K.J. Zandbergen

Material Model Validation for Flexible Targets - Bird Strike Crashworthiness

Material Model Validation for Flexible Targets

Bird Strike Crashworthiness

By

K.J. Zandbergen

To obtain the Master of Science degree in Aerospace Engineering

at the Delft University of Technology,

to be defended publicly on Tuesday, July 14, 2025, at 15:00.

Student number:	5185270	
Supervisor:	Dr.-Ing. S. G. P. Castro	TU Delft
Thesis committee:	Dr. S.R. Turteltaub	TU Delft
	Dr.ir. O.K. Bergsma	TU Delft

DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF AEROSPACE ENGINEERING
DEPARTMENT OF AEROSPACE STRUCTURES AND MATERIALS

GRADUATION COMMITTEE

Dated: 14-07-2025

Chair holder: Dr.ir. O.K. Bergsma

Committee member: Dr.-Ing. S.G.P. Castro

External committee member: Dr. S.R. Turteltaub

Preface

This thesis was written in fulfillment of the requirements for the Master's program in Aerospace Engineering at Delft University of Technology. The project initially began as an investigation into the structural implications of incorporating larger windows in the Flying-V aircraft. However, throughout the research, the focus gradually shifted - through several iterations - toward a broader study on the validation of material models for flexible targets. This topic proved particularly engaging, as it allowed me to build upon the foundational knowledge acquired earlier in the program and explore new aspects of computational and experimental mechanics.

This work is intended for readers with a background in the applied sciences and assumes familiarity with key concepts in mechanics, materials science, and numerical methods.

I would like to express my sincere gratitude to Dr. Saullo Giovani Pereira Castro, whose guidance, expertise, and critical feedback were invaluable throughout the development of this thesis. I am also grateful to my fellow student Floris de Vries for his steadfast academic and moral support during the course of this project. Finally, I would like to thank my partner, Alva Elmerson, for her continuous encouragement and patience throughout this process.

I hope that this research will serve as a useful foundation for future investigations, particularly those aimed at applying the findings to more specific configurations, such as the Flying-V aircraft.

Koen Jonathan Zandbergen
Katwijk, 20 May, 2025

Abstract

This thesis presents a methodology for constitutive material modeling of flexible targets, with a specific emphasis on finite element analysis. The Johnson-Cook constitutive model is employed in combination with a customized Bao-Wierzbicki damage model, both implemented in Abaqus/Explicit through a user-defined material subroutine (VUMAT). The primary objective is to evaluate the accuracy of the finite element simulations by comparing them to experimental tensile test data. To this end, the dogbone specimen used in an experimental test was reconstructed in Abaqus, and stress-strain as well as reaction force-strain data were extracted from a single finite element located in the necking region. These data were then compared to strain measurements obtained via Digital Image Correlation (DIC) from the physical experiments. The analysis was performed across four mesh densities to assess mesh convergence.

The results indicate a reasonable agreement between the simulated and experimental force-strain and stress-strain responses. Minor discrepancies were observed, largely attributable to uncertainties in the experimental measurements. Overall, the simulations demonstrate the ability of the implemented material and damage models to capture key aspects of the observed mechanical behavior.

This validated simulation framework provides a foundation for future investigations involving high-deformation phenomena. It is particularly relevant for impact analyses, such as bird strike scenarios in aerospace applications.

Summary

With the ever-increasing societal demand on the aviation sector and the efficiency of conventional aircraft designs approaching their theoretical limits, engineers are compelled to explore unconventional configurations, such as blended wing body (BWB) aircraft, for future commercial aviation. While these innovative designs offer significant aerodynamic and fuel efficiency advantages, they also introduce new structural challenges - particularly an increased vulnerability to bird strike incidents due to the expanded surface area and window exposure. This heightened risk underscores the importance of developing accurate and robust material models capable of reliably simulating impact conditions.

This thesis aims to establish a methodology for the development and validation of constitutive models for flexible targets, with the broader goal of supporting practical engineering applications such as bird strike simulations for blended wing body aircraft, including the Flying-V configuration. To achieve this, the study first establishes a foundational knowledge base through a literature review. This includes an overview of relevant bird strike regulations, the mechanics of soft-body impacts, Smoothed Particle Hydrodynamics (SPH) approaches for soft-body modeling, the functionality of Abaqus - particularly the explicit solver - and the fundamental principles of constitutive modeling as applied within finite element analysis.

Following the establishment of the theoretical framework, the Johnson-Cook constitutive model and a customized Bao-Wierzbicki damage model were parameterized using available experimental data. Curve-fitting techniques were employed to determine the material parameters A, B, n and C for the Johnson-Cook model as well as D_1 through D_5 for the damage model. Due to the absence of experimental datasets involving temperature-dependent stress-strain behavior, the thermal softening phenomenon was excluded from the parameterization in both models.

Building on the parameterization, the thesis then focuses on implementing the material behavior in Abaqus via a VUMAT subroutine. This begins with an analysis of the baseline code provided in the Abaqus documentation to identify components relevant to the Johnson-Cook model. Subsequently, a detailed step-by-step explanation of the custom Fortran subroutine developed for this project is presented, with each segment discussed in terms of its functional role in the simulation. Finally, the debugging process is outlined, including the tools and strategies used to validate and refine the subroutine's implementation.

Afterwards, the thesis discusses the integration of the VUMAT subroutine within Abaqus, beginning with the modeling of the experimental test specimen. This section outlines the key considerations involved in replicating the geometry and boundary conditions of the physical specimen. The simulation setup is further detailed with an explanation of how the analyses were configured and executed on a high-performance computing (HPC) system.

The results of the simulations are then presented and analyzed. Key findings indicate that the model exhibits mesh sensitivity, with a minimum element edge length of approximately 0.1 mm required to achieve convergence. The customized damage model successfully captures the moment of material failure, although a consistent offset is observed across all test velocities. Additionally, the implementation demonstrates a reasonable representation of strain rate dependency, aligning qualitatively with trends observed in experimental data.

Based on the findings of this study, several avenues for future work are recommended to further enhance the methodology and improve the reliability of the simulation results:

- **Improvement of experimental data quality**: The current dataset used for model parameterization exhibits limited fidelity, including sparse data at varying strain rates and notable scatter in fracture strain-triaxiality data. Acquiring higher-resolution experimental data with greater consistency would increase the accuracy and reliability of the parameterization process.
- **Extension to anisotropic materials**: This thesis focuses solely on isotropic ductile metals, specifically aluminum 2024-T3. To increase the model's applicability to aerospace structures, future work should consider extending the framework to account for anisotropic materials such as fiber-reinforced composites.
- **Strain-dependent formulation of parameter C**: The assumption of a constant strain rate sensitivity parameter C across all strain levels may be overly simplistic. Future studies could explore a more advanced formulation in which C varies as a function of equivalent plastic strain, potentially improving the accuracy of simulations in the post-yield regime.
- **Validation against real-world impact scenarios**: The current validation is limited to uniaxial tensile tests. Applying the developed material model to realistic impact scenarios - such as bird strike experiments - would provide valuable benchmarks and further assess the robustness of the simulation methodology.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AGL	Above Ground Level
ASK	Available Seat Kilometer
ASTM	American Society for Testing and Materials
BWB	Blended Wing Body
BW	Bao-Wierzbicki
CPU	Central Processing Unit
CFR	Code of Federal Regulations
CS-25	Certification Specifications 25
DIC	Digital Image Correlation
EASA	European Union Aviation Safety Agency
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulations
HPC	High-Performance Computing
JC	Johnson-Cook
SPH	Smoothed Particle Hydrodynamics
UMAT	User Material Subroutine
VC	Design Cruising Speed
VUMAT	Vectorized User Material Subroutine

Symbols

$\boldsymbol{Symbol}$	**Definition**	**Unit**
A	Johnson-Cook model parameter (yield stress)	Pa
B	Johnson-Cook model parameter (strain hardening)	Pa
C	Johnson-Cook model parameter (strain rate sensitivity)	–
c	Wave speed	m/s
$\boldsymbol{C}$	Elasticity matrix	Pa
$D_1 - D_5$	Damage model parameters	–
E	Young's modulus	Pa
f	Yield function	Pa
F	Force	N
$\boldsymbol{I}$	Internal force vector	N
J_2	Second invariant of deviatoric stress tensor	Pa²
K	Spring constant	N/m
L	Length	m
L_e	Element length	m
m	Johnson-Cook parameter (thermal softening)	–
$m(alt)$	Mass	kg
$\boldsymbol{M}$	Mass matrix	kg
n	Johnson-Cook strain hardening exponent	–
P	Pressure	Pa
$\boldsymbol{P}$	External force vector	N
r	Radius	m
T	Temperature	K

$\boldsymbol{Symbol}$	**Definition**	**Unit**
t	Time	s
Δt	Time increment	s
u	Displacement	m
$\boldsymbol{u}$	Nodal displacement vector	m
$\dot{\boldsymbol{u}}$	Nodal velocity vector	m/s
$\ddot{\boldsymbol{u}}$	Nodal acceleration vector	m/s²
W	Kernel function	–
α	Smoothing parameter (SPH)	m
δ	Dirac delta function	–
ϵ	Strain	–
$\boldsymbol{\epsilon}$	Strain tensor	–
ϵ_p	Plastic strain	–
$\Delta\epsilon$	Incremental strain	–
$\dot{\epsilon}$	Strain rate	1/s
η	Stress triaxiality ratio	–
λ	Plastic multiplier	–
ζ	Damping ratio	–
σ	Stress	Pa
σ_y	Yield stress	Pa
σ_{eq}	Equivalent (von Mises) stress	Pa
$\boldsymbol{\sigma}$	Stress tensor	Pa
$\boldsymbol{\omega}$	Frequency	rad/s or Hz
ν	Poisson’s ratio	–

List of Figures

List of Tables

1. Introduction

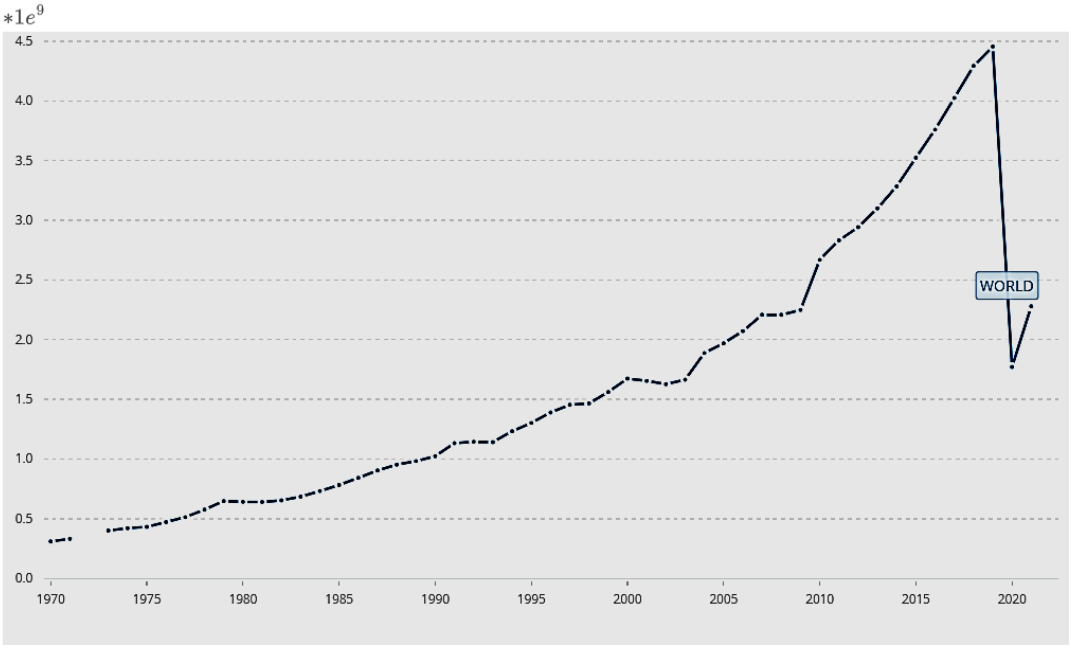
For as long as history is known, mankind has been on a continuous path to enrich its lives with inventions/tools to make their lives easier. What started with the invention of the wheel developed into the creation of the compass, the printing press, the steam engine, the utilization of electricity, and later on technological marvels such as the personal computer and the internet.

One area in particular that contributed greatly to the quality of life for the human race was the means of transportation. Where initially man was forced to go everywhere on foot, soon the usefulness of horses was discovered. And where horses fulfilled mankind's need for transportation for multiple millennia, eventually the car was invented, and not that much later also airplanes and rockets.

With every invention/discovery that was made in the field of transportation, humans were able to expand their horizons. Where once the world was much too big for a single person, nowadays almost every corner of the globe can be reached within 1 or 2 days.

1.1. Relevance of Blended Wing Body Aircraft

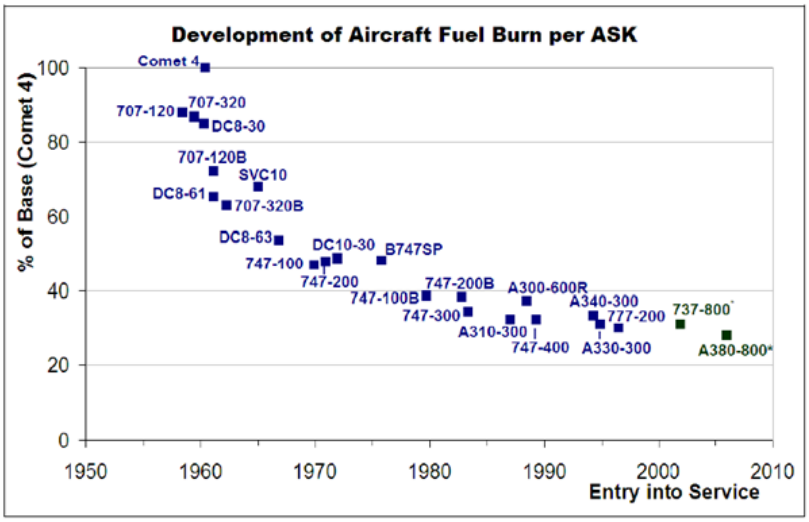
In today's interconnected world, the ability to travel between continents within a matter of days or even hours has become essential, and the airplane plays a central role in making this possible. Over the past five decades, air travel has steadily grown to become one of the most relied-upon modes of long-distance transportation. As shown in Figure 1, the number of airline passengers has consistently increased year after year - except for the COVID-19 pandemic -, reflecting the growing global demand for fast and efficient travel.



(The World Bank Group, 2025) Figure 1: Number of passengers transported by airplanes annually

With this ever-increasing trend in air travel, it is well understood that aviation contributes significantly to the release of greenhouse gases into the atmosphere. To mitigate these emissions, the industry faces a clear choice: either reduce the number of flights or develop aircraft that produce fewer emissions. However, with the global population projected to grow and stabilize around 10 billion people by 2100 (United Nations - Department of Economic and Social Affairs, 2022), the demand for air travel is unlikely to diminish. In fact, maintaining a consistent quality of life across a growing global population will likely require an even greater number of flights. As such, the only viable path forward is to design and develop aircraft that are fundamentally more sustainable and capable of meeting future demand while significantly reducing their environmental impact.

One approach to reducing aviation's environmental impact is to improve the fuel efficiency of aircraft. However, conventional aircraft designs have already undergone decades of optimization and are now approaching a practical efficiency ceiling. This plateau becomes evident when examining the trend shown in Figure 2, where gains in fuel efficiency have started to taper off in recent years.



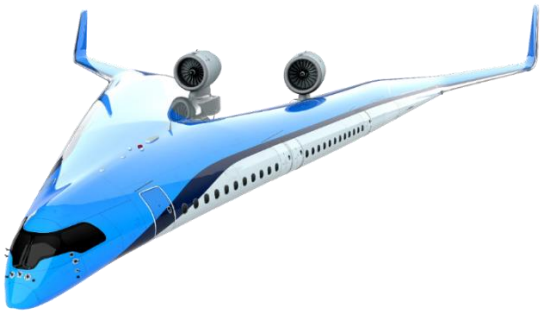
(Schaefer, 2013) Figure 2: Development of aircraft fuel efficiency

The aircraft featured in Figure 2 all represent conventional configurations, characterized by a cylindrical fuselage and two distinct wing halves. Given the limited potential for further efficiency improvements within this traditional layout, it becomes sensible to explore alternative aircraft configurations. One particularly promising concept is the blended wing body (BWB) configuration.

The BWB design integrates the fuselage and wings into a unified aerodynamic surface, effectively transforming the entire aircraft into a single, lifting structure. This integration offers several key advantages. First, it reduces the overall wetted area and virtually eliminates interference drag, which in conventional aircraft arises at the junction between the wing and fuselage. As a result, the BWB achieves a higher total lift and a significantly improved lift-to-drag ratio (Okonkwo & Smith, 2016).

Beyond aerodynamic efficiency, the BWB layout also enables a more distributed arrangement of payload and propulsion systems. Engines can be placed across the span of the wing, and payload can be spread throughout the body, reducing structural loads such as wing bending moments and shear forces. These characteristics contribute to a lower structural weight requirement and overall improved performance.

A prime example of a blended wing body aircraft is the Flying-V. This innovative concept was first developed by Justus Benad as part of his graduation project at Airbus in Hamburg, and was later significantly refined by the Aerospace Engineering faculty at TU Delft. The Flying-V embodies the core principles of the BWB configuration, integrating the passenger cabin, cargo hold, and fuel tanks into the wing structure itself. A visual representation of the Flying-V is provided in Figure 3.

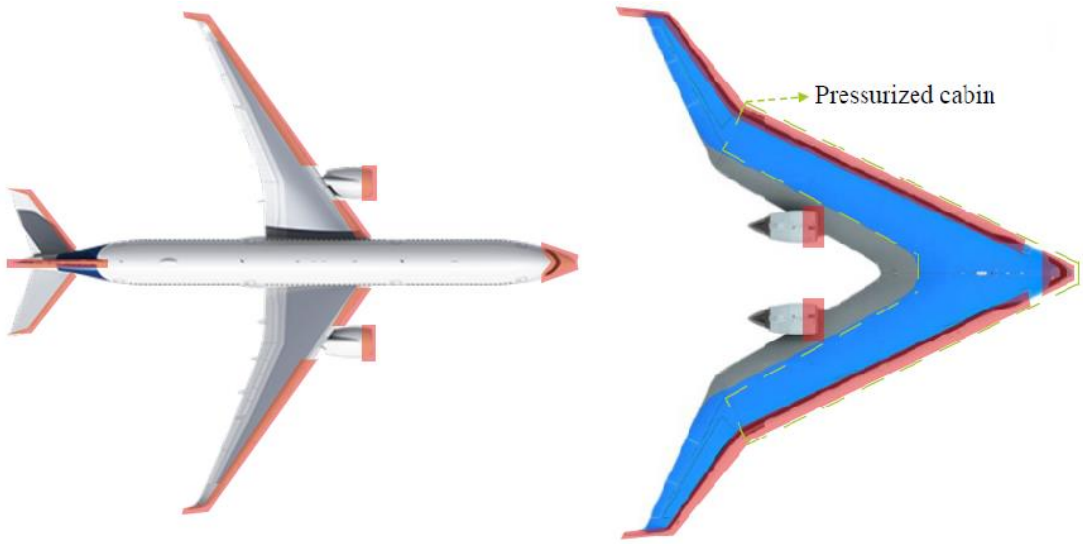


(TU Delft, n.d.) Figure 3: The Flying V

The Flying-V is designed to be comparable in size and capacity to the Airbus A350-900, featuring a similar wingspan of 65 meters, a length of 55 meters, and the ability to carry approximately 314 passengers (TU Delft, n.d.). According to Benad, the Flying-V has the potential to achieve up to a 20% reduction in fuel consumption compared to the A350. If this reduction can be realized in practice, it would represent a significant leap in efficiency - far exceeding the gradual improvements observed in recent decades, as illustrated in Figure 2.

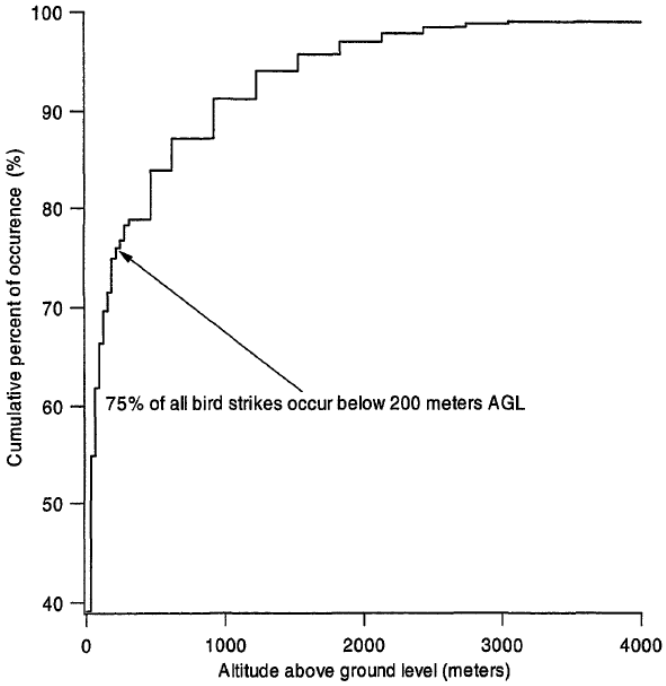
1.2. Unique Challenge of Blended Wing Body Aircraft

However, moving away from conventional aircraft configurations presents several challenges that must be addressed. One of the key concerns is the safety implications related to bird strikes. In particular, the increased exposure of structurally vulnerable sections of the airframe - specifically those surrounding passenger windows now positioned along the direct line of potential bird strike trajectories - is a concern, as illustrated in Figure 4 (Chen, van de Waerdt, & Castro, 2023). Of all reported bird strikes between 2000 and 2014, the aircraft wing accounted for the highest number of incidents. Furthermore, bird strikes involving the pilot's windshield present the greatest risk of human injury or fatality.



(Chen, van de Waerdt, & Castro, 2023) Figure 4: Aircraft regions prone to bird strike

According to data from van Es & Smit, bird strikes occur approximately once every 2,000 aircraft movements (van Es & Smit, 1999). When the cumulative distribution of bird strikes is plotted against altitude above ground level, it becomes clear that approximately 75% of these incidents take place within the first 200 meters of altitude, as can be seen within Figure 5.



(van Es & Smit, 1999) Figure 5: Cumulative occurrence of bird strike vs. altitude

While bird strikes are most common during low-altitude flight phase - typically at lower speeds - incidents at higher altitudes cannot be entirely disregarded. For example, Brotak reports the highest confirmed bird strike occurred in 1973, when a commercial aircraft collided with a Rüppell's griffon vulture at an altitude of 11.3 km (Brotak, 2018). This event, which occurred at a standard cruising altitude, underscores the fact that bird strikes can happen throughout the entire flight envelope, regardless of altitude or speed.

