemType(elemCode=C3D4, elemLibrary=EXPLICIT)), regions=(
   mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#80 ]', ), ), ))

#Partition 2 and 3---------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].setElementType(
  elemTypes=(
   ElemType(elemCode=ElemP23, elemLibrary=EXPLICIT, secondOrderAccuracy=
  OFF,
   distortionControl=DEFAULT), ElemType(elemCode=C3D6, elemLibrary=
  EXPLICIT),
   ElemType(elemCode=C3D4, elemLibrary=EXPLICIT)), regions=(
   mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#20 ]', ), ), ))

#Partition 4 and 5-------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].setElementType(
  elemTypes=(
   ElemType(elemCode=ElemP45, elemLibrary=EXPLICIT, secondOrderAccuracy=
  OFF,
   distortionControl=DEFAULT), ElemType(elemCode=C3D6, elemLibrary=
  EXPLICIT),
   ElemType(elemCode=C3D4, elemLibrary=EXPLICIT)), regions=(
   mdb.models['RochaBallisticTest'].parts['PlateA'].cells.
  getSequenceFromMask(
   ('[#4 ]', ), ), ))

#-----------------------------------------------------------
# Create Node Sets

# First Area of Impact Node Set (Nodal Area = Bullet Face Area)

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#0 #10a ]', ), ), name='FR_IR_A')

# Second Area of Impact Node Set (Nodal Area = 2 * Bullet Face Area)

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#80000000 #10e ]', ), ), name='FR_IR_B')

# Rear Impact Region A-------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#2480000 ]', ), ), name='R_IR_A')

# Rear Impact Region B-------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#2488080 ]', ), ), name='R_IR_B')

# Nodes on Impact Region along thickness-------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(edges=
   mdb.models['RochaBallisticTest'].parts['PlateA'].edges.
  getSequenceFromMask(
   ('[#448000 #2084080 ]', ), ), name='AlongThickness')

# Front Surface-------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#f0000000 #10f ]', ), ), name='Frontal_Surface')

# Rear Surface--------------------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#2488892 #200 ]', ), ), name='Rear_Surface')

#Side Surface (for Encastre)------------------------------------

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
   mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
   ('[#0 #60 ]', ), ), name='SideSurfaces')

# Symmetry XX Surface- ------------------------------------------
```

```

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
  mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
  ('[#4044400 #10 ]', ), ), name='SurfaceXX')

# Symmetry YY Surface

mdb.models['RochaBallisticTest'].parts['PlateA'].Set(faces=
  mdb.models['RochaBallisticTest'].parts['PlateA'].faces.
  getSequenceFromMask(
  ('[#8000124 #80 ]', ), ), name='SurfaceYY')


#-----------------------------------------------------------------
# Copy Plate (Creates a new plate with the same geometry but editable
  thickness

n=0
for n in range(1,NPlates):
  mdb.models['RochaBallisticTest'].Part(name='Plate' + LETTERS[n],
  objectToCopy=
      mdb.models['RochaBallisticTest'].parts['PlateA'])

  #-----------------------------------------------------------------
# Thickness of the New Plates-------------------------------------

n=0
for n in range(1,NPlates):
  mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].features
  ['Solid extrude-1'].setValues(
     depth=TpALL[n]/1000)
  mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
  regenerate()

  #-----------------------------------------------------------------
# Apply Section to Plates

n=0
for n in range(NPlates):
  mdb.models['RochaBallisticTest'].HomogeneousSolidSection(material=
  AllPlateMat[n],
    name='Plate' + LETTERS[n] + 'Section', thickness=None)

  mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].Set(
  cells=
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
  getSequenceFromMask(
    ('[#fff ]', ), ), name='FullPlate_' + LETTERS[n])
  mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
  SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].sets['
  FullPlate_' + LETTERS[n]],
    sectionName='Plate' + LETTERS[n] + 'Section', thicknessAssignment=
  FROM_SECTION)