However, as emphasized in Jonas Bertholdt's MSc thesis *"Methodology to Identify and Quantify Flight Path Dependent Bird Strike Scenarios over Aircraft"*, bird strike probability is not solely determined by the projected frontal area of an aircraft, but more accurately by its trajectory volume - the three-dimensional space the aircraft sweeps through along its flight path (Bertholdt, 2024). This volume-based approach accounts for factors such as geometry, velocity, flight angles, and time spent at various altitudes, offering a more comprehensive assessment of exposure to bird-rich zones. Importantly, the flight trajectories of the Flying-V and the A350 are quite similar, meaning that differences in bird strike risk arise primarily from differences in aircraft geometry and exposure rather than from differences in flight paths.

When applying this method, the Flying-V does not present a universally higher bird strike risk than the A350. Despite the aforementioned concerns about the vulnerability of passenger windows and forward-facing cabin sections, the overall area exposed to bird strikes is significantly smaller in the Flying-V due to its integrated design and high wing sweep angle. In addition, critical components such as the engines are better shielded by the aircraft body than in conventional configurations. Therefore, while the Flying-V introduces specific design challenges, especially in terms of cabin protection, its reduced trajectory volume and more compact exposure geometry may yield a net safety advantage under certain operational profiles.

Based on these insights, it is reasonable to conclude that although the Flying-V may have a lower overall probability of bird strikes, designing effective bird strike protection remains especially important due to the increased vulnerability of its forward-facing windows.

1.3. Research Objective and Questions

To effectively address bird strike safety, accurate material models are essential during the design process. A reliable material model characterizes the material behavior throughout all stages of loading, taking into account potential external factors that could influence the material's response. These models ensure that the aircraft can withstand impacts from bird strikes, allowing for a more robust and safer design.

In the context of bird strikes, two distinct bodies need to be considered. The first is the bird itself, which, at high speeds, behaves like a viscous liquid. The second is the impacted region of the aircraft, which typically comprises leading-edge structures. These areas are generally more compliant than internal components such as spars or frames and thus exhibit relatively greater deformation under high-speed impact conditions.

In his thesis, Melvin Post outlines a method for creating accurate material models for birds to be used in bird strike simulations (Post, 2024). However, his work assumes the impacted area of the aircraft to be rigid. In his recommendations, Post suggests further research into the use of a flexible target material model to better replicate the real-world response of the aircraft structure during a bird strike.

To complement Melvin Post’s work, this thesis aims to develop a methodology for implementing material models for flexible targets and to establish best practices for its formulation. To guide this investigation, the following main research question is proposed:

What methodological steps are necessary to develop and validate a simulation framework for predicting the material response of flexible targets under impact loading?

To effectively address the main research question, a set of sub-research questions has been formulated. These sub-questions are designed to explore the various aspects involved in the development of a reliable tool for modeling flexible target material behavior under bird strike conditions. Together, they aim to provide a comprehensive answer to the overarching research question. The sub-research questions are as follows:

1. What existing literature is relevant to the numerical constitutive modeling of flexible targets under impact conditions?
2. Which constitutive and damage models are suitable for capturing the dynamic impact behavior of ductile metals, and how can their parameters be determined?
3. How can a constitutive model be implemented within a numerical simulation framework for dynamic impact events?
4. How can the numerical implementation of a constitutive model be validated against experimental data?

1.4. Report Outline

In this document, Chapter 2 will review the relevant literature pertinent to the research. Chapter 3 will focus on the constitutive modeling. Chapter 4 will address the parameterization of the Johnson-Cook model. Chapter 5 will provide an in-depth discussion on Abaqus subroutines and the process of VUMAT development. Chapter 6 will explore the application and verification of the model. Finally, Chapter 7 will present the conclusions and offer recommendations for future work.

2. Literature Review

This chapter presents a review and discussion of the relevant literature. It opens with a brief overview of the legislative context governing bird strike requirements, then examines the mechanics of soft-body impacts and the use of Smoothed Particle Hydrodynamics (SPH) for modeling such projectiles, and concludes with a discussion of relevant Abaqus solvers for dynamic impact simulations.

2.1. Bird Strikes - Legislation

As illustrated in Figure 4, the majority of bird strikes occur within the first kilometer of altitude; however, incidents have also been documented at significantly higher elevations. A critical factor influencing the severity of a bird strike is the mass of the bird involved. (Guinness World Records, 1936) According to the *Guinness World Records*, the heaviest flying bird is the Kori Bustard, which can reach a mass of up to 18 kg (40 lb). A collision with such a large bird has the potential to result in catastrophic damage. However, the statistical likelihood of an aircraft encountering a bird of this size during cruise conditions is extremely low. Therefore, designing for this extreme scenario would likely be overly conservative. Adopting a more balanced approach to bird strike resistance - by applying less stringent design criteria - leads to significant reductions in aircraft weight and fuel consumption without compromising overall safety.

For intercontinental flights, an aircraft will inevitably operate within the airspace jurisdictions of multiple regulatory authorities. To be granted access to these jurisdictions, the aircraft must comply with the specific requirements established by each respective authority. In the context of European airspace, the regulatory framework is governed by the European Aviation Safety Agency (EASA). EASA outlines all relevant aircraft design regulations within its Certification Specifications (CS). For large aircraft - which constitute the majority of commercial aviation traffic - Certification Specifications CS-25 is applicable. Under CS-25, EASA sets forth the following requirements:

"The aeroplane must be designed to assure capability of continued safe flight and landing of the aeroplane after impact with a 4 lb bird when the velocity of the aeroplane (relative to the bird along the aeroplane's flight path) is equal to VC at sea level or 0·85 VC at 2438 m (8000 ft), whichever is the more critical" (EASA, 2023).

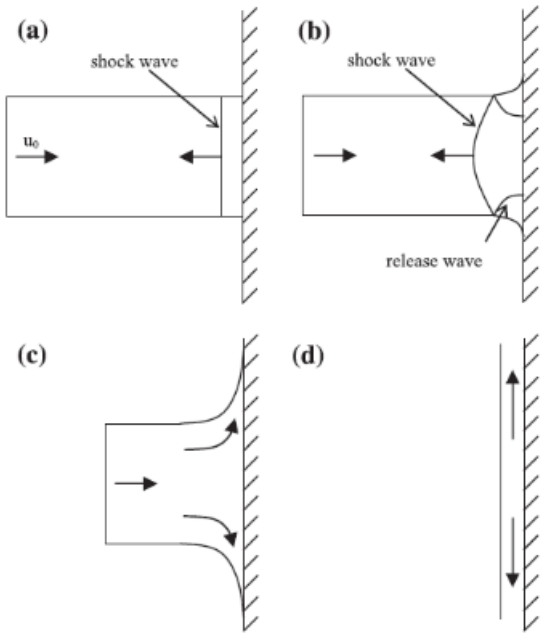
EASA's North American counterpart, the Federal Aviation Administration (FAA), specifies its regulations in a similar manner. Their regulations are encapsulated by the Code of Federal Regulations (CFR) part 25, also known as Federal Aviation Regulations (FAR). Within CFR

part 25 paragraph 25.631, the FAA specifies the following requirement regarding bird strike safety:

"The empennage structure must be designed to assure capability of continued safe flight and landing of the airplane after impact with an 8-pound bird when the velocity of the airplane (relative to the bird along the airplane's flight path) is equal to V_C at sea level, selected under § 25.335(a). Compliance with this section by provision of redundant structure and protected location of control system elements or protective devices such as splitter plates or energy absorbing material is acceptable. Where compliance is shown by analysis, tests, or both, use of data on airplanes having similar structural design is acceptable." (National Archives and Records Administration, 2025)

2.2. Soft-Body Impact Mechanics

When a soft body impacts a significantly more rigid body - such as a bird striking an aircraft panel - energy is transferred through the contact surface at the point of impact. The impact event can be divided into three distinct phases: the shock phase, the release phase, and the steady flow phase (Wilbeck, 1978). These phases characterize the progression of the impact dynamics, from initial contact to material deformation and flow. A visual representation of these phases is provided in Figure 6.



(Larsson, 2015) Figure 6: Phases of impact moment. (a) initial shock (b) pressure decay (c,d) steady flow

At time $t = 0$, the particles located at the front surface of the projectile are brought to rest relative to the impacted panel, initiating the formation of a compression shock wave. This shock wave propagates into the projectile, progressively decelerating successive layers of

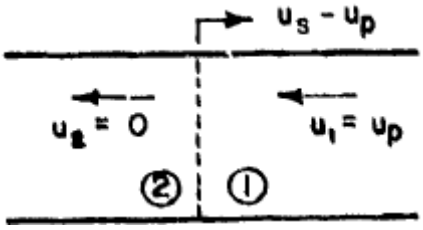
material behind the front surface. At low impact velocities, the stresses induced by the shock wave remain within the material's strength limits, allowing the projectile to respond as a solid undergoing elastic deformation. However, as the pressure generated by the shock exceeds the material strength of the projectile - typically at higher impact velocities - the soft body begins to exhibit fluid-like behavior.

Under these conditions, particles near the periphery of the projectile experience a steep strain gradient due to the combination of shock loading on one side and the presence of a free surface on the other. This differential loading causes these edge particles to accelerate radially outward, generating what are known as release waves. As these release waves reflect within the projectile, they eventually lead to the establishment of a steady flow condition. In this phase, the pressure and velocity fields within the projectile become relatively uniform, giving rise to fixed flow paths, or streamlines, that the material particles follow.

During the initial shock, the principles of continuity and preservation of momentum can be applied. For the case of impact, the conservation of mass (continuity) and momentum can be expressed by equations 1 and 2, where Figure 7 specifies the direction of the velocity components.

$$\rho_1 u_s = \rho_2 (u - u_p) \quad (1)$$

$$P_1 + \rho_1 u_s^2 = P_2 + \rho_2 (u_s - u_p) \quad (2)$$



(Wilbeck, 1978) Figure 7: Notation of velocities within the shock phase

By combining these equations, the pressure behind the shock can be expressed by equation 3. This peak pressure is also known as the Hugoniot pressure.

$$P_H = \rho_1 u_s u_p \quad (3)$$

For the impact of a cylinder on a rigid plate, where $u_p = u_0$, this becomes equation 4.

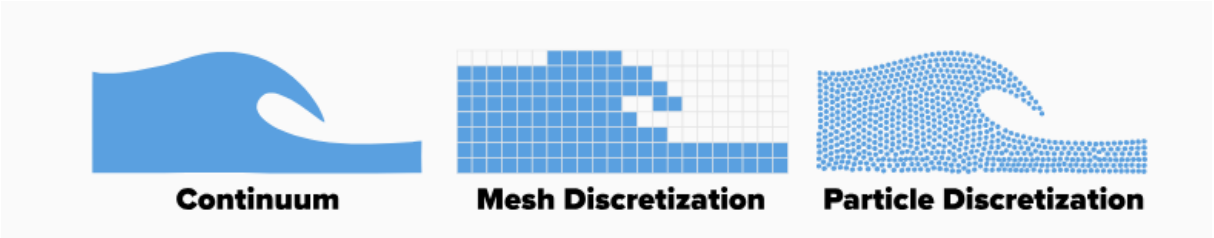
$$P_H = \rho_1 u_s u_0 \quad (4)$$

At low impact velocities, the shock velocity u_s can be approximated by the material's isentropic wave speed c_0, which results in equation 5.

$$P_H \approx \rho_1 c_0 u_0 \quad (5)$$

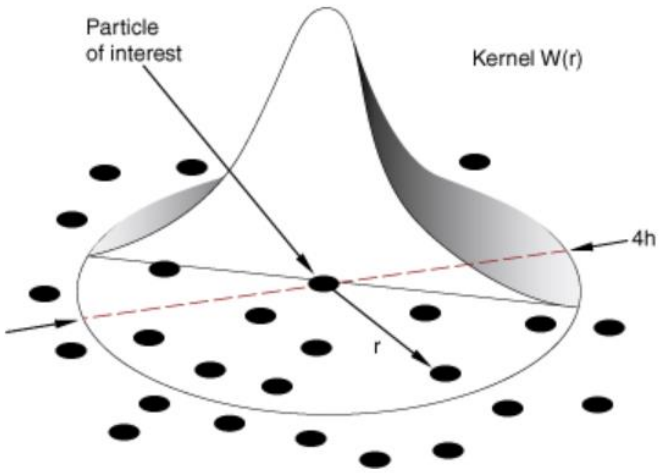
2.3. SPH Modeling of Soft-Body Projectiles

To better understand complex physical phenomena - particularly those that are difficult to capture analytically - engineers and scientists often turn to numerical modeling. In scenarios involving soft bodies that display liquid-like behavior, such as a bird during a bird strike event, the Smoothed Particle Hydrodynamics (SPH) method is frequently employed. SPH is a mesh-free, Lagrangian numerical technique well-suited for modeling problems involving fluids or highly deformable continua (Sommavilla, 2020). Unlike traditional Finite Element Methods (FEM), which solve governing equations over a fixed mesh, SPH represents the material as a collection of discrete particles. These particles interact with each other based on interpolation functions and carry physical properties such as mass, velocity, and pressure. A visual representation of this particle-based approach is shown in Figure 8.



(Sommavilla, 2020) Figure 8: Discretization of a continuum by a mesh and particle basis

As mentioned, these particles interact with one another through interpolation functions, commonly referred to as kernel functions. The kernel function defines the magnitude of influence a neighboring particle exerts on the properties of the particle of interest. This influence diminishes with increasing distance: particles that are farther away contribute less to the interpolation. Mathematically, the kernel function W depends on the distance r between particles. Figure 9 illustrates the typical shape of a kernel function, highlighting how influence decreases as r increases.



(Abaqus, n.d.) Figure 9: Kernel function curve

(Monaghan, 1992) When the SPH method is formulated mathematically, it yields a general expression represented by equation 6.

$$A(\boldsymbol{r}) = \int A(\boldsymbol{r}')W(\boldsymbol{r} - \boldsymbol{r}', h)\boldsymbol{dr}' \quad 6$$

Within this equation, A describes the physical property and W describes the aforementioned kernel function and h controls the smoothing of the kernel function (smaller h results in a steeper spike). This kernel function adheres to the properties described by equations 7 and 8.

$$\int W(\boldsymbol{r} - \boldsymbol{r}', h)\boldsymbol{dr}' = 1 \quad 7$$

$$\lim_{h \to 0} W(\boldsymbol{r} - \boldsymbol{r}', h)\boldsymbol{dr}' = \delta(\boldsymbol{r} - \boldsymbol{r}') \quad 8$$

This limit can be interpreted as the kernel function becoming a Dirac delta function when h approaches zero. For numerical applications, the equations above (6 to 8) can also be rewritten as equation 9.

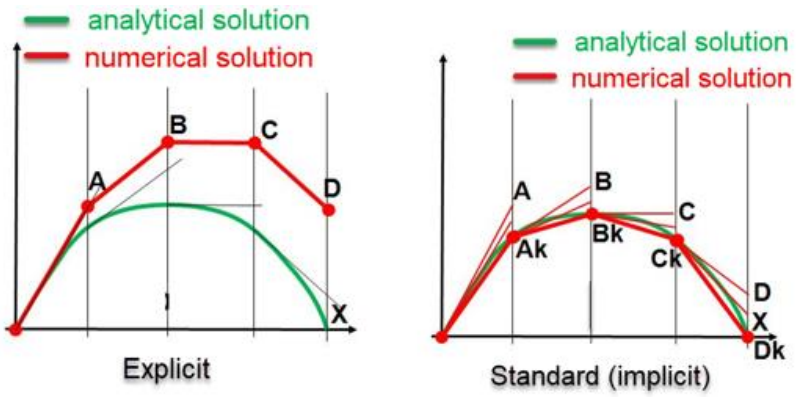
$$A_s(\boldsymbol{r}) = \sum_b m_b \frac{A_b}{\rho_b} W(\boldsymbol{r} - \boldsymbol{r_b}, h) \quad 9$$

Where parameter b denotes the particle index, and this particle has mass m_b, density ρ_b and position r_b.

2.4. Abaqus

In the numerical simulation of real-world scenarios, the use of advanced finite element software such as Abaqus provides a practical and reliable approach. Within Abaqus, users can select between two primary solution schemes: the implicit and explicit solvers, each suited to different types of analyses.

The terms *implicit* and *explicit* refer to the numerical methods employed for time integration in finite element analysis, specifically in the context of establishing dynamic equilibrium (Obbink-Huizer, 2025). The explicit solver estimates the system's state at the next time step using only information from the current state, typically through a central difference scheme. This approach is conditionally stable and particularly suited for problems involving short-duration, high-speed events. In contrast, the implicit solver involves solving a system of nonlinear equations at each time step, commonly using iterative techniques such as the Newton-Raphson method. This method is unconditionally stable and generally applied to problems requiring larger time steps or involving quasi-static loading. The difference in time-stepping behavior between the two solvers is illustrated in Figure 10.



(Obbink-Huizer, 2025) Figure 10: Implicit vs. explicit time integration

As a result of this fundamental distinction, several consequential differences arise between the two approaches. These differences are outlined below.

- Computational cost per increment: The implicit solver is computationally significantly more expensive than the explicit solver due to the iterative nature of the implicit solver, in which the Jacobian must be recalculated every iteration.
- Stable time increment: The implicit solver does not have a critical maximum time increment that affects solution stability. Unlike the explicit solver, which is conditionally stable and requires significantly smaller time steps. As a result, explicit analyses typically involve a much larger number of time increments compared to implicit analyses.
- Increment size through time: The increment size in explicit analyses can remain relatively constant throughout the simulation. In contrast, the implicit solver often requires variable increment sizes due to its sensitivity to the nonlinearity of the system being analyzed.

To determine which solver is most appropriate, the user must carefully consider the nature of the analysis to be performed. For (quasi-) static analyses or cases involving linear to mildly nonlinear behavior, the implicit solver is generally recommended. On the other hand, for highly dynamic scenarios where significant non-linearity is present, the explicit solver is typically the preferred choice. Given that bird strike modeling is the underlying focus of this paper, it is reasonable to conclude that the explicit solver is the more suitable option.

2.4.1. Abaqus Explicit Solver

(Abaqus, n.d.) To compute the kinematic conditions at a given time increment, the solution relies on the known kinematic state from the previous increment. Each time increment begins with the enforcement of dynamic equilibrium, which is mathematically represented by equation 10.

$$\boldsymbol{M\ddot{u} = P - I} \quad (10)$$

Within this equation, $\boldsymbol{M}$ is the mass matrix, $\ddot{\boldsymbol{u}}$ is the nodal acceleration vector, $\boldsymbol{P}$ is the externally applied force vector and $\boldsymbol{I}$ is the internal element force vector. In order to obtain the nodal accelerations, equation 11 can be utilized.

$$\ddot{\boldsymbol{u}}_{(\boldsymbol{t})} = \boldsymbol{M^{-1}}\,(\boldsymbol{P - I})_{(\boldsymbol{t})} \quad (11)$$

Since Abaqus consistently employs a diagonal (lumped) mass matrix, the solution of equation 11 becomes straightforward and computationally efficient. To compute the nodal velocity vector, the central difference method is typically applied. This method provides an explicit approximation of the change in nodal velocity based on current acceleration values, which is then added to the nodal velocity vector from the previous time increment. This procedure is mathematically expressed by equation 12.

$$\dot{\boldsymbol{u}}|_{(t+\frac{\Delta t}{2})} = \dot{\boldsymbol{u}}|_{(t-\frac{\Delta t}{2})} + \frac{\left(\Delta t|_{(t+\Delta t)} + \Delta t|_{(t)}\right)}{2}\ddot{\boldsymbol{u}}|_{(t)} \quad (12)$$

And by integration with respect to time, the nodal displacement vector $\boldsymbol{u}$ can be found. This integration can be approximated by equation 13.

$$\boldsymbol{u}|_{(t+\Delta t)} = \boldsymbol{u}|_{(t+\Delta t)} + \Delta t|_{(t+\Delta t)}\dot{\boldsymbol{u}}|_{(t+\frac{\Delta t}{2})} \quad (13)$$

From the nodal velocity vector, the nodal displacement vector can be derived. This displacement vector is then used to compute the element incremental strain vector over the current time increment. Once the incremental strain is known, the corresponding stresses at the element level can be calculated using user-defined constitutive equations. The resulting stress state can be represented by equation 14.

$$\boldsymbol{\sigma_{t(+\Delta t)}} = f(\boldsymbol{\sigma}_{(t)}, d\boldsymbol{\epsilon}) \quad (14)$$

2.4.2. Analysis Stability

During an Abaqus/Explicit analysis, the user must account for a phenomenon known as the stability limit, which governs the maximum allowable time increment for stable computation. Due to the explicit solver's incremental nature - where the state of the system at the next time step is approximated solely based on current values using a linear formulation - the analysis requires extremely small time increments to maintain both stability and accuracy. The maximum permissible time increment is referred to as the stability limit. Exceeding this limit can lead to numerical instability, often manifesting as non-physical, divergent, or unbounded results. Because the stability limit directly affects both the accuracy and computational cost of the simulation, the time increment must be selected with care. Ideally, it should remain just below the calculated stability limit, ensuring numerical stability while optimizing computational efficiency.

(Abaqus, n.d.) The stability limit is governed by the highest frequency that is present within the model. For situations without damping, this can be expressed by equation 15. With damping this equation gets extended to equation 16.

$$\Delta t_{\text{stable}} = \frac{2}{\omega_{max}} \tag{15}$$

$$\Delta t_{\text{stable}} = \frac{2}{\omega_{max}}\left(\sqrt{1+\xi^2}-\xi\right) \tag{16}$$

Here, ξ represents the ratio of critical damping associated with the mode possessing the highest natural frequency ω_{max}. Determining the exact value of ω_{max} is computationally expensive, as it depends on a complex interplay of various factors within the model. To avoid this, a more practical and conservative approach is adopted by evaluating the stability limit on an element-by-element basis. In this method, the highest natural frequency is estimated individually for each element, and the maximum of these local frequencies is assumed to govern the time increment for the entire analysis. This approach ensures a conservative estimate, as the highest elemental frequency will typically exceed the highest global mode frequency of the complete model. When this element-by-element approach is used, the stability limit can be redefined by equation 17.

$$\Delta t_{\text{stable}} = \frac{L_e}{C_d} \tag{17}$$

Within this equation L_e is the shortest element length and C_d is the wave speed of the material. (COMSOL, n.d.) When dealing with isotropic materials, C_d is dependent on the material properties E, ρ and ν, and can be expressed by equation 18.

$$C_d = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}} \quad (18)$$

2.4.3. Stability Influencing Factors

Several factors influence the stability limit of an explicit analysis, many of which can be adjusted by the user to optimize the trade-off between computational efficiency and solution accuracy (Abaqus, n.d.).

2.4.3.1. Material Model

One of the primary factors is the material model used. As discussed in the previous section, material properties significantly affect the dilatational wave speed, which in turn governs the critical time step. In a linear elastic regime, where material properties remain constant, the wave speed remains unchanged throughout the analysis. However, when nonlinear behavior such as plasticity is introduced, material properties may vary with deformation. For example, in models incorporating plasticity, the Young's modulus decreases beyond the yield point, leading to a corresponding reduction in wave speed. This reduction can increase the allowable time increment, thus potentially improving computational efficiency.

2.4.3.2. Mass scaling

A second factor that is influential to the stability limit is the mass scaling. Mass scaling effectively changes the material density by a certain factor, which can be user-specified or automated by Abaqus. Since the material density is linked to the material wave speed, it can directly influence the stable time increment. Mass scaling can be applied to either the whole model or specific regions, such as the elements that are expected to be critical for the stability behavior of the model. When mass scaling is applied in a fitting manner, it can drastically improve the stability limit while the effect it has on the overall dynamic behavior is insignificant.

2.4.3.3. Mesh Refinement

A third factor that influences the stability limit is mesh refinement. As indicated by equation 17, the stability limit is approximately proportional to the smallest element size in the model. Consequently, maintaining uniform and relatively large element sizes can increase the allowable time increment. However, a coarse mesh may compromise the accuracy of the simulation, particularly in regions with high gradients or complex interactions. To ensure sufficient accuracy, a refined mesh is often required, which may reduce the stability limit. This reduction can be mitigated by applying mass scaling selectively to the critical regions, thereby improving computational efficiency without significantly affecting the fidelity of the results.

2.4.3.4. Miscellaneous factors

In addition to the three primary factors discussed above, other parameter that affects either the smallest characteristic length of the mesh or the material's dilatational wave speed will also influence the stability limit. In cases where such effects are significant, they must be carefully considered and appropriately accounted for to ensure both the stability and accuracy of the analysis.

One of such secondary factors is the choice of element type, particularly with respect to the interpolation order used to approximate the displacement field (FEA tips, 2019). Abaqus offers both linear elements (first-order interpolation) and quadratic elements (second-order interpolation). Quadratic elements can capture curvature and stress gradients more accurately due to their higher-order shape functions. This generally allows for coarser meshes to achieve similar levels of accuracy compared to linear elements. This reduction in mesh density generally increases the smallest characteristic length. Since the critical time step is proportional to this length, it will result in a larger stable time increment for quadratic elements compared to linear ones.

3. Constitutive Material Models

This chapter explores the domain of constitutive material models. To fully understand the concept of a constitutive model, one must examine the relationship between stress and strain, which characterizes how a material deforms when subjected to an applied load. This relationship is typically represented by a stress-strain curve. For the purposes of this discussion, it is assumed that the material under consideration is a ductile metal exhibiting isotropic behavior.

3.1. Stress-Strain Curve

Stress-strain curves are typically derived from tensile tests, which follow the ASTM procedure designated as ASTM E8 (American Society for Testing and Materials, 2000). In this procedure, the test specimen is gradually elongated until failure occurs. During the test, the force applied to the specimen and the displacement of the test clamp are monitored. These measured parameters are subsequently used to quantify various material properties. Common material properties extracted from a tensile test include:

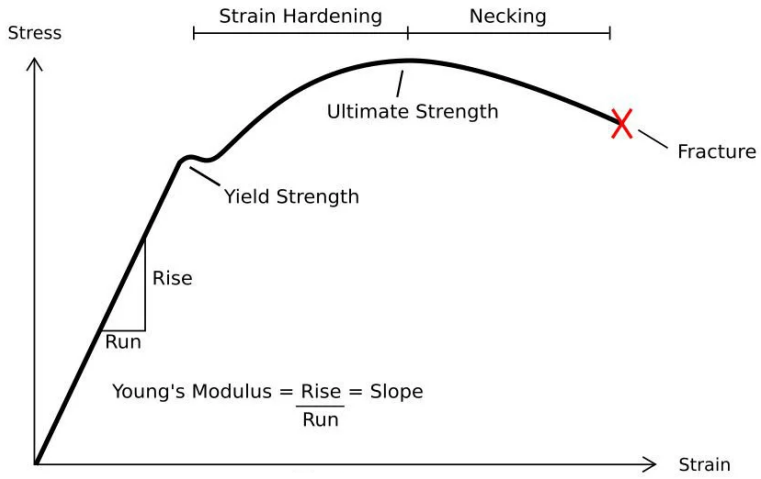
- Young's modulus and Poisson's ratio
- Yield and ultimate strength
- Ductility properties (elongation and reduction in area)
- Strain hardening characteristics

The monitored force and displacement data can be converted into engineering stress and strain by equations 19 and 20.

$$\epsilon_{\text{eng}} = \frac{L - L_0}{L_0} \qquad 19$$

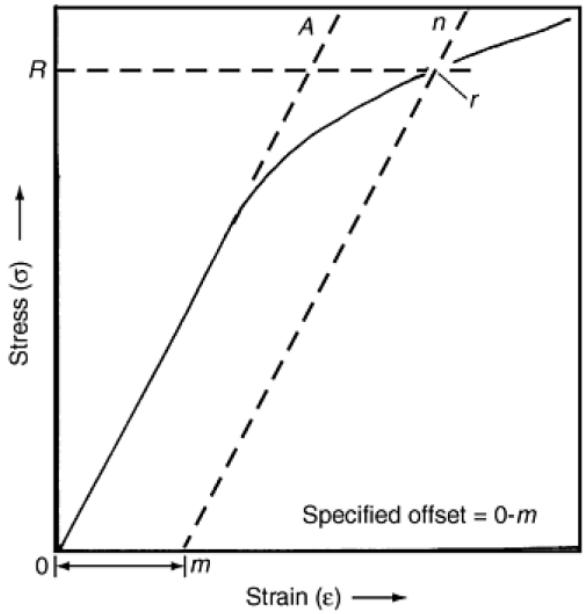
$$\sigma_{eng} = \frac{F}{A_0} \qquad 20$$

Where A and L represent the cross-sectional area and length of the specimen, respectively. Plotting stress against strain yields the typical stress-strain curve. For ductile metals, this stress-strain relationship is illustrated in Figure 11.



(Autodesk Support, 2018) Figure 11: Typical stress-strain curve

From this figure, both an elastic and a plastic region can be identified. In the elastic region, the Young's modulus remains constant until the yield point is reached. Since the precise location of the yield point is not always apparent on the stress-strain curve, an offset method is commonly employed to determine it. This method is visually illustrated in Figure 12.



(American Society for Testing and Materials, 2000) Figure 12: Offset method to determine yield point

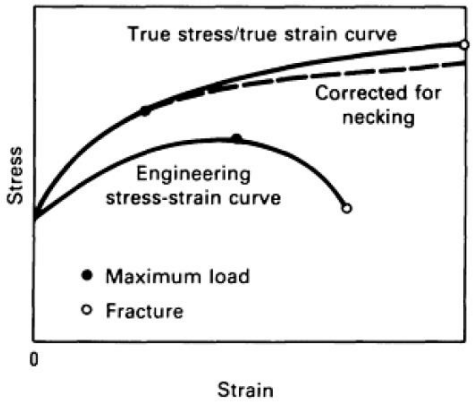
As illustrated in Figure 12, the offset method determines the yield point r as the intersection between the experimental stress-strain curve and the straight line $m - n$ which is parallel to the Young's modulus with an offset m. The yield strength is then obtained by projecting point r onto the vertical axis, where it is denoted as point R in the figure. For many materials, including ductile metals, m is typically chosen as 0.2%. However, other values can be specified when this is required.

Once the yield point is surpassed, the Young's modulus no longer remains constant and begins to vary with increasing strain. This behavior characterizes the plastic region of the stress-strain curve. The plastic region can be further divided into a strain hardening phase and a necking phase, which are distinguished by the transition of the material's stiffness from positive to negative. In this region, the dimensions of the test specimen change significantly due to plastic deformation. As a result, the assumptions underlying engineering stress and strain - expressed by equations 19 and 20 - become increasingly inaccurate as elongation progresses (Roylance, n.d.). To more accurately represent the material behavior under large deformations, true stress and true strain must be calculated using equations 21 and 22.

$$\epsilon_{true} = \int_{L_0}^{L} \frac{dL}{L} = \ln\left(\frac{L}{L_0}\right) \qquad 21$$

$$\sigma_{true} = \frac{F}{A} = \frac{F}{A_0}\frac{A_0}{A} = \frac{F}{A_0}\frac{L_0}{L} = \sigma_{eng}(1 + \epsilon_{eng}) \qquad 22$$

Figure 13 illustrates the comparison between the true stress-strain relationship and the engineering stress-strain relationship.



(American Society for Testing and Materials, 2000) Figure 13: Engineering stress-strain vs. true stress-strain

3.2. Constitutive Modeling

For analytical purposes, it is essential to derive a mathematical representation of the stress-strain data curve, which is precisely the fundamental objective of constitutive modeling. Constitutive modeling offers a solid tool that can be used for practical applications such as the simulation of a bird strike.

3.2.1. Elastic Region

Mathematical Modeling of the elastic region is rather trivial and can be described using Hooke's law. Hooke's law essentially states that under relatively small deformations, the applied force on a spring or elastic body is directly proportional to the applied deformation. This is described in equation 23.

$$F = k \cdot x \tag{23}$$

Where F is the force, k the spring constant and x is the deformation. When applied to the general definition of stress, Hooke's law can be used to find a relationship between stress and strain via equation 24.

$$\sigma = \frac{F}{A} = \frac{k \cdot x}{A} = \frac{k \cdot \epsilon \cdot L_0}{A} = \frac{\frac{EA}{L_0} \cdot \epsilon \cdot L_0}{A} = E \cdot \epsilon \tag{24}$$

As can be established from this equation, the only parameter that governs the elastic region is the Young's modulus E. Extending Hooke's law to a 3-dimensional case, equation 24 must be expanded to equation 25 (COMSOL, n.d.).

$$\boldsymbol{\sigma} = \boldsymbol{C} : \boldsymbol{\Delta\varepsilon} \tag{25}$$

Where $\boldsymbol{C}$ is the elasticity matrix, defined by equation 26.

$$\boldsymbol{C} = \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{26}$$

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{27}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{28}$$

In this case the parameters governing the elastic region are the Young's modulus E and the Poisson ratio ν.

3.2.2. Plastic Region

The mathematical modeling of the plastic region is inherently more complex due to the nonlinear relationship between stress and strain and the onset of irreversible deformation. Accurately capturing the material behavior in this regime requires the use of advanced constitutive models. In highly dynamic applications, such as impact or high-speed deformation, these models must account not only for strain but also for additional factors such as strain rate sensitivity and temperature dependence.

3.2.2.1. Johnson-Cook Material Model

A prime example of such a constitutive model is the Johnson-Cook (JC) model. The JC model includes the effects of strain rate and temperature and is widely recognized for its versatility and accuracy in describing the plastic deformation of materials under extreme loading conditions. These characteristics make the JC model an ideal choice for simulating

the complex, nonlinear behavior of materials during dynamic events. Within the JC model, the flow stress is given by equation 29 (Gordon R. Johnson, 1983).

$$\sigma_{flow} = [A + B \cdot \epsilon^{n}][\,1 + C \cdot \ln \dot{\epsilon}^{\star}]\,[1 - T^{\star m}] \quad (29)$$

$$\dot{\epsilon}^{\star} = \frac{\dot{\epsilon}}{\dot{\epsilon_0}} \quad (30)$$

$$T^{\star} = \frac{T - T_r}{T_{melt} - T_r} \quad (31)$$

Within equation 29, the parameter ϵ is the equivalent plastic strain, T^{*} is the equivalent temperature and A, B, n, C, m are material dependent constants. Where,

- $\dot{\epsilon_0}$ represents the reference strain rate
- T, T_r, T_{melt} represent the instantaneous, reference and melting temperature
- A represents the yield stress
- B and n incorporate the effects of strain hardening
- C incorporates the effect of strain rate
- m incorporates the effect of temperature

Equation 29 describes the mathematical relationship between stress and strain within the plastic region of the stress-strain curve. However, it does not account for material failure or the accumulation of damage within the material. To address this limitation, the JC model introduces an additional equation to represent damage accumulation through a cumulative damage parameter D. For each increment of plastic strain, a corresponding incremental damage value is computed using equation 32 . According to the JC model, fracture is predicted to occur once the damage parameter D reaches a value of 1 (Murugesan & Won Jung, 2019).

$$D = \sum \frac{\Delta \epsilon_p}{\epsilon_f} \quad (32)$$

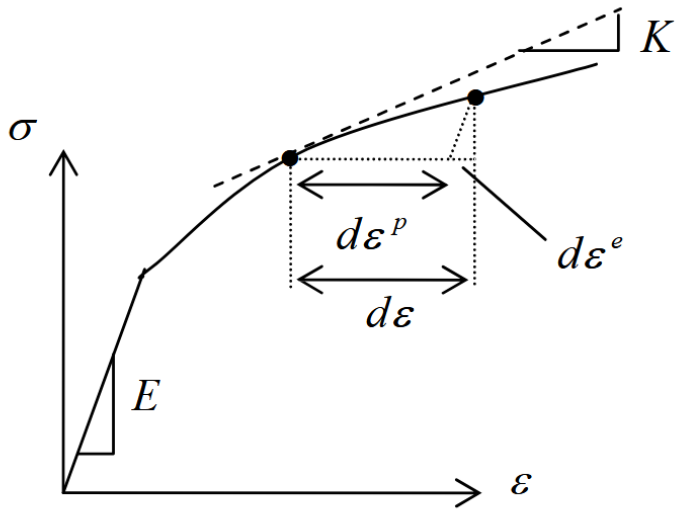
$$\epsilon_f = (D_1 + D_2 \cdot e^{D_3 \cdot \eta})\left(1 + D_4 \cdot \ln\left(\frac{\dot{\epsilon}}{\dot{\epsilon_0}}\right)\right)\left(1 + D_5\left(\frac{T - T_{room}}{T_{melt} - T_{room}}\right)\right) \quad (33)$$

$$\eta = \frac{\sigma_{hydrostatic}}{\sigma_{flow}} \quad (34)$$

Within equation 33, the failure strain for the given boundary conditions is given. Within this equation, D_1 , D_2 , D_3 , D_4 and D_5 are material-specific parameters and the parameter η is the stress triaxiality.

3.2.2.2. Associative Flow Rule

Perfect plasticity refers to a material behavior in which further deformation occurs without any increase in stress. Based on this definition, it can be reasonably concluded that isotropic materials such as Aluminum 2024-T3 do not exhibit perfectly plastic behavior. Nevertheless, the JC model formulates the material response in terms of plastic strain, thereby necessitating a procedure to distinguish and compute the proportion of elastic and plastic strain during deformation. When examining Figure 11, particularly the plastic region of the stress-strain curve in greater detail, the distribution of total strain into its elastic and plastic components can be illustrated as shown in Figure 14.



(Kelly, 2015) Figure 14: Distribution of elastic vs. plastic strain

Since the stress state described by the JC model in equation 29 is dependent on the total equivalent plastic strain, the decomposition between elastic and plastic strain components cannot be explicitly determined. Instead, this partitioning must be resolved iteratively. This iterative process is governed by an associative flow rule, which is mathematically expressed in equation 35 (Kelly, 2018).

$$d\boldsymbol{\epsilon^p} = d\lambda \cdot \frac{\partial f}{\partial \boldsymbol{\sigma}} \qquad 35$$

Here, f represents the yield function, defined by equation 36, and $d\lambda$ is the plastic multiplier, which is determined iteratively by enforcing the consistency condition from Equation 37. This condition ensures that the stress state remains on the yield surface. which requires the stress state to be on the yield surface. The iterative solution procedure is described in the next section, which covers the radial return method.

$$f = \sigma_{eq} - \sigma_y \qquad 36$$

$$f = 0 \quad (37)$$

In equation 36, the equivalent stress is typically defined as the von Mises equivalent stress, expressed as $\sigma_{eq} = \sqrt{3J_2}$, where J_2 is the second invariant of the deviatoric stress tensor. The yield stress σ_y corresponds to the flow stress as defined by the constitutive model, such as the JC model. Substituting these definitions into equation 36 yields equation 38, which defines the plastic strain rate, and equation 39, which expresses the incremental plastic strain (Kouhia, 2019).

$$d\boldsymbol{\epsilon^p} = d\lambda \cdot \frac{\partial f}{\partial J_2}\frac{\partial J_2}{\partial \boldsymbol{\sigma}} = d\lambda \cdot \frac{3}{2\sqrt{3J_2}}\boldsymbol{\sigma_{dev}} = d\lambda \cdot \frac{3 \cdot \boldsymbol{\sigma_{dev}}}{2 \cdot \sigma_{eq}} \quad (38)$$

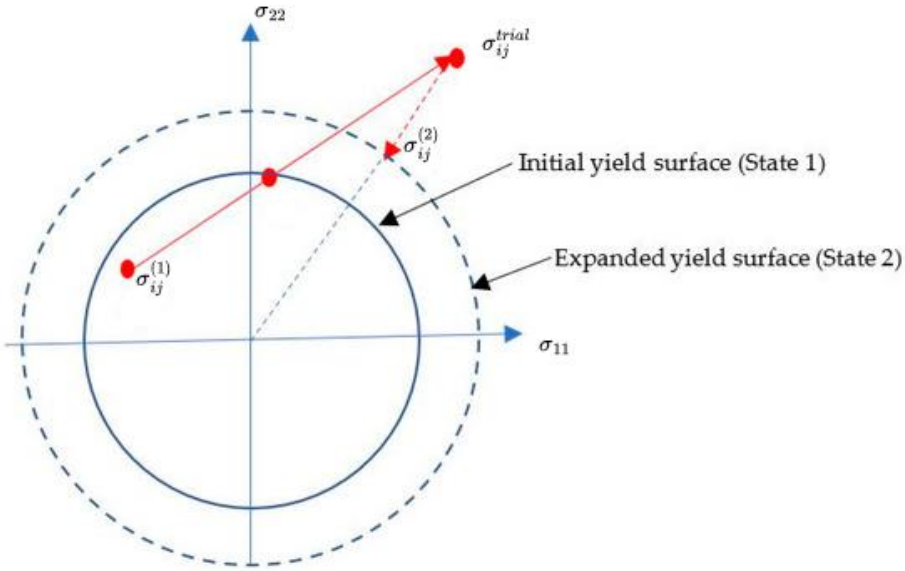
$$d\boldsymbol{\epsilon^p} = d\lambda \cdot \frac{3 \cdot \boldsymbol{\sigma_{dev}}}{2 \cdot \sigma_{eq}} \quad (39)$$

It should be noted that the term $\frac{3 \cdot \boldsymbol{\sigma_{dev}}}{2 \cdot \sigma_{eq}}$ is a unit vector normal to the yield surface.

3.2.2.3. *Radial Return Mapping*

When applying the associative flow rule within the context of incremental plasticity, it is necessary to determine a value for $d\lambda$. One effective approach for this is the radial return method. This method operates under the assumption that, at the beginning of each increment, the entire incremental strain is elastic. Based on this assumption, a trial stress state is computed. The trial stress is then evaluated to determine whether it lies inside, exactly on, or outside the yield surface, which is governed by the chosen constitutive model. In the cases where the trial stress state stays within or is exactly on the yield surface, the incremental strain must be assumed as elastic (LS-DYNA, 2005).

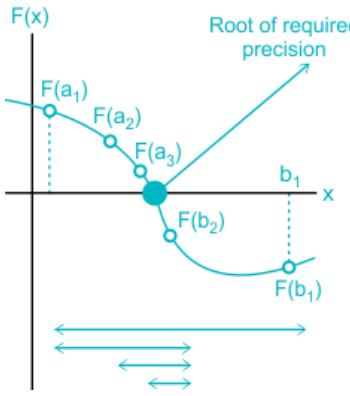
In situations where the trial stress state lies outside the yield surface, the radial return method is employed to enforce the consistency condition, which requires that equation 37 be satisfied. This method ensures that the final stress state at the end of the current increment remains on the yield surface by projecting the trial stress state back onto it. As a result, the actual stress state corresponds to this projected point on the yield surface. This concept is visually represented in Figure 15.



(Muiruri, Maringa, & du Preez, 2022) Figure 15: Radial return method

As discussed in the preceding section, the yield surface is defined by the governing constitutive model. For the JC model, it is evident that the yield surface is not fixed in stress space; instead, it expands with increasing total equivalent plastic strain. This behavior introduces an implicit dependency that necessitates the use of iterative techniques to compute the projected stress state. One such technique commonly employed is the bisection method.

The bisection method is fundamentally a root-finding technique. It operates by evaluating a function at the midpoint between two bounds and then updating these bounds based on the sign of the function value at the midpoint. This process is repeated iteratively until the root is located within a desired tolerance. A visual representation of the bisection scheme is provided in Figure 16.



(Testbook, 2024) Figure 16: Bisection root-finding method

In the context of the radial return method, the function defined in equation 36 serves as the basis for the bisection scheme. To apply this method in determining the true stress state, it is first necessary to establish the bounds within which the incremental plastic strain must lie.

The lower bound is trivial, as the minimum possible incremental plastic strain is zero. For the upper bound, it is reasonable to assume that the maximum incremental plastic strain equals the total incremental strain. From Hooke's law, the deviatoric stress-strain relationship can be expressed as shown in equation 40. Assuming the material behaves as a perfectly plastic solid (no elastic deformation), the total deviatoric strain increment is entirely plastic, such that $\boldsymbol{\epsilon_{ij}^{dev} = \epsilon_{ij}^{p}}$. Based on this fact and the previously assumed trial stress, the upper bound for the incremental plastic strain can be derived as shown in equation 41 (McGinty, 2012).

$$\boldsymbol{\sigma_{ij}^{dev} = 2G \cdot \epsilon_{ij}^{dev}} \quad 40$$

$$\Delta\epsilon_{ij}^{p} = \frac{\sigma_{tr_{eq}}}{2G} \quad 41$$

Figure 17 presents a visualization of the possible outcomes of the bisection scheme when applied within the radial return method during a single iteration.