# Create Assembly
```

```
  ---------------------------------------------------------------

# Create Instance
  --------------------------------------------------------------

# Add Plates to the assembly
  --------------------------------------------------

n=0
for n in range(1,NPlates):
  mdb.models['RochaBallisticTest'].rootAssembly.Instance(dependent=ON,
  name=
     'Plate' + LETTERS[n] + '-1', part=mdb.models['RochaBallisticTest'].
  parts['Plate' + LETTERS[n]])

#Translate Plates
  -----------------------------------------------------------

n=0
currTp=0
for n in range(1,NPlates):

  currTp = currTp + TpALL[n]   #Total Current Thickness
  mdb.models['RochaBallisticTest'].rootAssembly.translate(instanceList
  =(
     'Plate' + LETTERS[n] + '-1', ), vector=(0.0, 0.0, -currTp/1000))

# Boundary Conditions
  -----------------------------------------------------

# Encastre
  ---------------------------------------------------------------

n=0
for n in range(NPlates):
  mdb.models['RochaBallisticTest'].EncastreBC(createStepName='Initial',
     localCsys=None, name='BC-EncastreP' + LETTERS[n], region=
     mdb.models['RochaBallisticTest'].rootAssembly.instances['Plate' +
  LETTERS[n] + '-1'].sets['SideSurfaces'])

# Symmetry XX

n=0
for n in range(1,NPlates):
  mdb.models['RochaBallisticTest'].XsymmBC(createStepName='Initial',
  localCsys=
     None, name='SymmXX' + LETTERS[n], region=
     mdb.models['RochaBallisticTest'].rootAssembly.instances['Plate' +
  LETTERS[n] + '-1'].sets['SurfaceXX'])


# Symmetry YY-----------------------------------------------------------

n=0
for n in range(1,NPlates):
  mdb.models['RochaBallisticTest'].YsymmBC(createStepName='Initial',
  localCsys=
     None, name='SymmYY' + LETTERS[n], region=
        mdb.models['RochaBallisticTest'].rootAssembly.instances['Plate' +
      LETTERS[n] + '-1'].sets['SurfaceYY'])

# Tie Constraint-----------------------------------------------------

n=0
for n in range(1,NPlates):
    mdb.models['RochaBallisticTest'].rootAssembly.Surface(name='
      Tie_R_Surf' + LETTERS[n-1],
        side1Faces=
        mdb.models['RochaBallisticTest'].rootAssembly.instances['Plate' +
      LETTERS[n-1] + '-1'].faces.getSequenceFromMask(
        ('[#2488892 #200 ]', ), ))
    mdb.models['RochaBallisticTest'].rootAssembly.Surface(name='
      Tie_FR_Surf' + LETTERS[n],
        side1Faces=
        mdb.models['RochaBallisticTest'].rootAssembly.instances['Plate' +
      LETTERS[n] + '-1'].faces.getSequenceFromMask(
        ('[#f0000000 #10f ]', ), ))
    mdb.models['RochaBallisticTest'].Tie(adjust=ON, constraintEnforcement
      =
        NODE_TO_SURFACE, master=
        mdb.models['RochaBallisticTest'].rootAssembly.surfaces['Tie_R_Surf'
       + LETTERS[n-1]],
        name='TieSurfaces' + LETTERS[n-1] + LETTERS[n],
      positionToleranceMethod=COMPUTED, slave=
        mdb.models['RochaBallisticTest'].rootAssembly.surfaces['Tie_FR_Surf
      ' + LETTERS[n]],
        thickness=ON, tieRotations=ON)


# Thickness Seeds New Plates-----------------------------------------

n=0
for n in range(1,NPlates):
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
      seedEdgeByNumber(constraint=
        FINER, edges=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].edges.
      getSequenceFromMask(
        ('[#1244830a #8208c082 ]', ), ), number=NElemThicknessALL[n])
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].Set(
      edges=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].edges.
      getSequenceFromMask(
        ('[#1244830a #8208c082 ]', ), ), name='ThicknessSeeds')