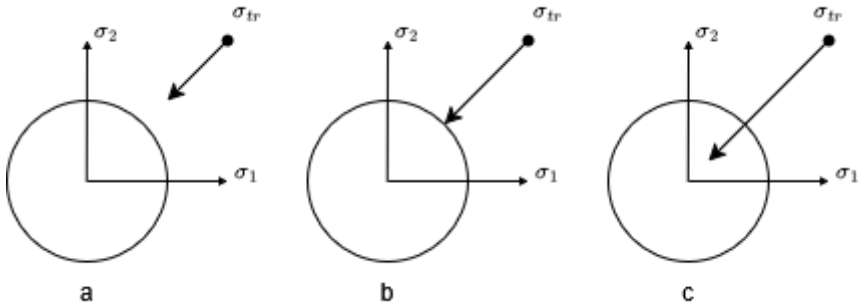


Figure 17: Possible outcomes for each bisection iteration

The possible outcomes within this figure can be described as follows:

- Subfigure "a" shows a weak plastic strain increment. The lower bound is updated (increased), the midpoint is reassessed and the process is repeated until convergence is achieved.
- Subfigure "b" shows a plastic strain increment that falls within the predefined tolerance. The bisection scheme has converged.
- Subfigure "c" shows a too strong plastic strain increment. The upper bound is updated (decreased), the midpoint is reassessed and the process is repeated until convergence is achieved.

4. Methodology - JC Model Parameterization

The value of the JC model hinges critically on the accurate determination of its material parameters. These parameters - representing mechanical properties such as yield strength, strain hardening coefficients, strain rate sensitivity, and thermal degradation - must be calibrated to reflect the specific characteristics of a given material. Without precise parameterization, the predictive capability of the JC model weakens, which leads to significant inconsistencies between simulated and experimental results.

This chapter investigates the methodology employed to extract the JC material parameters from experimental data. This methodology involves analyzing datasets generated from a series of controlled mechanical tests designed to capture the influence of strain hardening, strain rate sensitivity, and thermal softening on material behavior. To give meaning to this analysis, Python scripts that process the datasets have been written. The procedures employed within these Python scripts are elaborated on within sections 4.1 to 4.6.

4.1. Determination of A

As discussed in Section 3.2.2, the parameter A represents the yield stress of the material. There are two approaches to assigning a value to A. The first involves referencing reliable online sources. For instance, in the case of Aluminum 2024-T3, the yield stress is well-documented, making it straightforward to obtain a credible value. (American Society for Testing and Materials, 1990) One such source reports the yield stress for this specific aluminum alloy as 345 MPa.

The second approach is to derive the yield stress directly from tensile test data, as illustrated in Figure 12. The dataset analyzed in this section is sourced from the paper titled *"Effects of Strain Rate on Stress-Strain Curves in 2024 Aluminum Alloy After Solution Heat Treatment"* by Masahiro Nishida et al. (Masahiro , Satoshi , Ziyi , Masaki , & Masanobu , 2023). This paper documents a series of tensile tests conducted at various strain rates, providing valuable data for determining a value for A. The tests performed in the study are visualized in Figure 18.

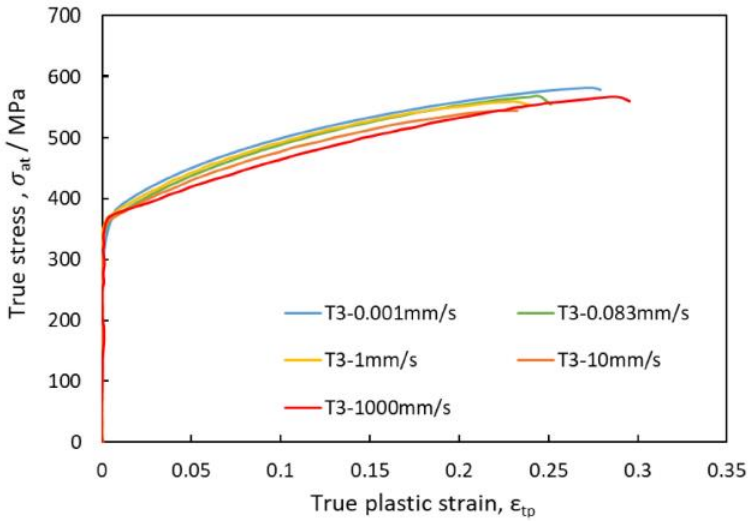


Figure 18: Tensile stress-strain test dataset for aluminum 2024-T3

To extract quantitative data from the figure, the curves were digitized, and the resulting datasets are provided in Appendix A. It is important to note that these datasets exhibit minor inconsistencies. In particular, certain data points possess identical strain values (x-coordinates) but differ in their corresponding stress values. These discrepancies are attributed to resolution limitations of the digitization software and potential manual inaccuracies during the digitization process. Since the 5 curves all lie on top of each other within the elastic region, the blue curve will be treated as a reference test from which A is derived. To determine A, the 0.2% offset method is applied using the Python script provided in Appendices C.1 and C.2. Running this script generates Figure 19, from which the yield stress $A = 365.3\ MPa$ is found.

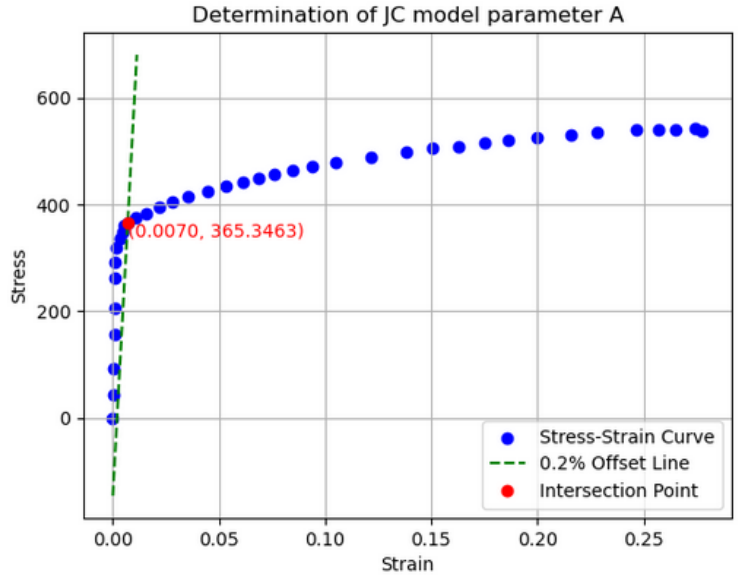


Figure 19: Determination of JC model parameter A

4.2. Determination of B and n

The blue curve can as well be utilized to capture the influence of strain on the material behavior of aluminum 2024-T3. From equation 29, it can be deduced that for the equation to eliminate the effect of strain rate and temperature, the strain rate $\dot{\epsilon}$ must equal the reference strain rate $\dot{\epsilon_0}$ and the temperature T must equal the room temperature T_{room}, creating equation 42.

$$\sigma_{flow} = A + B \cdot \epsilon^n \qquad 42$$

To analyze the effect of plastic strain on the flow stress equation, curve fitting is performed using a Python script. This script utilizes the curve-fit function from the SciPy library to determine the optimal values of parameters B and n through a nonlinear least squares method. The script is provided in Appendix C.3. Executing this script produces the fitted curve, which is shown in Figure 20. The corresponding parameter values for this curve are $B = 380.4\ MPa$ and $n = 0.5449$. It should be noted that these values are derived using a predefined initial guess within the curve-fit function and therefore may not correspond to the global minimum of the least squares error. As a result, while the fit is locally optimal, it is possible that alternative initial guesses could yield improved parameter estimates.

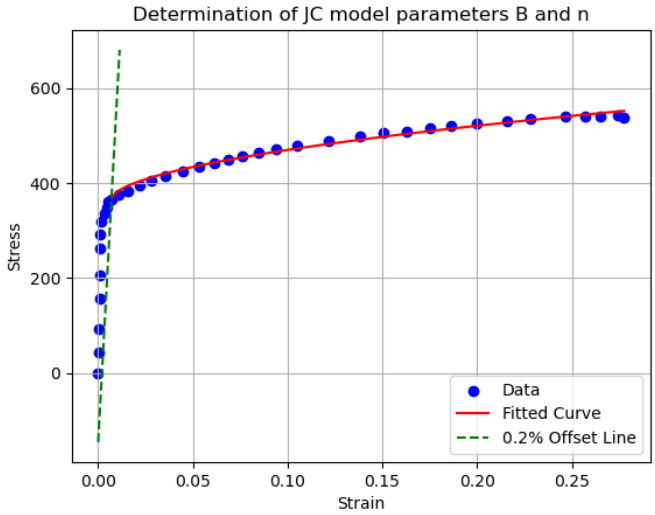


Figure 20: Curve fitting utilized to extract B and n

4.3. Determination of C and m

To quantify the influence of strain rate sensitivity (parameter *C*) and thermal softening (parameter *m*), additional stress-strain datacurves incorporating these effects must be analyzed. Beginning with the effect of strain rate, the curves presented in Figure 18 can be considered. This figure illustrates five distinct stress-strain responses, each corresponding to a different applied strain rate, thereby enabling evaluation of the material's strain rate dependence.

To determine a value for the strain rate sensitivity parameter *C*, a Python script was developed, as presented in Appendix C.4. Within this script, equation 29 is reduced such that the effect of temperature is nullified. The resulting simplified expression is provided in equation 43.

$$\sigma_{flow} = [A + B \cdot \epsilon^n][\,1 + C \cdot \ln \dot{\epsilon}^\star] \qquad 43$$

Rearranging this equation such that the strain rate dependency is isolated leads to equation 44 (Murugesan & Won Jung, 2019).

$$\frac{\sigma_{flow}}{[A + B \cdot \epsilon^n]} = 1 + C \cdot \ln \dot{\epsilon}^\star \qquad 44$$

By substituting the previously determined values for parameters A, B and n from sections 4.1 and 4.2 into the simplified equation, a plot can be constructed for each of the five experimental stress-strain curves. In this plot, the horizontal axis represents the natural logarithm of the normalized strain rate, $\ln \dot{\epsilon}^\star$. To extract the value of parameter C, linear regression is applied to the transformed data. This procedure yields a linear relationship, with the slope of the fitted line corresponding to the strain rate sensitivity parameter C. The resulting plot, which includes both the experimental data and the fitted regression line, is shown in Figure 21. The derived value for $C = -0.0045$.

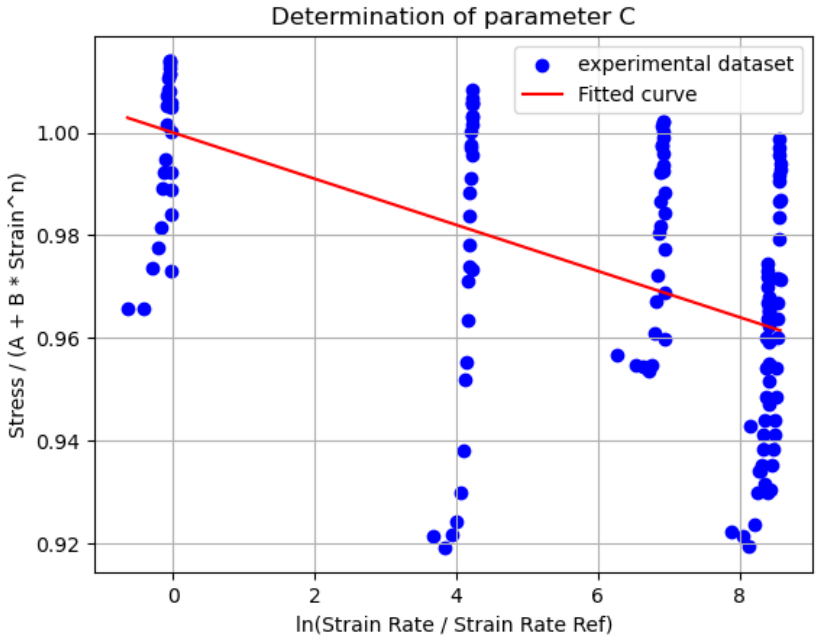


Figure 21: Curve fitting utilized to extract C

Similarly, the effect of thermal softening can be evaluated. By simplifying equation 25 to eliminate the influence of strain rate, the temperature-dependent form of the equation is obtained, as shown in equation 45.

$$\sigma_{flow} = [A + B \cdot \epsilon^n][1 - T^{\star m}] \quad (45)$$

Rearranging this equation to an expression in which the temperature effects are isolated gives equation 46 and, by extension, equation 47 (Murugesan & Won Jung, 2019).

$$1 - \frac{\sigma_{flow}}{[A + B \cdot \epsilon^n]} = T^{\star m} \quad (46)$$

$$\ln\left(1 - \frac{\sigma_{flow}}{[A + B \cdot \epsilon^n]}\right) = m * \ln(T^{\star}) \quad (47)$$

Determining the parameter m requires plotting experimental data in the format specified by equation 36 and applying linear regression. This would yield a rather similar plot to the one given in Figure 21, with $\ln\left(1 - \frac{\sigma_{flow}}{[A+B\cdot\epsilon^n]}\right)$ on the y-axis and $\ln(T^{\star})$ on the x-axis. The parameter m would then be the slope of the linear curve that results from the linear regression. However, stress-strain curves for aluminum 2024-T3 at various distinct temperatures are not always readily available in public resources. This may be due to several factors, such as the high cost of the testing process or the data being protected as intellectual property. As a result, no suitable dataset could be used to determine a value for m. Consequently, for the purpose of completion and continuation of this study, the parameter *m* is assumed to be 1, thereby effectively eliminating the influence of temperature from the JC model.

4.4. Critical Assessment of Parameter C

As illustrated in Figure 21, the curve-fitting procedure for determining parameter *C* presents significant challenges due to the structure and distribution of the experimental data in the format described by equation 44. Consequently, the resulting value for *C* represents an averaged fit across all available data points, which, while suitable as an initial estimate, may not fully capture the nuanced strain-rate dependence observed in the material behavior. Closer inspection of the fitted curve reveals that lower strain levels tend to correspond with lower values of *C*, whereas higher strain levels are associated with increased values. This observation can be clearly visualized by Figure 22.

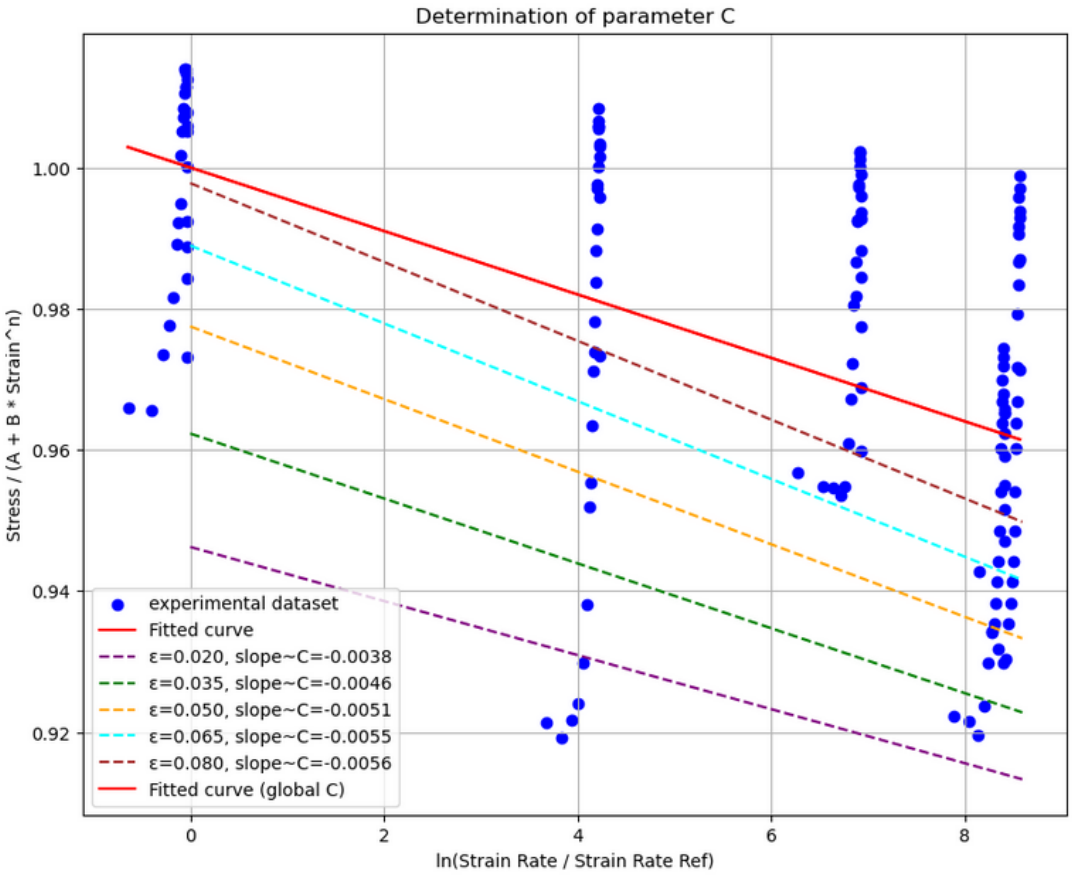


Figure 22: Observed variation of parameter C with equivalent plastic strain

This trend suggests that a more sophisticated, adaptive parameterization approach could have been employed. Specifically, one in which *C* is treated as a function of the equivalent plastic strain. By incorporating strain-dependent behavior into the definition of *C*, the model could more accurately reflect the material's dynamic response over a wider range of deformation conditions. A similar approach could have been applied to the thermal softening coefficient m, provided that experimental data exhibit a comparable dependency on equivalent plastic strain.

Since exploring this approach to parameterizing the value of C requires a complex implementation process later on in the numerical simulation framework, it is considered beyond the scope of this thesis and is therefore proposed as a recommendation for future study in Section 7. For the remaining parts of this study, the previously determined constant value of C will be maintained.

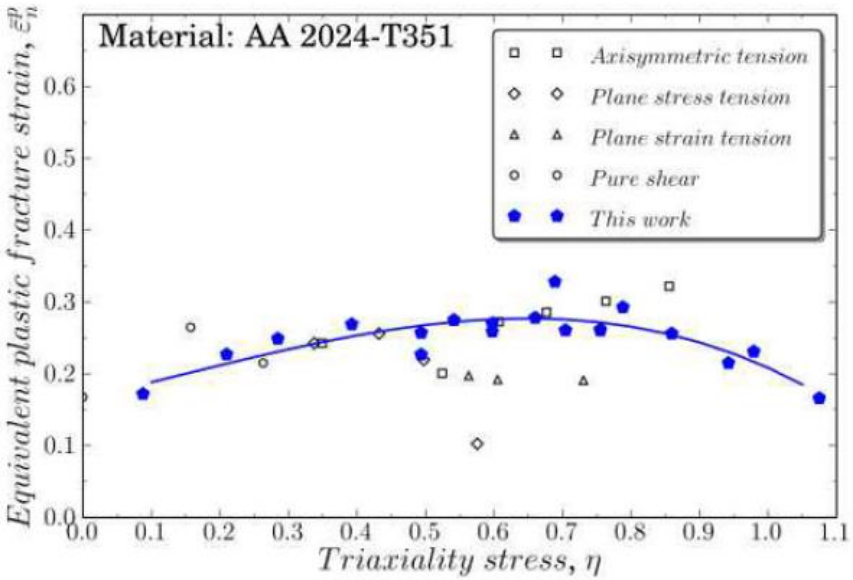
4.5. Determination of D1, D2 and D3

With the material parameters for the JC constitutive model now established, focus can now shift to defining the parameters associated with the corresponding damage model. As indicated by equation 33, the fracture strain is influenced by the stress triaxiality ratio, strain rate, and temperature. The parameters D_2 and D_3 characterize the dependence on stress triaxiality, while D_4 governs strain rate sensitivity and D_5 captures the influence of temperature.

To determine these parameters, the analysis begins by isolating the effect of the stress triaxiality ratio. By neglecting the influence of strain rate and temperature, the damage model can be simplified, resulting in the formulation presented in equation 48 (Murugesan & Won Jung, 2019).

$$\epsilon_f = D_1 + D_2 \cdot e^{D_3 \cdot \eta} \qquad 48$$

To determine the parameters D_1, D_2 and D_3, experimental data are necessary. These data should include a graph plotting the failure strain against the stress triaxiality ratio. Such a dataset is provided in the paper *"Failure Behavior of 2024-T3 Aluminum Under Tension-Torsion Conditions"* by Rodríguez-Millán et al. The dataset is illustrated in Figure 23.



(Rodríguez-Millán, Vaz-Romero, & Arias, 2015) Figure 23: Fracture locus of aluminum 2024-T3 on the space of equivalent plastic strain versus stress triaxiality ratio

The paper utilizes a simplified version of the Bao-Wierzbicki (BW) damage model, which differs distinctly from the JC damage model. The curve depicted in Figure 23 is derived from this specific damage model. To find parameter values for D_1, D_2 and D_3 and find a suitable curve depicting the JC damage model, the data was carefully digitized and included in Appendix B for reference.

With the application of the curve-fitting approach outlined in Section 4.2, a Python script was developed. This script takes the digitized dataset and applies the curve-fitting process to extract the parameters D_1, D_2 and D_3. The results generated by this Python script are depicted by Figure 24 and the parameter found are $D_1 = -132.7$, $D_2 = -133.0$, and $D_3 = 0.0004$.

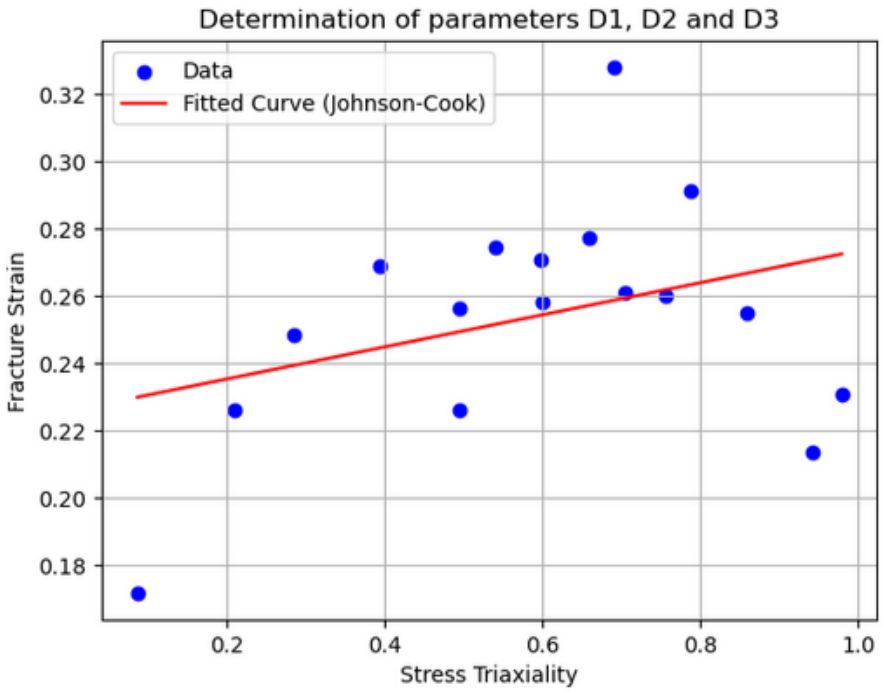


Figure 24: JC damage model D1, D2, and D3 fitted curve

4.6. Determination of D4 and D5

To determine the parameters D_4 and D_5, equation 33 must be reformulated to isolate the effects of strain rate and temperature individually. By neglecting the influence of temperature, the expression simplifies to equation 49, which isolates the strain rate contribution (Murugesan & Won Jung, 2019).

$$\frac{\epsilon_f}{(D_1 + D_2 \cdot e^{D_3 \cdot \eta})} = \left(1 + D_4 \cdot \ln\left(\frac{\dot{\epsilon}}{\dot{\epsilon_0}}\right)\right) \qquad 49$$

The Python script provided in Appendix C.8 assumes the dataset described in Section 4.1, along with the previously determined values for D_1, D_2 and D_3 as input. The script plots a data point for each curve within the dataset based on equation 38 and searches for the best curve fit by optimizing D_4. The outcome of the Python script is shown in Figure 25 and the parameter value $D_4 = 0.0072$ is found.

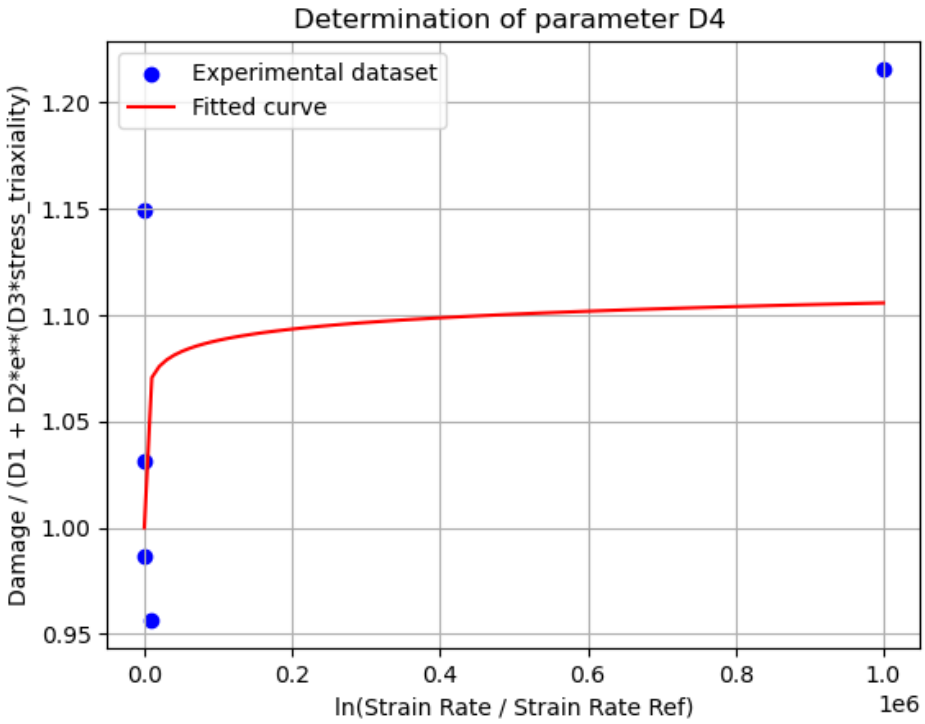


Figure 25: JC damage model D4 fitted curve

For the temperature, equation 33 can be rewritten as equation 50.

$$\frac{\epsilon_f}{(D_1 + D_2 \cdot e^{D_3 \cdot \eta})} = 1 + D_5 \left(\frac{T - T_{room}}{T_{melt} - T_{room}} \right) \quad 50$$

Similarly to the determination of D_4, the parameter D_5 can be evaluated by analyzing a dataset comprising temperature-dependent stress-strain curves. However, as previously discussed in Section 4.3, such datasets are difficult to obtain due to limited public availability, high experimental costs, or proprietary restrictions. As a result, the parameter D_5 could not be determined within the scope of this study and will be set to zero to ignore any temperature effect.

4.7. Damage Model Reliability and Revision

In the preceding two sections, the JC damage model was parameterized. However, based on the results of this parameterization process, questions arise regarding the model's accuracy in capturing damage accumulation. This concern is particularly evident when using the coefficient of determination (R^2). (New Castle University, n.d.) This coefficient quantifies how well a regression model explains the variability of the dependent variable based on the independent variable. This coefficient can be calculated by equation 40.

$$R^2 = 1 - \frac{\sum_n^{i=1} (y_i - \hat{y}_i)^2}{\sum_n^{i=1} (y_i - \bar{y})^2} \quad 51$$

Where

- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- $\bar{y}$ is the mean of all actual values
- n is the number of datapoints
- $0 < R^2 < 1$

It should be noted that for $R^2 = 1$, the regression model is a perfect fit. For $0 < R^2 < 1$ the model explains some of the variance in the data. If $R^2 = 0$, the model performs no better at predicting than the mean. When $R^2 < 0$, the model is worse than using the mean as a predictor.

Calculating R^2 for the curve fits shown in Figure 24 and Figure 25 results in the values given below.

- $R^2_{D1,2,3} = 0.1192$
- $R^2_{D4} = -0.2456$

To improve the predictive accuracy of the damage model, two key modifications are proposed. The first involves replacing the original failure strain formulation given by equation 33 with a revised expression, presented as equation 52. This updated equation is based on the BW damage model, which offers greater flexibility compared to the JC formulation. (Gilioli , Manes , Giglio, & Wierzbicki, 2014) Unlike the JC model's unified analytical form, the BW approach enables the derivation of failure strain equations through empirical curve fitting, thereby allowing for a more tailored representation of material behavior under varying conditions (Gilioli, Manes, Giglio, & Wierzbicki, 2014).

$$\epsilon_f = (D_1 + D_2 \cdot \eta + D_3 \cdot \eta^2)\left(1 + D_4 \cdot \ln\left(D_5 \cdot \frac{\dot{\epsilon}}{\dot{\epsilon}_0}\right)\right) \qquad 52$$

This modification introduces a parabolic dependency on the stress triaxiality ratio (η), which is essential for achieving the curve fit depicted in Figure 24. Furthermore, it allows for a more accurate representation of the asymptotic behavior that is observed in Figure 25. The second proposed adjustment involves the removal of a clear outlier visible in Figure 25. This anomaly originates from the red curve shown in Figure 18, which diverges from the expected inverse relationship between strain rate and failure strain. Excluding this data point is expected to significantly improve the accuracy of the curve-fitting procedure.

By applying these two modifications to the Python scripts used to calculate the damage parameters, an updated curve fit can be made for both Figure 24 and Figure 25. These updated curves compared to the JC counterparts can be seen within Figure 26, and the updated parameters D_1 to D_5 of the BW model are listed below.

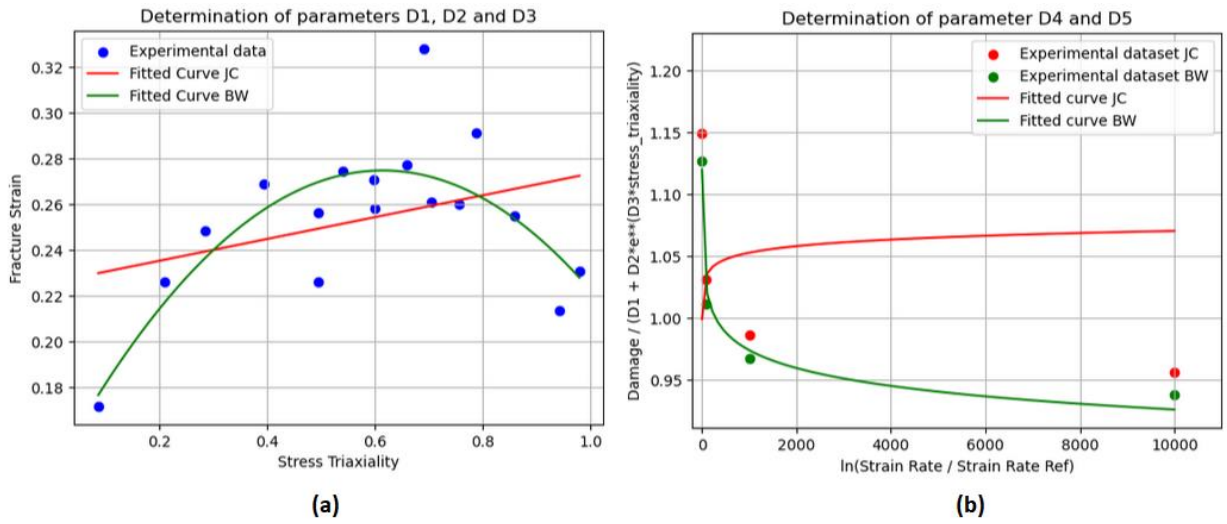


Figure 26: Revised curve fits. (a) incorporating parabolic dependency on stress triaxiality (b) removal of the outlier

- $D_{1_{BW}} = 0.1415$
- $D_{2_{BW}} = 0.4340$
- $D_{3_{BW}} = -0.3530$
- $D_{4_{BW}} = -0.0222$
- $D_{5_{BW}} = 0.0045$

The calculation of the coefficient of determination for these updated curve fits yields the following values:

- $R^2_{D1,2,3_{BW}} = 0.6175$
- $R^2_{D4,5_{BW}} = 0.9862$

With the revised model evaluated, a comparative analysis can be conducted between the custom BW damage model and the initially proposed JC model. An initial assessment of Figure 26 suggests that the BW model provides a more physically consistent representation of the damage behavior than its JC counterpart. This improvement is further supported by the coefficients of determination. Specifically, for damage datasets $D_{1,2,3}$, the JC model yields $R^2_{D1,2,3} = 0.1192$, whereas the BW model achieves a significantly higher value of $R^2_{D1,2,3} = 0.6175$. Similarly, for datasets $D_{4,5}$, the JC model results in a negative coefficient $R^2_{D4,5} = -0.2456$, indicating poor predictive performance, while the BW model attains a near-perfect fit with $R^2_{D4,5} = 0.9862$. These results clearly indicate that the revised BW damage model offers a substantial improvement over the original JC formulation.

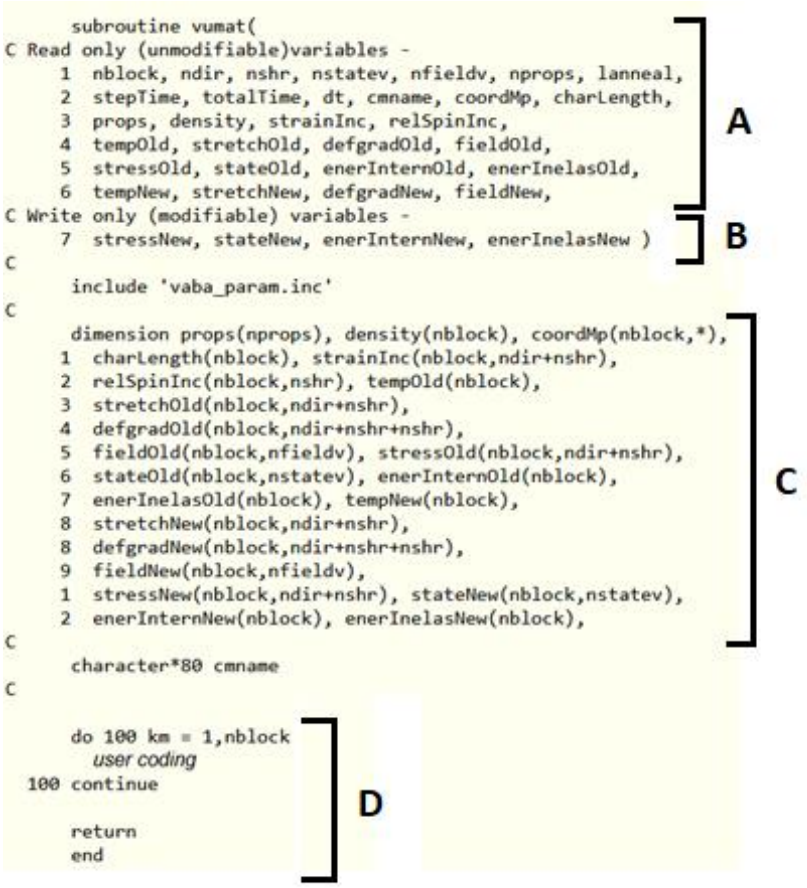
5. Methodology - Abaqus Subroutines

In the Abaqus finite element software, users can define custom constitutive material models through user-written subroutines. These subroutines are categorized into two types: UMAT and VUMAT. While both allow for the implementation of user-defined material behavior, they differ in terms of solver compatibility. UMAT is used in conjunction with the implicit solver, whereas VUMAT is specifically designed for explicit dynamic analysis (Zohrevand, 2023). As outlined in Section 2.4, bird strike simulations involve highly dynamic loading conditions, necessitating the use of the explicit solver. Consequently, VUMAT is the appropriate subroutine for implementing the constitutive material model in such analyses.

5.1. Vumat Baseline Code

At its core, a VUMAT is a user-defined subroutine - typically written in Fortran - that is invoked by the Abaqus Explicit solver at each time increment for a group of material calculation points. The solver supplies a predefined set of input variables that users can utilize when coding their constitutive model. Within the subroutine, the user is responsible for updating a specific set of output variables, which are then carried forward to the next increment in the simulation (Abaqus, n.d.).

As a stepping stone, the Abaqus documentation provides the user with a baseline code that can be altered to fit the user's objectives. This baseline code is shown in Figure 27.



(Abaqus, n.d.) Figure 27: Vumat baseline code

The code given in Figure 27 can be broken down into 4 parts, which are numbered A, B, C, and D. The purpose of each part is specified below.

- Part “A” declares all the unmodifiable input variables given by Abaqus.
- Part “B” declares all the modifiable variables that can be updated and passed on to Abaqus. The variables stressNew and stateNew are required to be udpated, and enerInternNew and enerInelasNew are optional.
- Part “C” specifies the dimensions of all the variables declared within parts A and B.
- Part “D” specifies the loop over the calculation points in which the constitutive model must be defined. The user is expected to supplement the code for the desired constitutive material model at the “*user coding”* location.

To implement the JC material model within the VUMAT baseline code, it is essential to first identify and understand the variables that are relevant to this specific constitutive formulation. A thorough examination of both modifiable and predefined (non-modifiable) variables reveals a subset that plays a critical role in the implementation. These key variables are outlined below.

- **nblocks:** Number of material points processed within the current call to VUMAT
- **ndir:** Number of direct components in symmetric tensor (principal directions)
- **nshr:** Number of indirect components of symmetric tensor (shear directions)
- **steptime:** Value of time since the beginning of the step
- **dt:** Time increment size
- **props:** Material properties specified by user
- **strainInc:** Strain increment tensor for each material point
- **tempOld:** Temperatures of each material point at the beginning of the increment
- **stressOld:** Stress tensor for each material point at the beginning of the increment
- **stateOld**: State variables for each material point at the beginning of the increment
- **stressNew:** Stress tensor for each material point at the beginning of the increment
- **stateNew:** State variables for each material point at the beginning of the increment

All other variables are deemed irrelevant for the application of the JC model. Definitions for these variables can be found in the Abaqus documentation if needed.

5.2. Applying Johnson-Cook to VUMAT

With the baseline code thoroughly analyzed, user-defined additions can now be integrated at the designated location. A flowchart illustrating the structure and logical flow of the proposed Fortran code is provided in Appendix D, offering a visual summary of these additions. For completeness, the full Fortran code of the VUMAT subroutine is included in Appendix E. The subsequent sections provide a detailed discussion of the implemented additions, outlining their functionality and the rationale behind their integration into the overall code structure.

5.2.1. Initialization Phase

The first addition that was made can be categorized as the initialization phase. In the initialization phase of the VUMAT subroutine, 3 distinct parts can be established. The 3 parts consist of declaring all relevant variables, material properties assignment, and variable constant allocation. Together these parts must ensure a well-defined foundation for following computations and enabling accurate material behavior modeling.

5.2.1.1. Declaring Variables

The first part of the code, located in lines 10 to 40 of the Fortran code provided in Appendix E, is dedicated to the declaration of all variables. This section serves as a modified extension of Part C in the baseline code, specifying both the type and dimensions of the variables. To prevent incorrect assumptions in later assignments, the statement implicit none is included. This directive ensures that all variables must be explicitly declared, enforcing clarity and reducing the risk of unintended errors throughout the code.

5.2.1.2. Assigning Material Properties & Variable Constants

In lines 41 to 57, the material properties passed in by Abaqus are assigned to local variables, improving code readability. Similarly, in lines 60 to 67, various miscellaneous variables are initialized with constant values to ensure consistency and stability throughout the remainder of the code.

5.2.2. Iteration Phase

After the initialization phase, the code begins iterating over the integration points provided by Abaqus in line 74. The purpose of this iteration is to update the stress vector (stressNew) and state variables (stateNew) for each integration point. Once all integration points have been processed, the updated information is passed back to Abaqus for further analysis.

5.2.2.1. Integration-point-specific variables, elastic trial stress & stress correction term

The first step in the iteration process involves initializing integration-point-specific variables, as outlined in lines 77 to 94 of the code. This includes assigning values to the state variables, retrieving the stress vector from the previous increment, and defining the strain increment vector. These variables serve as inputs for the subsequent computations within the current iteration. The second step entails calculating the elastic trial stress vector, which is derived under the assumption that the entire strain increment is elastic. Assuming linear elastic behavior, the trial stress vector is computed using equation 53.

$$\boldsymbol{\sigma_{trial}} = \boldsymbol{\sigma_{old}} + \boldsymbol{C} : \boldsymbol{\Delta\varepsilon} \qquad 53$$

Next, based on the trial stress and the associative flow rule described in Section 3.2.2.2, the plastic flow direction is determined, as defined by equation 54. This plastic flow direction is represented by a unit vector that points normal to the yield surface at the location of the trial stress, indicating the direction of plastic deformation in the 3-dimensional stress space.

$$\boldsymbol{plasticflowdirection} = \frac{3 \cdot \boldsymbol{\sigma_{dev}}}{2 \cdot \sigma_{eq}} \qquad 54$$

To convert the plastic flow direction into a stiffness-like vector suitable for stress computation from a given strain vector, it must be multiplied by the elasticity matrix $\boldsymbol{C}$. In the VUMAT implementation, this resulting vector is referred to as the *stress correction term*.

5.2.2.2. Bisection Scheme – Evaluate Yield Function

To compute the incremental plastic strain, a bisection scheme is utilized to resolve the implicit dependence of the yield surface on the total equivalent plastic strain, which necessitates an iterative solution procedure. The bisection process begins at line 113 of the code, where initial lower and upper bounds are defined, representing the expected range of

the incremental plastic strain. An initial guess of zero is also assigned. The iterative loop commences at line 118, where the equivalent plastic strain rate is first computed, followed by an update of the total equivalent plastic strain. Subsequently, the updated stress vector is determined using the radial return method, as expressed in equation 55. Based on this updated stress state, the equivalent (von Mises) stress is calculated in accordance with equation 56.

$$\boldsymbol{\sigma_{new}} = \boldsymbol{\sigma_{trial}} - \epsilon_{plas_{inc_{eq}}} \cdot \boldsymbol{E_{stress_{correction}}} \quad 55$$

$$\sigma_{eq} = \sqrt{\frac{((\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2) + 6(\sigma_{12}^2 + \sigma_{23}^2 + \sigma_{31}^2)}{2}} \quad 56$$

In the subsequent step, the flow stress is computed using the JC constitutive equation, as defined in equation 29. This computation is carried out between lines 138 and 145 of the code. Following this, the yield function is evaluated at line 153, using the formulation provided in equation 57, to determine whether the trial stress state satisfies the yield criterion.

$$f = \sigma_{eq} - \sigma_y \quad 57$$

5.2.2.3. Bisection scheme – Elastic Exit

Once the yield function has been evaluated, four possible outcomes may be observed within the bisection iteration. The first is a special case in which the yield function is negative while the incremental plastic strain remains zero. This indicates that the material is still within the elastic regime and does not require plastic correction. This scenario is referred to as the elastic exit of the loop and is implemented in lines 156 to 162. Within these lines, the new stress state is put equal to the trial stress and the damage, and element deletion variables are updated accordingly.

5.2.2.4. Bisection scheme – Plastic Exit

The second possible outcome is the plastic exit, defined in lines 165 to 179. In this case, the value of the yield function falls within a specified tolerance near zero, indicating that the updated stress vector lies on the yield surface. Upon reaching this condition, the algorithm proceeds to update the damage variable and its associated parameters based on the BW damage model as defined in section 4.7. It then checks whether the damage at the integration point has reached the critical threshold of one. If so, the corresponding calculation point is removed from further analysis in subsequent time increments by setting the element deletion variable to zero.

5.2.2.5. Bisection Scheme – Undershooting Yield Surface

The third possible outcome occurs when the radial return to the yield surface is insufficient. This condition is handled in lines 181 to 184. In this case, the yield function returns positive while the equivalent plastic strain increment is greater than zero. This indicates that the trial stress still lies outside the yield surface, and the plastic correction is not yet sufficient. As a result, the lower bound of the bisection interval is updated to the current value of the equivalent plastic strain increment. A new bisection iteration is then initiated by recalculating the midpoint between the updated lower and upper bounds, as implemented in line 192.

5.2.2.6. Bisection scheme – Overshooting Yield Surface

The fourth possible outcome arises when the radial return to the yield surface is too aggressive. This condition is addressed in lines 187 to 189. Here, the yield function becomes negative while the equivalent plastic strain increment is greater than zero. This indicates that the trial stress has overshot the yield surface, pushing the stress state inside the elastic domain. Consequently, the upper bound of the bisection interval is updated and reduced to the current value of the equivalent plastic strain increment. A new bisection iteration is then initiated by recalculating the midpoint between the revised lower and upper bounds, as implemented in line 192.

5.2.2.7. Update State Variables

Once the bisection scheme has converged and the final stress state has been determined, the iteration for the current calculation point is concluded by updating its stress and state variables. This update is carried out in lines 196 to 204. Following this, the iteration ends with the corresponding end-of-iteration and end-of-VUMAT statements.

5.3. Testing and Verification of VUMAT

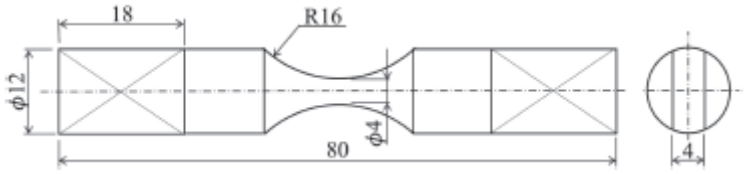
With the VUMAT implementation completed and ready for use, a testing and verification process is essential. The goal of this process is to answer the question: *Was the VUMAT built correctly?* To facilitate this, a dummy program has been developed to debug the VUMAT where necessary, which is given in Appendix F. This program models one single calculation point subjected to strain in only one principal direction. Additionally, it assumes predefined values for all relevant variables to ensure controlled and repeatable conditions. The Fortran code for this dummy program, along with the initial predefined values, is provided in Appendix F. This setup allows the user to debug the VUMAT code while remaining within the Fortran environment. As a result, the debugging process becomes significantly more efficient, enabling the use of logical checks to verify the correct functioning of the VUMAT implementation.