# Meshing Plates-----------------------------------------------------

n=0
for n in range(1,NPlates):
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
      generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
      getSequenceFromMask(
        ('[#80 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#20 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#4 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#100 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#2 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#8 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#1 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#40 ]', ), ))
    mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].
    generateMesh(regions=
        mdb.models['RochaBallisticTest'].parts['Plate' + LETTERS[n]].cells.
    getSequenceFromMask(
        ('[#10 ]', ), ))

#----------------------------------------------------------------
# Create Job
# Print Parameters of the model

print("Process Job")
print(TpA,"Plate A Thickness")
print(TpB,"Plate B Thickness")
print(TpC,"Plate C Thickness")

mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
    description='', echoPrint=OFF, explicitPrecision=DOUBLE_PLUS_PACK,
    historyPrint=OFF, memory=70, memoryUnits=PERCENTAGE, model='
    RochaBallisticTest', modelPrint=OFF, multiprocessingMode=DEFAULT, name
    ='RochaPSO_V2', nodalOutputPrecision=FULL, numCpus=4, numDomains=4,
    parallelizationMethodExplicit=DOMAIN, queue=None, resultsFormat=ODB,
    scratch='', type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes
    =0)

```

```python
  if (SimulON == 1):

    mdb.jobs['RochaPSO_V2'].submit(consistencyChecking=OFF)
    mdb.jobs['RochaPSO_V2'].waitForCompletion()


def DataAcq(): # Data acquisition function

  #Create rear surface stress plot

  o1 = session.openOdb(name='C:/temp/RochaPSO_V2.odb')
  session.viewports['Viewport: 1'].setValues(displayedObject=o1)

  odb = session.odbs['C:/temp/RochaPSO_V2.odb']
  xyList = xyPlot.xyDataListFromField(odb=odb, outputPosition=
     INTEGRATION_POINT,
    variable=(('S', INTEGRATION_POINT, ((COMPONENT, 'S33'), )), ),
    elementSets=('PLATEC-1.R_IR_A', ))
   # Calculations

#--------------------------------
# Calculates Maximum Stress
#--------------------------------

  AvgStress=np.zeros(Frames)

  for a in range(Frames):
    TensaoTotal=0
    for b in range(len(xyList)-1):
      TensaoTotal += xyList[b][a][1]
    AvgStress[a]=TensaoTotal/(len(xyList))

  MaxStress=0

  for i in range(len(AvgStress)):
    if (AvgStress[i] <= MaxStress):
      MaxStress=AvgStress[i]

  # Print (MaxStress)
  MaxStress=round(MaxStress/pow(10,6) ,2)
  print(MaxStress)

  return MaxStress

def WeightCalc(L1,L2,tA,rhoA,tB,rhoB,tC,rhoC): # Calculates the true
     weight of the armour system

  wt = (L1/1000.0) * (L2/1000.0) * ((tA/1000.0)*rhoA + (tB/1000.0)*rhoB +
     (tC/1000.0)*rhoC)

  return wt

def PSO_calc(): # Particle Swarm Optimisation Algorithm

  nP = 15        # Number of Particles
  nIter = 30     # Number of Iterations
  Inert = 0.5    # Weight Inertia
  c1 = 1.0       # Cognitive Parameter
```

```
c2 = 2.0       # Social Parameter
rpen = 1000.0 # Penalty Parameter

Cond = 1 # Use --- Subject to --- Condition if Cond = 1

#Input (x variable)
nVar=3 # Number of variables of the objective function
xmin = np.zeros((nVar)) # Minimum Thickness
xmax = np.zeros((nVar)) # Maxinum Thickness
xmin[0] = 10.0  # Condition x min
xmax[0] = 50.0  # Condition x max
xmin[1] = 10.0  # Condition y min
xmax[1] = 50.0  # Condition y max
xmin[2] = 10.0  # Condition z min
xmax[2] = 50.0  # Condition z max
gmax = 1.0      # Subject to Condition | Maximum Stress (MPa)