6. Model Validation and Discussion

With all necessary components in place, the application and validation of the VUMAT can now proceed. The objective of this phase is to answer the question: *Was the correct VUMAT built?* To achieve this, the VUMAT will be applied to the dogbone test specimen described in the study referenced in Section 4.1. The validation will be considered successful if the experimental data within this paper can be reproduced with reasonable accuracy.

6.1. Setting up Abaqus

The initial step in the application and verification process involves constructing an Abaqus model of the dogbone specimen described in the referenced study. The geometric dimensions required for this model are presented in Figure 28.



(Masahiro , Satoshi , Ziyi , Masaki , & Masanobu , 2023) Figure 28: Dogbone specimen dimensions

Once the specimen shown in the figure has been modeled, the corresponding Abaqus configuration, as illustrated in Figure 29, can be generated. All steps involved in constructing the model - including part design, material assignment, and assembly creation - are detailed in Appendix G.

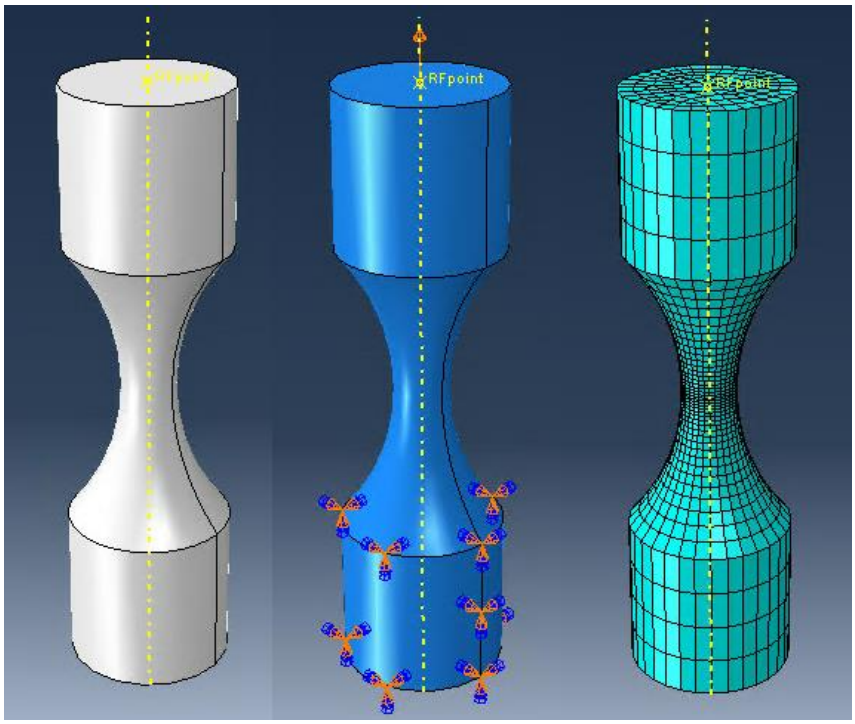
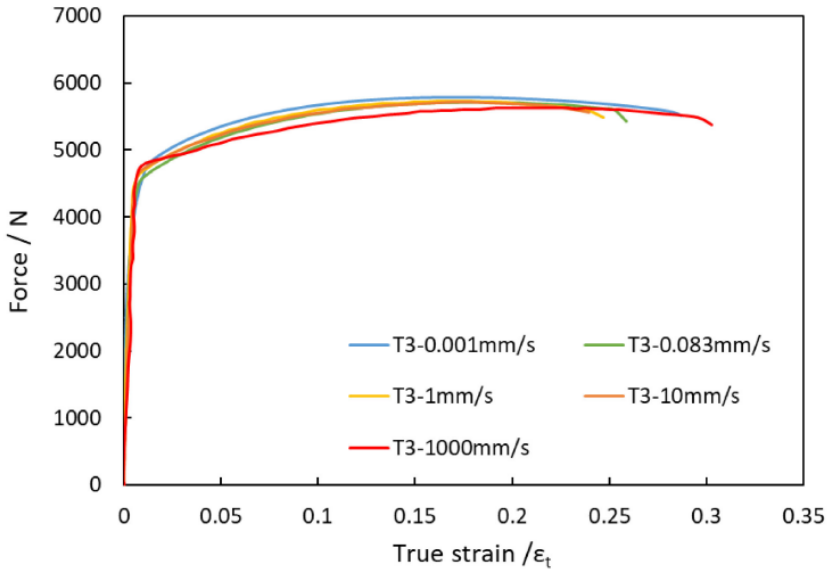


Figure 29: Abaqus dogbone specimen model, showing (from left to right) the part, load, and mesh modules.

6.2. Running Jobs

The referenced study includes five distinct tests, the results of which are presented in Figure 18 as stress-strain curves. In addition to these, the original paper also provides force-strain curves for the same set of tests, which are shown in Figure 30.



(Masahiro , Satoshi , Ziyi , Masaki , & Masanobu , 2023) Figure 30: Tensile force-strain test datasets for aluminum 2024-T3

6.2.1. Application of Loading Rate

To replicate the results shown in Figure 18 and Figure 30, five distinct Abaqus simulation jobs were created using the model described in the previous section. The differences between each simulation lie in the duration over which the displacement boundary condition is applied and the size of the time increment used. These variations were implemented by modifying the time period and user-defined time increment in the analysis step, as detailed in Step 3 of Appendix G, and by adjusting the final time in the amplitude definition of the displacement boundary condition to match the corresponding time period, as outlined in Step 4. The specific time periods and time increment values applied in each simulation are summarized in Table 1.

Table 1: Time periods per simulation job

Test loading rate $[mm/s]$	**Time period** $[s]$	**Time increment** $[s]$
0.001	2000	$1 \cdot 10^{-3}$
0.083	24.096	$1 \cdot 10^{-5}$
1	2	$1 \cdot 10^{-6}$
10	0.2	$1 \cdot 10^{-7}$
1000	0.002	$1 \cdot 10^{-9}$

6.2.2. Submission of Simulation Jobs on HPC System

Due to the high computational demand of the simulation jobs - each requiring approximately 2,000,000 increments - they are executed on the HPC12 cluster provided by TU Delft. This high-performance computing system operates on a CentOS Linux environment, necessitating command-line-based execution. To run the Abaqus jobs on this machine, the following command is used: ”*qsub abaqus_job.qsub”.* This command executes the file *abaqus_job.qsub*, which consists of the code given in Figure 31.

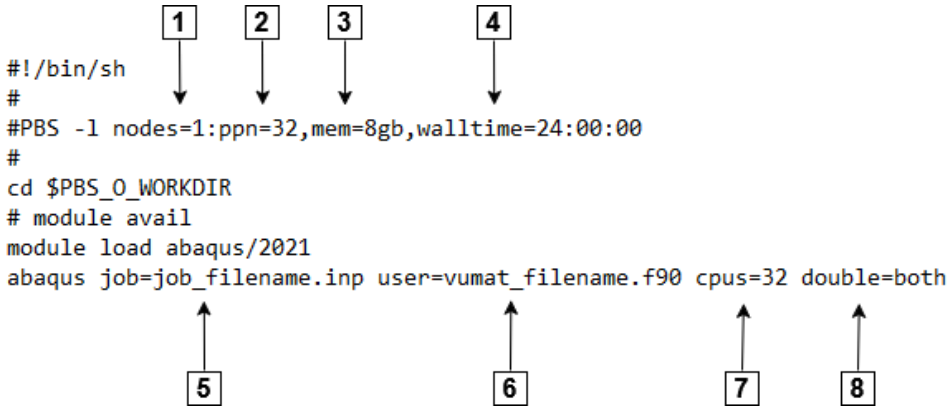


Figure 31: Qsub file code

Several key parameters within this script are highlighted by the numbered annotations in the figure. These parameters can be modified to suit the user's specific requirements and are described below.

1. Number of nodes requested for the job
2. Number of CPUs allocated per node
3. Estimated memory required for the analysis
4. Expected runtime of the simulation
5. Name of the Abaqus input file
6. Name of the user-defined VUMAT Fortran file
7. Total number of CPUs to be used
8. Command to enable double precision for all variables

After execution on the HPC12 cluster, the output for all simulation jobs is provided in the form of .odb files. These output database files can then be transferred to a local machine equipped with Abaqus/CAE for postprocessing and analysis of the simulation results.

6.3. Results

Once the .odb files have been transferred to a local machine, they can be accessed using the Visualization module within Abaqus/CAE. By selecting the appropriate output variables - such as reaction forces, as well as stress and strain components - it is possible to derive force-strain and stress-strain curves. These curves are generated by extracting XY data from either history output or field output and subsequently plotting the data.

It is important to note that, in the referenced experimental study, the true strain values presented along the x-axis were obtained using Digital Image Correlation (DIC). This technique enabled the measurement of strain within a highly localized region of the specimen, specifically surrounding the notched section, with a longitudinal extent of approximately 0.175 mm. To facilitate a meaningful comparison with the experimental results, strain values from the simulation were extracted from a single finite element situated within the same region of interest. This element, identified in Figure 32, was selected as the reference point for strain data extraction from the .odb files.

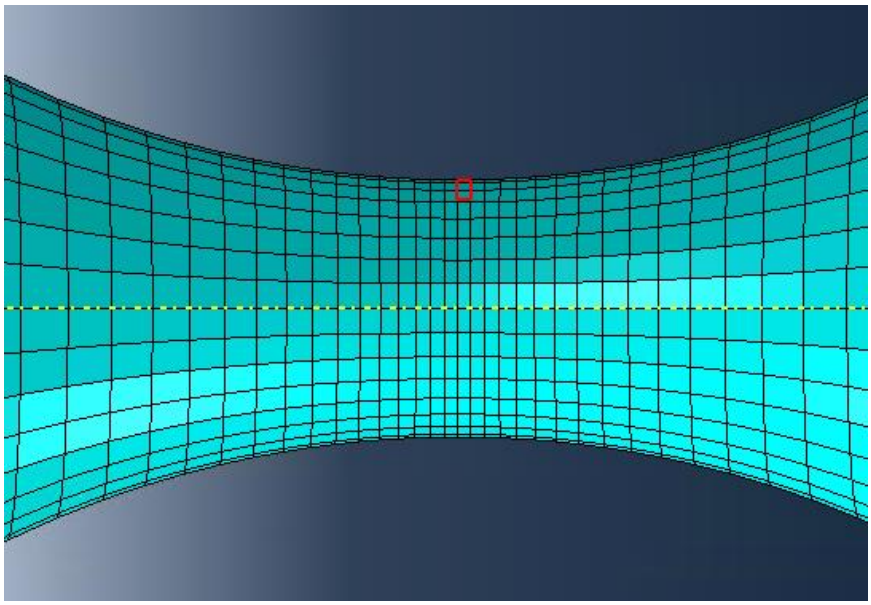


Figure 32: Element utilized for strain and stress output extraction

6.3.1. Mesh Convergence

An initial consideration in conducting simulations of the five tensile tests is the adequacy of the mesh in producing a reasonably converged solution. To evaluate mesh convergence, four similar finite element models with progressively increasing mesh densities were developed, with particular refinement in the notched region of the specimen. These meshes are classified as coarse, medium, fine, and superfine. Their respective characteristics are illustrated in Figure 33 and summarized in Table 2.

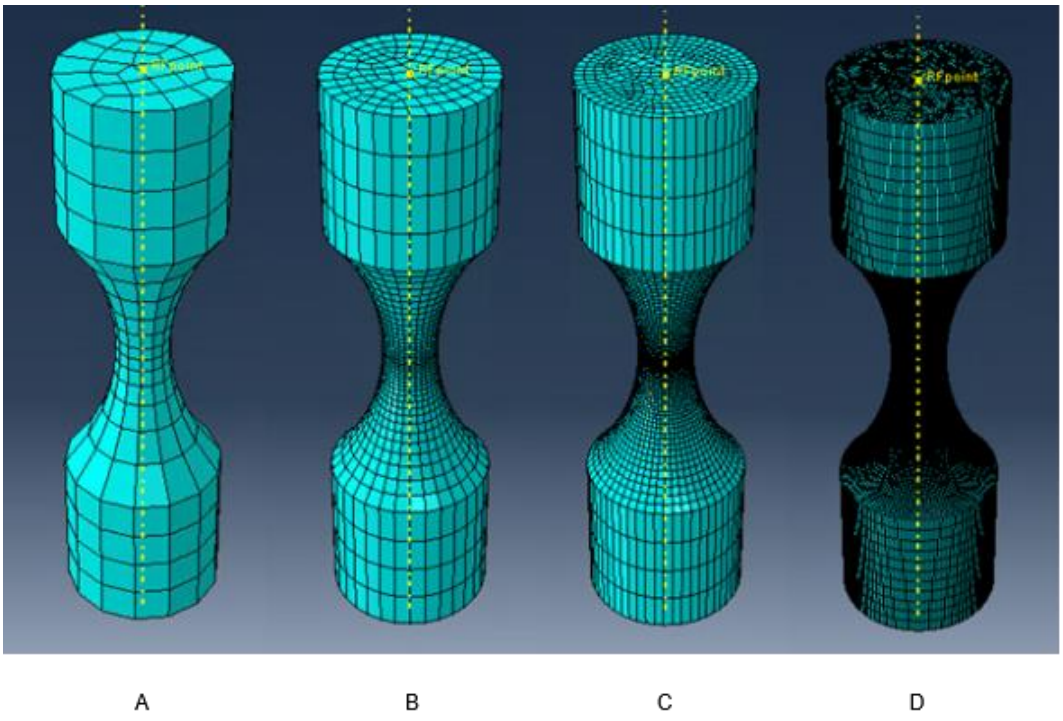


Figure 33: Mesh resolutions: (a) coarse mesh, (b) medium mesh, (c) fine mesh, and (d) superfine mesh.

Table 2: Mesh characteristics

Mesh density	Number of elements	Shortest element edge length [mm]	Average shortest element edge length [mm]
Coarse	560	0.416	1.1
Medium	5568	0.19	0.453
Fine	26322	0.1	0.252
Superfine	741576	0.02	0.0767

When the four different mesh densities are applied to simulate the five tensile tests, the resulting stress-strain and reaction force-strain curves, as presented in Appendix H, can be obtained. Analysis of these plots indicates that with each increase in mesh refinement, the results demonstrate improved convergence, as evidenced by the diminishing differences between successive mesh densities. Notably, the results from the fine and superfine meshes are nearly identical, suggesting that the fine mesh provides sufficient accuracy for the simulation. While the superfine mesh may offer marginal improvements in precision, its use incurs a substantial computational cost, significantly increasing simulation time. Therefore, the fine mesh is deemed adequate for most purposes, unless maximum accuracy is required, in which case the superfine mesh could be considered despite its computational demands.

6.3.2. Damage Model Accuracy

Another important aspect to evaluate is the accuracy of the implemented damage model. In the present simulations, the BW damage model, as defined by equation 52, is employed. The resulting stress-strain curves and corresponding failure points from the five tensile test simulations - conducted using a fine mesh - are presented in Figure 34. The corresponding failure strains and stresses are given in Table 3.

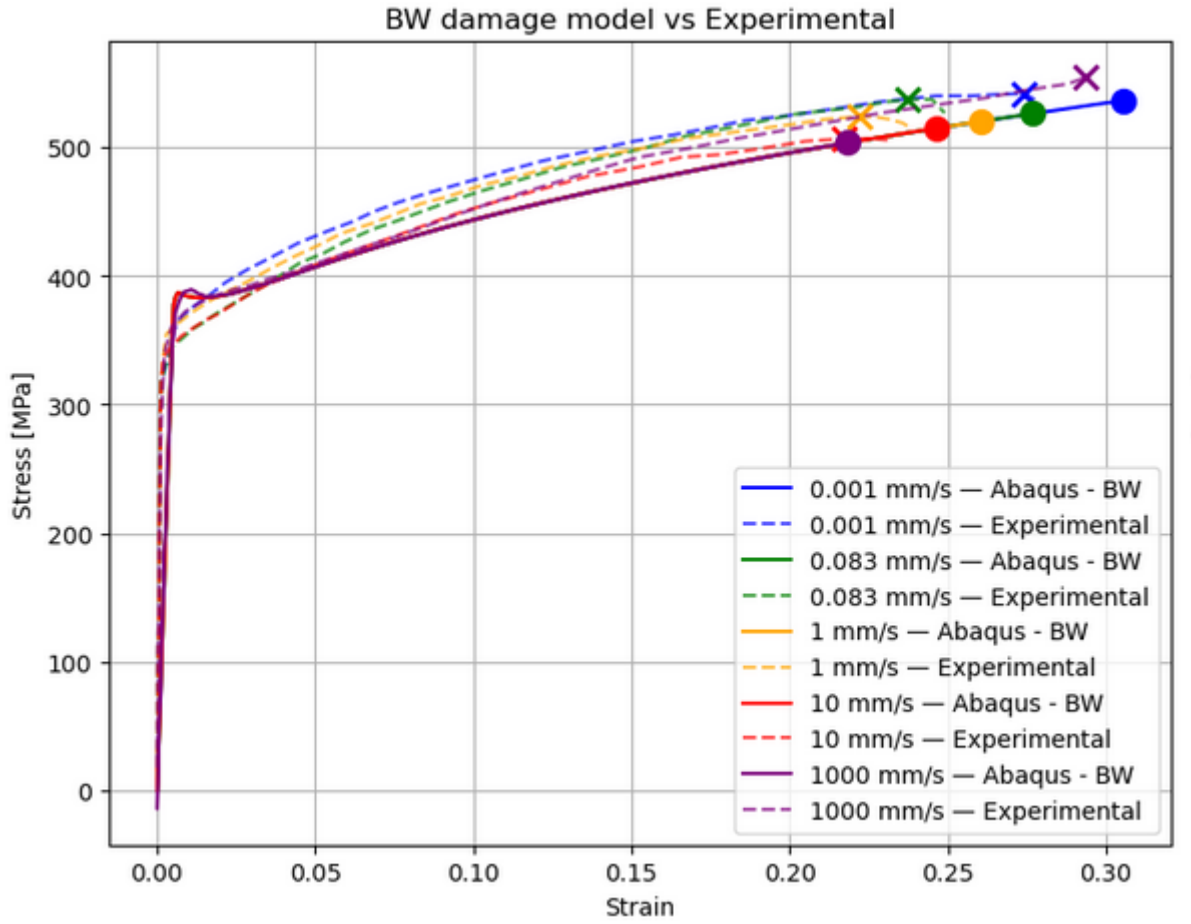


Figure 34: Failure point comparison - BW vs. experimental

As illustrated in Figure 34, the predicted trend in failure strain aligns well with the experimental data, with the exception of the test conducted at 1000 mm/s. This deviation was anticipated, as the corresponding data point was previously identified as an outlier in Figure 25. However, a consistent offset is evident across all remaining data points, with the BW model systematically overestimating the failure strain by approximately 0.03. This discrepancy suggests that the damage parameter D_1 is underestimated and should be increased to achieve better agreement with the experimental observations.

In Section 4.7, the transition from the JC damage model to the BW damage model was proposed, with the primary objective of improving the accuracy of parameter fitting for D_1 through D_5. This modification demonstrated enhanced curve-fitting accuracy for these parameters. To evaluate whether this improvement also translated into greater accuracy in the predictive capability of the damage model, a comparative set of simulations was conducted. These simulations replicated the original tensile tests, with the only modification being the use of the JC damage model and its corresponding parameter values. The results of the five tensile test simulations governed by the JC damage model are presented in Figure 35 and Table 3.

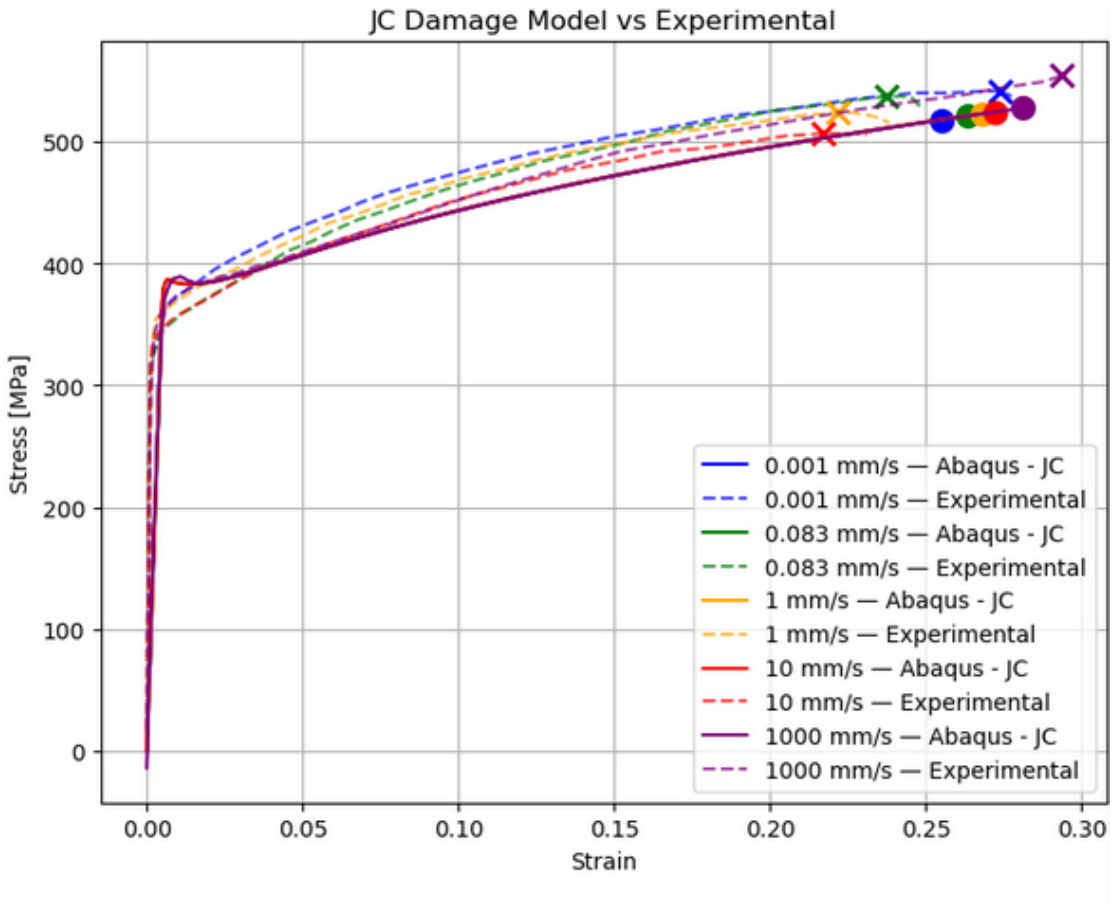


Figure 35: Failure point comparison - JC vs. experimental

Analysis of Figure 35 reveals that the JC damage model predicts an inverse trend, whereby the failure strain increases with increasing strain rate. This behavior contradicts the trend observed in the experimental data, where higher strain rates typically correspond to lower failure strains. Consequently, it can be concluded that the transition from the JC damage model to the BW model was justified, as it yields predictions that more accurately reflect the physical behavior observed in experimental results.