CounterPb=0
CounterGb=0

a=0

# 1ST ITERATION (STARTING POINT)
  ----------------------------------------

x = np.zeros((nP,nIter,nVar)) # x,y,...(nVar) variable matrix
y = np.zeros((nP,nIter))      # y variable matrix
fxP = np.zeros((nP,nIter))    # f(x) [PENALIZED] variable matrix
fx = np.zeros((nP,nIter))     # f(x) [TRUE] variable matrix
gval = np.zeros((nP,nIter))   # Value of g(x)
Gx = np.zeros((nP,nIter))     # g(x) variable matrix
Pb=np.zeros((nP,nIter,nVar))  # Pb Personal Best matrix
Pby=np.zeros((nP,nIter))
indPb=np.zeros((nP,1))        # Index of current personnel best matrix

Gb=np.zeros((1,nIter,nVar))   # Gb Global Best matrix
Gby=np.zeros((1,nIter))
indGbp=0                      # Current Global Best Particle
indGbi=0                      # Current Global Best Iteration

xRan = np.zeros((nVar,nP))

for r in range(nVar): # Generate Initial Population
  xRan[r]=random.sample(range(int(xmin[r]),int(xmax[r])),nP)

for n in range(nP):
  curn = n+1
  print("Next particle, p. %d " %curn) # Prints which particle is
  simulated

  for b in range(nVar):
    x[n][0][b] = round(xRan[b][n],0) #Input x

  fx[n][0] = WeightCalc(LpA,WpA,x[n][0][0],RochaMat[NumMatA][0],x[n
  ][0][1],RochaMat[NumMatB][0],x[n][0][2],RochaMat[NumMatC][0]) #Output
  f(x)
  # fx[n][0] = Calc2(x[n][0][0],x[n][0][1])

  if (Cond == 0):
    fxP[n][0] = fx[n][0]

  Rocha3PlateModel(x[n][0][0],x[n][0][1],x[n][0][2]) # Initialises the simulation

  # "Subject to" test
  if (Cond == 1):
    gval[n][0] = abs(DataAcq())
    Gx[n][0] = gval[n][0]-gmax
    if (Gx[n][0]>0): #Penalty Condition
      fxP[n][0] = fx[n][0] + (pow(Gx[n][0],2)*rpen)
    else:
      fxP[n][0] = fx[n][0]

  # PERSONAL BEST
  for b in range(nVar):
    Pb[n][0][b]=x[n][0][b]
  # Pby[n][0]=y[n][0]
  indPb[n][0]=0

  #GLOBAL BEST (minimise => "<=" / maximise => ">=)
  if (fxP[n][0]<=fxP[a][0]):
    a=n
    for b in range(nVar):
      Gb[0][0][b]=x[n][0][b]

    indGbp = n
    indGbi = 0

# PSO LOOP (2nd to n iterations)-----------------------------------


r1 = np.zeros((1,nIter)) # Random factor r1
r2 = np.zeros((1,nIter)) # Random factor r2

v = np.zeros((nP,nIter,nVar)) # Velocity matrix

for j in range(1,nIter):
  curj=j+1
  print("Next iteration, it. %d " %curj) # Prints the number of the next iteration
  r1[0][j]=random.randrange(0,100)/100.0
  r2[0][j]=random.randrange(0,100)/100.0
  for m in range(nP):

    curm=m+1
    print("Next particle, p. %d " %curm) # Prints the number of the current particle
    print("Current Iteration, it. %d " %curj) # Prints the number of the current iteration
    print(gval[indGbp][indGbi],"Convergence Stress") # Prints current Global Best solution

    for b in range(nVar):
      v[m][j][b] = Inert*v[m][j-1][b] + c1*r1[0][j]*(Pb[m][j-1][b]-x[m][j-1][b])+c2*r2[0][j]*(Gb[0][j-1][b]-x[m][j-1][b]) # Velocity x
      x[m][j][b] = (x[m][j-1][b] + v[m][j][b]) # New x value
```