Table 3: Comparison of Failure Stress and Strain Predictions Between the BW and JC Damage Models

Strain Rate	Experimental Strain	Experimental Stress [MPa]	BW Model Strain	BW Model Stress [MPa]	JC Model Strain	JC Model Stress [MPa]
$0.001\ mm/s$	0.2740	541.69	0.3055	536.11	0.2549	517.76
$0.083\ mm/s$	0.2373	536.92	0.2768	525.99	0.2636	521.10
$1\ mm/s$	0.2219	523.57	0.2607	520.08	0.2680	522.77
$10\ mm/s$	0.2173	506.40	0.2463	514.55	0.2724	524.33
$1000\ mm/s$	0.2936	553.99	0.2184	503.44	0.2813	527.45

6.3.3. Material Model Accuracy – Replication of Experimental Data

Furthermore, the accuracy of the constitutive modeling of the material warrants further discussion. An evaluation of Figure 34 suggests that the simulated results only moderately replicate the experimental observations. One key indicator of this discrepancy is the presence of a stress irregularity near the yield point in the simulation results - a phenomenon absent in the experimental data. This localized anomaly is clearly illustrated in Figure 36. A second indicator of the limited fidelity in replicating the experimental results is the consistent underestimation of stress values across all simulated curves when compared to the corresponding experimental data.

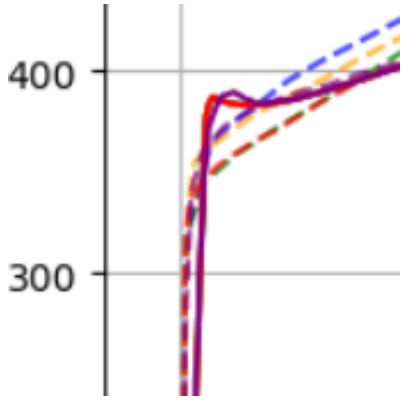


Figure 36: Stress fluctuation near the yield point

To further investigate these two indicators, plots of the principal stress components were generated for each of the five tensile test simulations. In all cases, the principal stresses exhibited a consistent pattern, closely resembling the behavior observed in the 0.001 mm/s test, as illustrated in Figure 37.

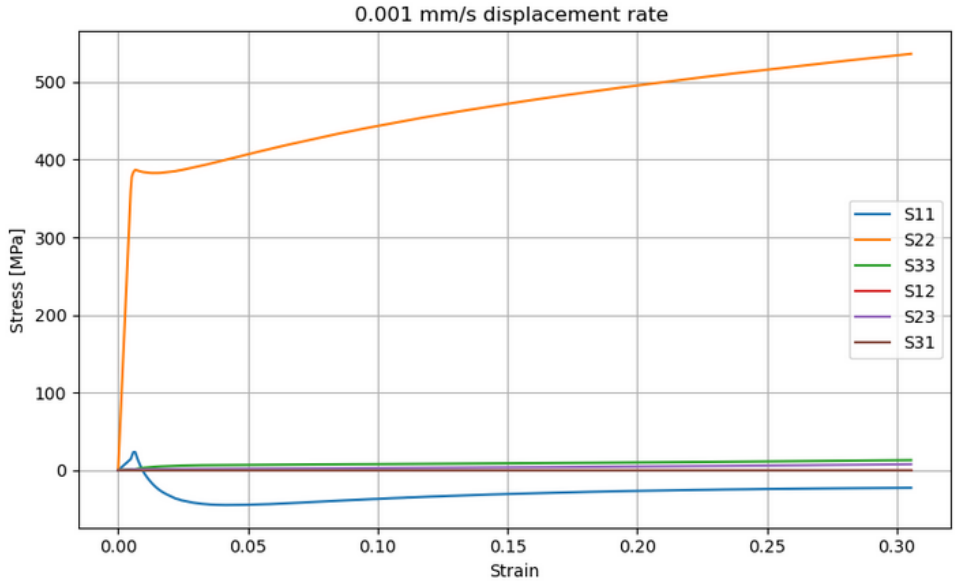


Figure 37: Principal stresses behavior

Within this figure, a significant presence of the secondary principal stress components - particularly in the 12-direction - is clearly observable. As discussed in Section 5.2.2.2, the yield function implemented in the VUMAT subroutine is based on the von Mises equivalent stress. Consequently, the appearance of additional principal stresses implies that the axial stress component (in the 22-direction) no longer directly corresponds to the von Mises equivalent stress. This deviation can be attributed to the geometric characteristics of the test specimen, which inherently promote the development of a multiaxial stress state, particularly in the necking region.

When the von Mises equivalent stress is plotted instead of the stress component in the 22-direction, the resulting comparison - presented in Figure 38 - demonstrates significantly improved agreement between the simulated and experimental stress-strain curves. This indicates that the previously observed discrepancies were primarily due to the multiaxial stress state present in the specimen, which is not captured when only the 22-direction stress is considered. A similar improvement in correlation is observed for the equivalent stress comparisons across the remaining four tensile tests.

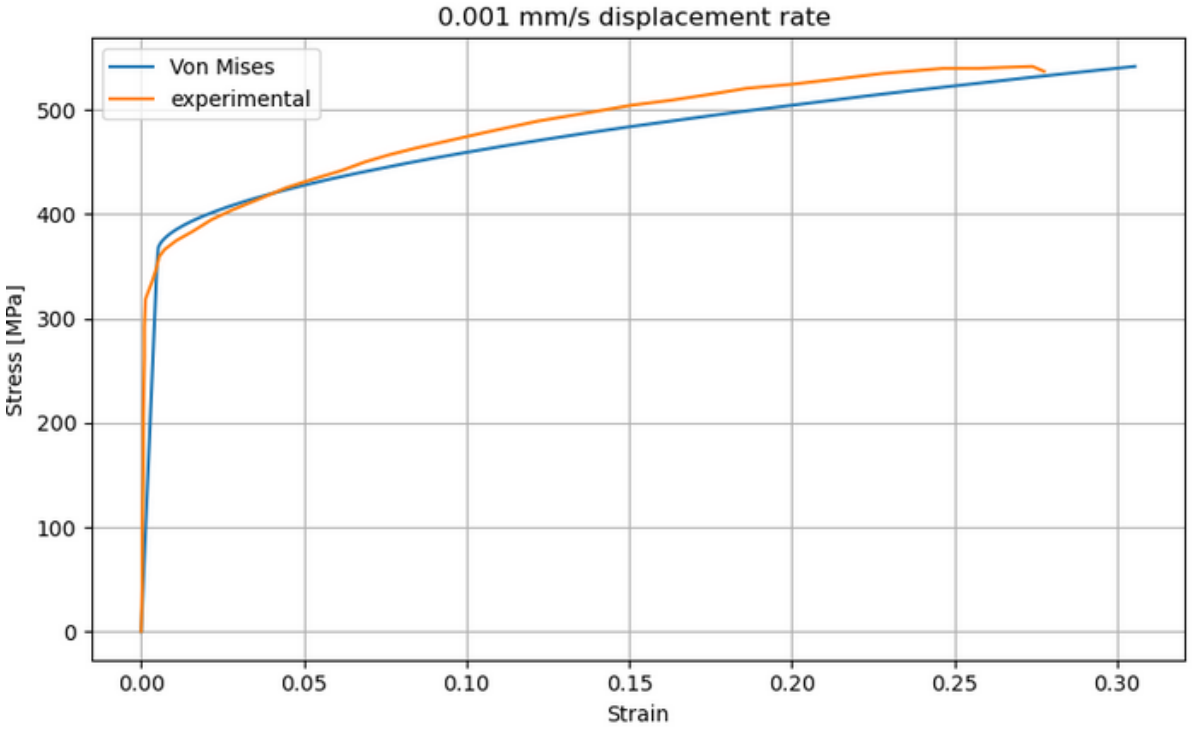


Figure 38: Comparison Between Simulated Equivalent Stress and Experimental Data

6.3.4. Material Model Accuracy – Strain Rate Sensitivity

A final aspect warranting discussion is the observation that all fine mesh simulations produce stress-strain curve paths that are identical, irrespective of the applied strain rate. This suggests a lack of strain rate sensitivity in the simulation results, which contradicts the experimental data where distinct differences between strain rates are evident.

Upon closer examination of the VUMAT implementation, it was concluded that lines 148 to 150 contain a constraint enforcing that the JC flow stress cannot decrease between time steps. This restriction effectively prevents the strain rate term from reducing the flow stress. Although tensile tests typically maintain a constant strain rate, this constraint artificially suppresses the material's expected strain rate dependency.

To investigate the impact of this limitation, the constraint was removed from the VUMAT code, and the simulations were repeated. The resulting stress-strain curves, shown in Figure 39, now no longer exhibit the lack of variation with strain rate, aligning more closely with experimental observations.

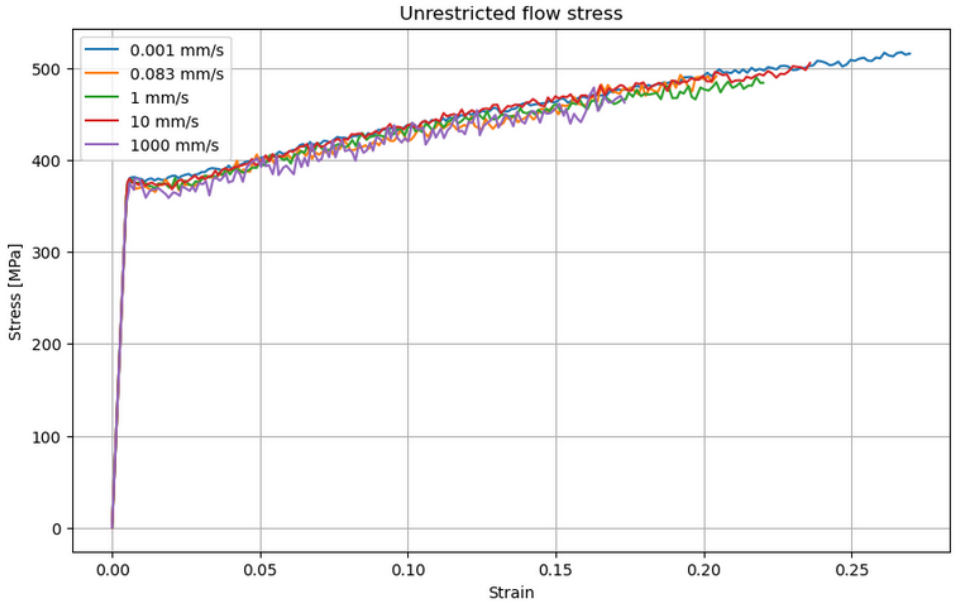


Figure 39: Simulation Results Without Flow Stress Limitation

However, the resulting curves display noticeable oscillatory behavior, which complicates the interpretation of strain rate effects. These oscillations likely stem from the excitation of higher-order frequencies. In undamped or underdamped systems, such frequencies persist due to the lack of sufficient energy dissipation. One way to mitigate these spurious oscillations is by introducing additional damping (Abaqus, n.d.). Nevertheless, as Figure 39 still allows for reasonable conclusions to be drawn, this step has been omitted.

From Figure 39, a small yet visible variation can be observed among the five tensile test data curves. The general trend indicates a decrease in material strength with increasing strain rate. This behavior aligns with expectations, given that the strain rate sensitivity parameter C in the JC model was previously determined to be negative.

7. Conclusion and Recommendations

This thesis has presented a comprehensive methodology for the development and validation of a constitutive material model for flexible targets, aimed at supporting high-strain-rate simulations such as bird strike analyses. The material behavior was implemented using a user-defined material subroutine (VUMAT), specifically developed for integration with Abaqus/Explicit to ensure accurate and efficient simulation under dynamic loading conditions.

This concluding chapter first summarizes the methodological steps undertaken throughout the research, then addresses the main research question along with its corresponding sub-questions. Finally, key insights are translated into recommendations for future work.

Methodology

The methodology employed for the development and validation process followed a structured sequence of steps as listed below. Together, these steps form the basis for applying advanced constitutive models in a numerical context.

- A review of relevant literature and theory to build a foundational understanding of impact mechanics, numerical simulation methods, and constitutive material modeling.
- Selection of appropriate constitutive and damage models based on physical relevance and its parameterization.
- Translation of the selected models into a computational format through the development of a user-defined subroutine compatible with finite element software (Abaqus/Explicit).
- Systematic evaluation of the implementation through controlled numerical tests to ensure correct behavior, stability, and reliability of the subroutine.
- Comparison of simulation results with experimental data to evaluate the accuracy of the model and its ability to reproduce observed material responses under varying conditions.

Answering Research Questions

As stated in the introduction, the primary research question seeks to identify the methodological steps necessary for the development and validation of a simulation approach that can reliably predict the material response of flexible targets under impact loading. Although the abovementioned methodology has already outlined these steps in a structured manner, the sub-questions require individual consideration and will be addressed in the sections below.

Sub-Question 1

What existing literature is relevant to the numerical constitutive modeling of flexible targets under impact conditions?

To address this question, the literature review must incorporate several key elements. First, it is essential to understand the functioning of the Abaqus/Explicit solver, with particular attention to its time integration scheme, stability criteria, and the factors that influence numerical stability during dynamic simulations. Second, the review should examine existing approaches to the constitutive modeling of ductile metals, focusing on the formulation of the stress-strain relationship across elastic and plastic regimes. This includes a mathematical description of the material response and an approach to determine the distribution between plastic and elastic strain within the plastic regime.

Sub-Question 2

Which constitutive and damage models are suitable for capturing the dynamic impact behavior of ductile metals, and how can their parameters be determined?

To capture the material response of a flexible target under dynamic impact conditions, the Johnson-Cook constitutive model is a suitable choice due to its ability to incorporate strain rate sensitivity and, where data is available, temperature dependence into the flow stress formulation. This makes it particularly well-suited for high-speed deformation scenarios, such as bird strike simulations or other impact-related analyses.

In addition to a constitutive model, the inclusion of a damage model is essential to accurately capture material failure mechanisms under dynamic loading conditions. In this study, both the Johnson-Cook damage model and a customized Bao-Wierzbicki damage model are considered. Based on the observed behavior and model performance, the customized Bao–Wierzbicki model is found to be more suitable for predicting ductile fracture initiation, as it accounts for key influencing factors such as stress triaxiality and strain rate. These parameters are particularly critical in scenarios involving high strain-rate, multi-axial stress states, making the Bao-Wierzbicki formulation more appropriate for impact simulations.

The parameters of both models are obtained through curve-fitting techniques applied to experimental data. For the Johnson-Cook model, parameters such as yield stress (A), strain hardening coefficients (B and n), and strain rate sensitivity (C) are calibrated using tensile test data at various strain rates. The thermal softening coefficient (m) is omitted in this case due to the absence of temperature-dependent experimental data. Similarly, the Bao-Wierzbicki model parameters (D_1 through D_5) are fitted using failure strain data obtained under varying stress triaxialities and strain rates.

Sub-Question 3

How can a constitutive model be implemented within a numerical simulation framework for dynamic impact events?

To implement the parameterized constitutive and damage models within a numerical simulation framework for dynamic impact events, this study makes use of the finite element software package Abaqus. Abaqus provides the capability to incorporate user-defined material behavior through user subroutines. For high-strain-rate, dynamic simulations - such as those involving bird strikes - the appropriate subroutine is the VUMAT subroutine, which is compatible with the explicit solver in Abaqus. This subroutine allows users to define custom stress-strain relationships and damage evolution laws, making it suitable for implementing advanced material models. The Abaqus documentation provides a baseline Fortran template for the VUMAT subroutine, which can be modified to suit the specific requirements and objectives of the user-defined material model.

Sub-Question 4

How can the numerical implementation of a constitutive model be validated against experimental data?

To validate the numerical implementation of the constitutive and damage models, an exact replica of the test specimen that is featured within the experimental data was constructed within Abaqus. The simulations were performed under loading conditions that precisely matched those of the physical tensile tests, ensuring consistency between experimental and numerical environments. Key response variables, such as force-displacement and stress-strain behavior, were extracted from the simulations and compared against the corresponding experimental measurements obtained via DIC and load cell data.

The validation focused on assessing the model's ability to accurately replicate the material's elastic-plastic response and the moment of failure. Mesh convergence studies were also conducted to ensure the reliability of the simulation results. Discrepancies between numerical and experimental data were analyzed to identify potential sources of error, such as

limitations in material parameterization or experimental measurement uncertainty. This validation approach provided confidence in the fidelity of the implemented constitutive model for simulating dynamic impact behavior of flexible targets.

Recommendations

While this thesis has addressed the main research objectives and presented a validated methodology, several avenues for future work remain. Throughout the course of the study, it became evident that further investigation would be particularly valuable in the following areas.

Improvement of Experimental Data Quality

One key area that would benefit from further investigation is the fidelity of the experimental datasets used for the parameterization of the constitutive and damage models. In this thesis, the strain rate dependency was characterized using only five distinct stress-strain curves, one of which exhibited inconsistencies that limited its reliability. Similarly, the data used to construct the triaxiality versus fracture strain relationship exhibited considerable scatter, making robust curve fitting challenging. To improve the confidence and accuracy of the parameterization process, future work should aim to collect more comprehensive experimental datasets with higher resolution and a greater number of data points for both strain rates and stress states.

Extension to Anisotropic Materials

Another potential direction for future research lies in extending the constitutive material model within its numerical framework. This thesis focuses exclusively on ductile metals, specifically aluminum 2024-T3, which limits the applicability of the current model to a relatively narrow class of materials. To broaden the scope and practical relevance of the simulation framework - particularly for aerospace applications - it would be valuable to develop and implement constitutive and damage models capable of capturing the behavior of anisotropic materials, such as fiber-reinforced composites. These materials are commonly used in modern aircraft structures and exhibit fundamentally different mechanical responses under dynamic loading, necessitating more complex modeling approaches.

Strain-Dependent Formulation of Parameter C

In this thesis, strain rate dependency is captured through parameter C in the Johnson-Cook constitutive equation. Traditionally, C is treated as a constant material property, independent of strain level. However, this simplification overlooks variations in strain rate sensitivity throughout the deformation process. In practice, the curve-fitting procedure yields an average value for C, which may not accurately reflect the material's behavior across different

stages of plastic deformation. This approach fails to capture the evolving nature of material behavior, particularly at higher levels of plastic deformation.

Empirical observations suggest that C increases with rising equivalent plastic strain. This trend aligns with the physical behavior observed during dynamic loading, where plastic strain rate typically increases as the material approaches failure. A more refined modeling approach would involve formulating C as a function of equivalent plastic strain, thereby enhancing the model's ability to reflect strain rate sensitivity more accurately throughout the deformation process.

Further research could explore the integration of this adaptive formulation of C into the numerical simulation framework. One promising direction is the use of tabulated data, where strain rate sensitivity values are defined as a function of equivalent plastic strain and interpolated during simulations. Such an enhancement could improve the predictive capability of the model.

Validation Against Real-World Scenarios

A final recommendation for future work involves validating the numerically implemented material model against real-world impact scenarios. An illustrative example is the experiment described in the thesis of Melvin Post, in which actual bird carcasses were launched at aluminum plates, and the resulting plate deflections were measured both during and after impact. Replicating such experiments through high-fidelity numerical simulations would provide a valuable benchmark for assessing the predictive capability of the developed material and damage models. This type of validation would significantly strengthen confidence in the robustness and applicability of the methodology presented in this thesis, particularly for complex, real-life impact events such as bird strike scenarios in aerospace structures.