```python

      # Make sure the particle is within the search space

        if (x[m][j][b]<=xmin[b]): # x search space
          x[m][j][b]=xmin[b]
        if (x[m][j][b]>=xmax[b]):
          x[m][j][b]=xmax[b]

      fx[m][j] = WeightCalc(LpA,WpA,x[m][j][0],RochaMat[NumMatA][0],x[m][j][1],RochaMat[NumMatB][0],x[m][j][2],RochaMat[NumMatC][0]) #Output f(x)

      # fx[m][j] = Calc2(x[m][j][0],x[m][j][1])
      if (Cond == 0):
        fxP[m][j] = fx[m][j]

      Rocha3PlateModel(x[m][j][0],x[m][j][1],x[m][j][2]) # Initialises Simulation

      gval[m][j] = abs(DataAcq())
      Gx[m][j] = gval[m][j] - gmax

      #PENALTY CONDITION
      if ((Gx[m][j]>0) & Cond == 1):
        # fxP[m][j] = fx[m][j] + pow(Gx[m][j],2)
        fxP[m][j] = fx[m][j] + (pow(Gx[m][j],2)*rpen)
      else:
        fxP[m][j] = fx[m][j]

      #PERSONAL BEST (minimise => "<=" / maximise => ">=)
      if (fxP[m][j] <= fxP[m][indPb[m][0]]):
        for b in range(nVar):
          Pb[m][j][b] = x[m][j][b]
          indPb[m][0] = j
      else:
        for b in range(nVar):
          Pb[m][j][b] = Pb[m][j-1][b]

      #GLOBAL BEST (minimise: "<=" / maximise: ">=")
      if (fxP[m][j] <= fxP[indGbp][indGbi]):
        for b in range(nVar):
          Gb[0][j][b] = x[m][j][b]
          indGbp = m
          indGbi = j
        CounterGb = CounterGb+1
      if (CounterGb == 0):
        for b in range(nVar):
          Gb[0][j][b] = Gb[0][j-1][b]

    CounterGb=0
    cdif=0

    for k in range(nP): # Termination Criteria
      for b in range(nVar):
        dif = abs(Gb[0][j][b] - x[k][j][b])
        if (dif <= 0.2):
          cdif = cdif + 1
    if (cdif == nP*nVar):
      break

  print(Gb,"Global Best (All)") # Prints the Global Best Solution

  x1 = np.zeros((nP,nIter)) # Results Variable Thickness Plate A
  x2 = np.zeros((nP,nIter)) # Results Variable Thickness Plate B
  x3 = np.zeros((nP,nIter)) # Results Variable Thickness Plate C

  for i in range(nIter):
    for n in range(nP):
      x1[n][i] = x[n][i][0]
      x2[n][i] = x[n][i][1]
      x3[n][i] = x[n][i][2]

  # Results matrix-------------------------------

  Results = np.concatenate((x1,x2))
  Results = np.concatenate((Results,x3))
  Results = np.concatenate((Results,gval))
  Results = np.concatenate((Results,fx))
  Results = np.concatenate((Results,fxP))

  np.savetxt('ResultsWS.txt', Results) # Saves the Results Matrix in a
    text file

def Main(): # This function processes the script, it allows the user to
    process the intended functions present in this script
  start_time = time.time()

  PSO_calc() # Function to process, in this case it will process the
    optimisation procedure of Problem II, throught the PSO algorithm

  elapsed_time = time.time()-start_time
  print(time.strftime("%H:%M:%S", time.gmtime(elapsed_time)),"Time
    Duration") # Prints the elapsed time of the process

Main() # Process the script
```