References

Abaqus. (n.d.). *1.2.4 VUMAT*. Retrieved from Abaqus Version 6.6 Documentation: https://classes.engineering.wustl.edu/2009/spring/mase5513/abaqus/docs/v6.6/books/sub/default.htm?startat=ch01s02asb04.html

Abaqus. (n.d.). *15.2.1 Smoothed particle hydrodynamics*. Retrieved from Abaqus analysis user's guide: https://ceae-server.colorado.edu/v2016/books/usb/default.htm?startat=pt04ch15s02aus96.html

Abaqus. (n.d.). *9.2 Damping*. Retrieved from Abaqus Version 6.5 Documentation: https://classes.engineering.wustl.edu/2009/spring/mase5513/abaqus/docs/v6.5/books/gss/default.htm?startat=ch09s02.html

Abaqus. (n.d.). *9.2 Explicit Dynamic Finite Element Methods*. Retrieved from Abaqus Version 6.5 Documentation: https://classes.engineering.wustl.edu/2009/spring/mase5513/abaqus/docs/v6.5/books/gsa/default.htm?startat=ch09s02.html

Abaqus. (n.d.). *9.3 Automatic Time Incrementation and Stability*. Retrieved from Abaqus Version 6.5 Documentation: https://classes.engineering.wustl.edu/2009/spring/mase5513/abaqus/docs/v6.5/books/gsa/default.htm?startat=ch09s03.html

American Society for Testing and Materials. (1990). *ASM Handbook, Volume 2 - Properties and Selection: Nonferrous Alloys and Special-Purpose Materials.* ASM International. Retrieved from https://dl.asminternational.org/handbooks/edited-volume/14/Properties-and-Selection-Nonferrous-Alloys-and

American Society for Testing and Materials. (2000). *ASM Handbook, Volume 8 - Mechanical Testing and Evaluation.* Materials Park: ASM International. Retrieved from https://dl.asminternational.org/handbooks/edited-volume/47/Mechanical-Testing-and-Evaluation

Autodesk Support. (2018, June 18). *Autodesk*. Retrieved from Metrics and the Basics of Mechanics - Part 2: https://www.autodesk.com/support/technical/article/caas/tsarticles/ts/2iMjLsg9VOc7z2KALCjdfR.html

Bertholdt, J. (2024). *Methodology to identify and quantify flight path dependent.* Delft: TU Delft. Retrieved from https://repository.tudelft.nl/record/uuid:3e2a8cd9-ac73-47aa-bbd5-d660a5c507ad

Brotak, E. (2018, March 23). *Historynet*. Retrieved from When Birds Strike: https://www.historynet.com/when-birds-strike/

Chen, S. Y., van de Waerdt, W., & Castro, S. G. (2023). *Design for bird strike crashworthiness using a building block approach applied to the Flying-V aircraft.* Heliyon. Retrieved from https://research.tudelft.nl/en/publications/design-for-bird-strike-crashworthiness-using-a-building-block-app-2

COMSOL. (n.d.). *COMSOL Documentation*. Retrieved from Linear Elastic Material: https://doc.comsol.com/5.5/doc/com.comsol.help.sme/sme_ug_theory.06.23.html

COMSOL. (n.d.). *COMSOL Documentation*. Retrieved from Elastic Waves, Time Explicit Model: https://doc.comsol.com/6.2/docserver/#!/com.comsol.help.aco/aco_ug_elastic_waves.06.25.html

EASA. (2023, December 19). *CS-25 Amendment 28.* Retrieved from European Union Aviation Safety Agency: https://www.easa.europa.eu/en/document-library/certification-specifications/cs-25-amendment-28

FEA tips. (2019, March 29). *FEA tips*. Retrieved from Linear vs Quadratic FE Elements: https://featips.com/2019/03/29/linear-vs-quadratic-fe-elements/

Gilioli , A., Manes , A., Giglio, M., & Wierzbicki, T. (2014). *Predicting Ballistic Impact Failure of Aluminium 6061-T6 with the Rate-Independent Bao-Wierzbicki Fracture Model.* Milan: Elsevier. Retrieved from https://www.sciencedirect.com/science/article/pii/S0734743X1400236X

Gordon R. Johnson, W. H. (1983). *A Constitutive Model and Data for Metals Subjected to Large Strains, High Strain Rates and High Temperatures.* The Hague. Retrieved from https://ia800406.us.archive.org/4/items/AConstitutiveModelAndDataForMetals/A%20constitutive%20model%20and%20data%20for%20metals_text.pdf

Guinness World Records. (1936). *Heaviest Flying Bird*. Retrieved from Guinness World Records: https://www.guinnessworldrecords.com/world-records/70907-heaviest-flying-bird

Kelly. (2015, April 23). *8.1 Introduction to Plasticity*. Retrieved from The University of Auckland:

https://pkel015.connect.amazon.auckland.ac.nz/SolidMechanicsBooks/Part_II/08_Plasticity/08_Plasticity_01_Introduction.pdf

Kelly. (2018, February 18). *8.7 Associated and Non-associated Flow Rules*. Retrieved from The University of Auckland: https://pkel015.connect.amazon.auckland.ac.nz/SolidMechanicsBooks/Part_II/08_Plasticity/08_Plasticity_07_Associated_Flow_Rules.pdf

Kouhia, R. (2019, November 13). *Introduction to Materials Modelling.* Retrieved from https://webpages.tuni.fi/rakmek/jmm/slides/jmm_lect_09.pdf

Larsson, H. (2015). *Bird strike analysis.* Stockholm: KTH Royal Institute of Technology. Retrieved from https://www.diva-portal.org/smash/get/diva2:1057190/FULLTEXT01.pdf

LS-DYNA. (2005). *LS-DYNA Support*. Retrieved from Radial Return: https://www.dynasupport.com/tutorial/computational-plasticity/radial-return

Masahiro , N., Satoshi , T., Ziyi , S., Masaki , S., & Masanobu , M. (2023). *Effects of Strain Rate on Stress-Strain Curves in 2024 Aluminum Alloy After Solution Heat Treatment.* Nagoya: The Japan Institute of Light Metals. Retrieved from https://www.jstage.jst.go.jp/article/matertrans/64/2/64_MT-LA2022020/_html/-char/ja

McGinty, B. (2012). *Hooke's Law*. Retrieved from Continuum Mechanics: https://www.continuummechanics.org/hookeslaw2.html

Monaghan, J. J. (1992). *SAO/NASA Astrophysics Data System (ADS).* Retrieved from Smoothed Particle Hydrodynamics: https://articles.adsabs.harvard.edu//full/1992ARA%26A..30..543M/0000543.000.html

Muiruri, A., Maringa, M., & du Preez, W. (2022). *Development of VUMAT and VUHARD Subroutines for Simulating the Dynamic Mechanical Properties of Additively Manufactured Parts.* Basel: MPDI.

Murugesan, M., & Won Jung, D. (2019). *Johnson Cook Material and Failure Model Parameters Estimation of AISI-1045 Medium Carbon Steel for Metal Forming Applications.* Basel: MPDI. Retrieved from https://www.mdpi.com/1996-1944/12/4/609

National Archives and Records Administration. (2025, May 30). *Code of Federal Regulations*. Retrieved from § 25.631 Bird Strike Damage: https://www.ecfr.gov/current/title-14/part-25/section-25.631

New Castle University. (n.d.). Retrieved from Coefficient of Determination, R-squared: https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html

Obbink-Huizer, C. (2025). *Implicit Vs Explicit Finite Element Analysis: When to Use Which*. Retrieved from Technia: https://blog.technia.com/en/simulation/implicit-vs-explicit-finite-element-analysis

Okonkwo, P., & Smith, H. (2016, July 9). *Review of Evolving Trends in Blended Wing Body Aircraft Design.* Bedford: Elsevier. Retrieved from https://www.sciencedirect.com/science/article/pii/S0376042115300336

Post, M. (2024). *Model Validation for Birdstrike Crashworthiness.* Delft: Delft University of Technology. Retrieved from https://repository.tudelft.nl/record/uuid:b1e3504c-53c3-4faf-837c-c9346dbddc97

Rodríguez-Millán, M., Vaz-Romero, Á., & Arias, Á. (2015). *Failure Behavior of 2024-T3 Aluminum under Tension-Torsion Conditions.* Madrid: KSME & Springer. Retrieved from https://link.springer.com/article/10.1007/s12206-015-1011-3

Roylance, D. (n.d.). *1.4: Stress-Strain Curves*. Retrieved from LibreTexts Engineering: https://eng.libretexts.org/Bookshelves/Mechanical_Engineering/Mechanics_of_Materials_(Roylance)/01%3A_Tensile_Response_of_Materials/1.04%3A_Stress-Strain_Curves

Schaefer, M. (2013). *Forecast of Air Traffic's CO2 and NOx Emissions until 2030.* Cologne: German Aerospace Center (DLR). Retrieved from https://www.researchgate.net/publication/259903815_Forecast_of_Air_Traffic%27s_CO2_and_NOx_Emissions_until_2030

Sommavilla, S. (2020, December). *Smoothed Particle Hydrodynamics*. Retrieved from Dive: https://www.divecae.com/resources/sph-basics

Testbook. (2024). *Bisection Method: Definition, Procedure, and Solved Examples*. Retrieved from Testbook: https://testbook.com/maths/bisection-method

The World Bank Group. (2025). *Air Transport, Passengers Carried*. (World Bank Group) Retrieved from World Bank Group: https://data.worldbank.org/indicator/IS.AIR.PSGR?end=2020&start=1970&view=chart

TU Delft. (n.d.). *Flying-V*. Retrieved from TU Delft: https://www.tudelft.nl/lr/flying-v

United Nations - Department of Economic and Social Affairs. (2022). *World Population Prospects 2022.* New York: United Nations Publications. Retrieved from https://www.un.org/development/desa/pd/sites/www.un.org.development.desa.pd/files/wpp2022_summary_of_results.pdf

van Es, G., & Smit, H. (1999). *A Method for Predicting Fatal Bird Strike Rates at Airports.* Ministerie van Verkeer en Waterstaat. Retrieved from https://www2.vlieghinder.nl/naslagdocs/CDrom/REGELS_SCHIPHOL/2.3_TNL/4.3.4.2_A_Method_for_Predicting_Fatal_Bird_Strike_Rates_at.pdf

Wilbeck, J. S. (1978). *Impact Behavior of Low Strength Projectiles.* Wright-Patterson AFB, OH: Airforce Materials Laboratory. Retrieved from https://apps.dtic.mil/sti/tr/pdf/ADA060423.pdf

Zohrevand, M. (2023, July 13). *UMAT & VUMAT: Advantages & Limitations.* Retrieved from Hyper Lyceum: https://hyperlyceum.com/umat-and-vumat-advantages-limitations/

A. Stress-Strain Curve Data Sets

Tensile test at 0.001 mm/s (blue curve)		Tensile test at 0.083 mm/s (green curve)		Tensile test at 1 mm/s (yellow curve)	
ϵ_{true}	$\sigma\ [MPa]$	ϵ_{true}	$\sigma\ [MPa]$	ϵ_{true}	$\sigma\ [MPa]$
0	0	0	0	0.000000	0.000
0.000328	41.962	0.000328	41.962	0.000328	41.962
0.000328	93.460	0.000328	93.460	0.000328	93.460
0.000655	155.450	0.000655	155.450	0.000655	155.450
0.000655	205.995	0.000655	205.995	0.000655	205.995
0.000983	263.215	0.000983	263.215	0.000983	263.215
0.000983	291.826	0.000983	291.826	0.000983	291.826
0.001311	318.529	0.001311	318.529	0.001311	318.529
0.003277	334.741	0.003277	334.741	0.001966	338.556
0.004588	347.139	0.004916	345.232	0.002949	354.768
0.005571	359.537	0.008193	351.907	0.006227	362.398
0.007210	366.213	0.011142	358.583	0.010159	370.027
0.010815	374.796	0.014419	363.351	0.014747	377.657
0.015730	383.379	0.018352	370.027	0.018680	383.379
0.021629	394.823	0.022940	376.703	0.024251	390.054
0.028184	404.360	0.029494	386.240	0.028839	395.777
0.035721	413.896	0.036704	396.730	0.035721	405.313
0.044569	425.341	0.045225	410.082	0.043258	414.850
0.053090	433.924	0.052762	417.711	0.050796	423.433
0.061283	441.553	0.059316	426.294	0.058333	432.970
0.068820	450.136	0.067837	435.831	0.067837	440.599
0.076030	456.812	0.075702	442.507	0.075702	448.229
0.084551	463.488	0.082912	449.183	0.083240	455.858
0.094054	470.163	0.092416	457.766	0.092088	461.580
0.104869	477.793	0.103230	466.349	0.102575	470.163
0.121910	489.237	0.114045	473.978	0.113717	476.839

0.138296	497.820	0.128137	484.469	0.126170	485.422
0.150749	504.496	0.140262	491.144	0.139607	493.052
0.162875	509.264	0.155009	499.728	0.155009	499.728
0.175000	514.986	0.166807	508.311	0.166807	506.403
0.186142	520.708	0.178933	514.033	0.178277	510.218
0.199906	524.523	0.190730	519.755	0.190075	514.033
0.215637	530.245	0.201545	525.477	0.201545	517.847
0.228090	535.014	0.211704	528.338	0.211704	521.662
0.246442	539.782	0.221536	531.199	0.221863	523.569
0.257257	539.782	0.230384	535.014	0.226779	523.569
0.264794	540.736	0.237266	536.921	0.231367	521.662
0.273970	541.689	0.245131	536.921	0.235627	518.801
0.277575	536.921	0.249064	526.431	0.238249	514.986

Tensile test at 10 mm/s (orange curve)		Tensile test at 1000 mm/s (red curve)	
ϵ_{true}	$\sigma\ [MPa]$	ϵ_{true}	$\sigma\ [MPa]$
0	0	0	0
0.000328	41.962	0.000328	41.962
0.000328	93.460	0.000328	93.460
0.000655	155.450	0.000655	155.450
0.000655	205.995	0.000655	205.995
0.000983	263.215	0.000983	263.215
0.000983	291.826	0.000983	291.826
0.001311	318.529	0.001311	318.529
0.001966	336.649	0.001811	341.417
0.004260	344.278	0.002294	352.861
0.007537	352.862	0.004588	359.537
0.011470	359.537	0.008848	367.166
0.015075	365.259	0.014092	371.935
0.019007	370.027	0.019007	375.749
0.023923	377.657	0.023596	380.518
0.029494	386.240	0.029494	386.240
0.035393	393.869	0.035066	391.962
0.042275	400.545	0.042275	400.545
0.048830	407.221	0.048830	407.221
0.057022	414.850	0.057022	414.850
0.065871	422.480	0.065871	422.480
0.075702	431.063	0.075702	431.063
0.085206	439.646	0.085206	439.646
0.094710	448.229	0.094710	448.229
0.104869	455.858	0.104869	455.858
0.116011	463.488	0.116011	463.488
0.125843	470.163	0.127809	472.071
0.135674	476.839	0.139279	481.608
0.152388	484.469	0.152388	490.191
0.165824	492.098	0.169101	499.728
0.177294	494.005	0.183521	508.311
0.188436	497.820	0.199579	515.940

0.197940	501.635	0.214654	524.523
0.204822	504.496	0.229073	531.199
0.210721	505.450	0.245787	538.828
0.217275	506.403	0.262500	542.643
0.222846	506.403	0.273970	546.458
0.227434	506.403	0.288062	548.365
0.231039	505.450	0.293633	541.689

B.Stress Triaxiality - Fracture Strain Data Set

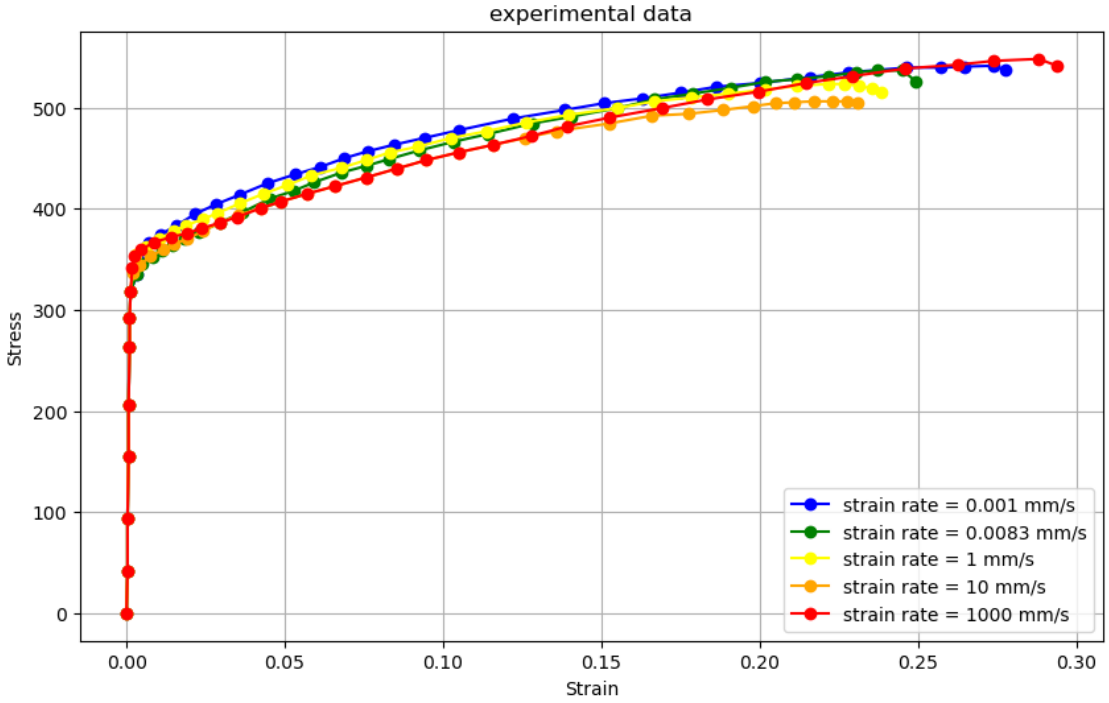
η	$\epsilon_{fracture}$
0.0871	0.1717
0.2106	0.2262
0.2846	0.2486
0.3932	0.2692
0.4943	0.2566
0.5411	0.2745
0.5991	0.2584
0.5973	0.2709
0.4943	0.2262
0.6600	0.2772
0.6900	0.3282
0.7049	0.2611
0.7555	0.2602
0.7883	0.2915
0.8594	0.2548
0.9427	0.2137
0.9792	0.2307

C. Python Script for Parameterization Johnson-Cook Model

C.1. Configuring the Datasets

```
#loading and plotting curve datasets
#importing relevant libraries
import numpy as np
import matplotlib.pyplot as plt
import os
from scipy.optimize import curve_fit
import math
from scipy.interpolate import interp1d

# Define the directory where the files are located
directory = r'D:\OneDrive - Delft University of
Technology\thesis\literature study\material&damage modelling\stress-
straincurves_frompaper'

# Define file names (without extension) and colors
file_names = ['eps_vs_sig_blue', 'eps_vs_sig_green', 'eps_vs_sig_yellow',
'eps_vs_sig_orange', 'eps_vs_sig_red']
colors = ['blue', 'green', 'yellow', 'orange', 'red', 'purple']
labels = ['strain rate = 0.001 mm/s', 'strain rate = 0.0083 mm/s', 'strain
rate = 1 mm/s', 'strain rate = 10 mm/s', 'strain rate = 1000 mm/s']

# Initialize the plot
plt.figure(figsize=(10, 6))

#empty list for storing data for further analysis
data = []

# Check if the file exists
if os.path.exists(directory):
    # Plot
    for i, file_name in enumerate(file_names):
        # Construct the full file path
        file_path = os.path.join(directory, f'{file_name}.txt')

        # Load the data using numpy, expecting two columns (x, y)
        data_curve = np.loadtxt(file_path)
        data.append(data_curve)
```

```
            # Separate x and y values
            x = data_curve[:, 0]
            y = data_curve[:, 1]

            # Dynamically create a list name, e.g., data_blue_curve
            exec(f"data_{colors[i]}_curve = list(zip(x, y))")

            # Plot the data
            plt.plot(x, y, label=labels[i], color=colors[i], marker = 'o')
        plt.xlabel('Strain')
        plt.ylabel('Stress')
        plt.title('experimental data' )
        plt.grid(True)
        plt.legend()
        plt.show()
else:
        print(f"File '{file_name}' not found in directory '{directory}'")
```

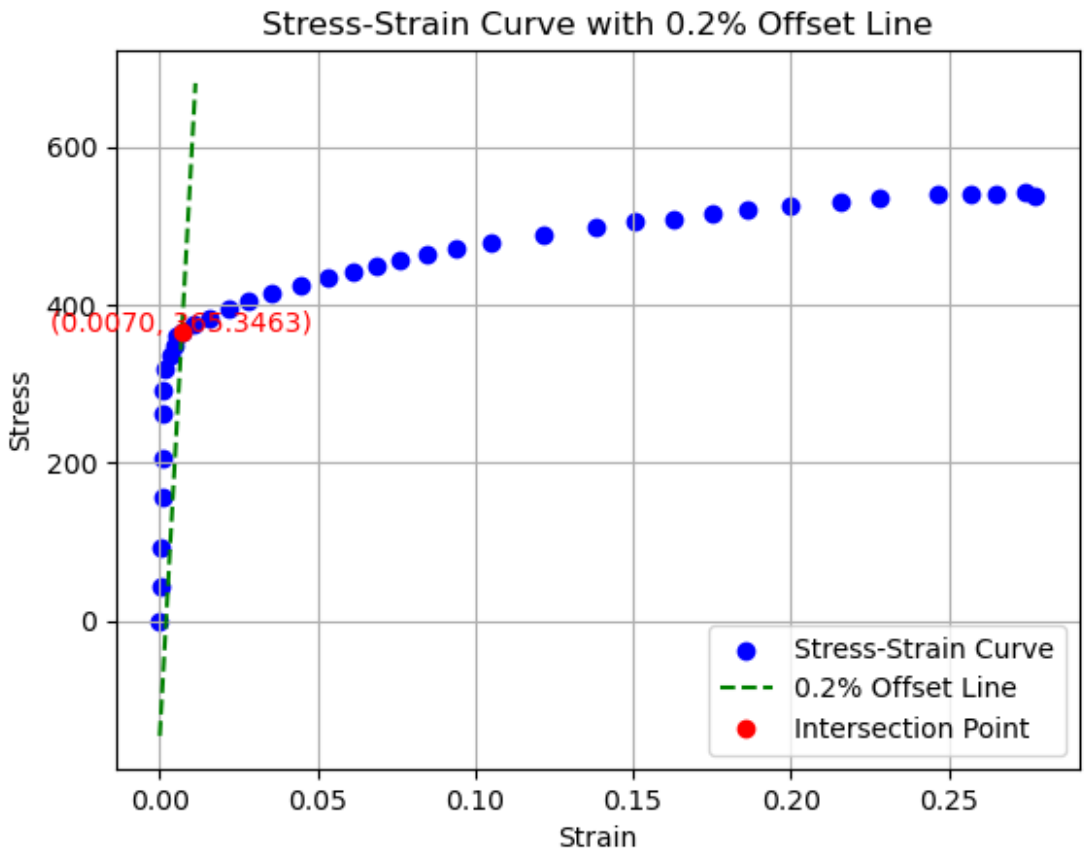


C.2. Determining Parameter A

```
#Determining JC model parameter A
def straight_line(E, x, b):
    return E * x + b

def find_intersection(x1, y1, x2, y2):
    # Ensure inputs are numpy arrays for easier manipulation
    x1, y1, x2, y2 = map(np.array, (x1, y1, x2, y2))

    # Interpolate both curves
    f1 = interp1d(x1, y1, kind='linear', fill_value='extrapolate')
    f2 = interp1d(x2, y2, kind='linear', fill_value='extrapolate')
```

```
    # Create a common x-axis range for comparison
    x_common = np.linspace(max(min(x1), min(x2)), min(max(x1), max(x2)),
1000)

    # Compute the difference between the two curves
    diff = f1(x_common) - f2(x_common)

    # Find the zero crossings (sign change points)
    sign_changes = np.where(np.diff(np.sign(diff)))[0]

    # Calculate intersection points
    intersections = []
    for idx in sign_changes:
        # Interpolate for a more precise intersection
        x_intersect = x_common[idx]
        y_intersect = f1(x_intersect)
        intersections.append((x_intersect, y_intersect))

    return intersections

# Separate strain and stress data
strain = np.array([point[0] for point in data_blue_curve])
stress = np.array([point[1] for point in data_blue_curve])

# Youngs modulus
E = 73100

# 0.2% offset method to calculate A (yield stress)
offset_line = E * (strain + 0.002)  # Compute offset line for the entire
strain range

# Generate straight line for elastic region
x_straight_line = np.linspace(min(strain_elastic),
max(strain_elastic+0.01), 100)
y_straight_line = straight_line(E, x_straight_line, -E * 0.002)

#employ intersection function to find yield stress strain
intersection = find_intersection(strain, stress, x_straight_line,
y_straight_line)
eps_yield, sig_yield = intersection[0]

# Plot the data
plt.scatter(strain, stress, color='blue', label='Stress-Strain Curve')
plt.plot(x_straight_line, y_straight_line, color='green', linestyle='--',
label='0.2% Offset Line')

# Highlight intersection points
for x, y in intersection:
    plt.plot(x, y, 'ro', label='Intersection Point' if 'Intersection Point'
not in plt.gca().get_legend_handles_labels()[1] else "")
    plt.text(x, y, f'({x:.4f}, {y:.4f})', color='red', fontsize=10,
ha='center')

# Add labels, legend, and grid
```

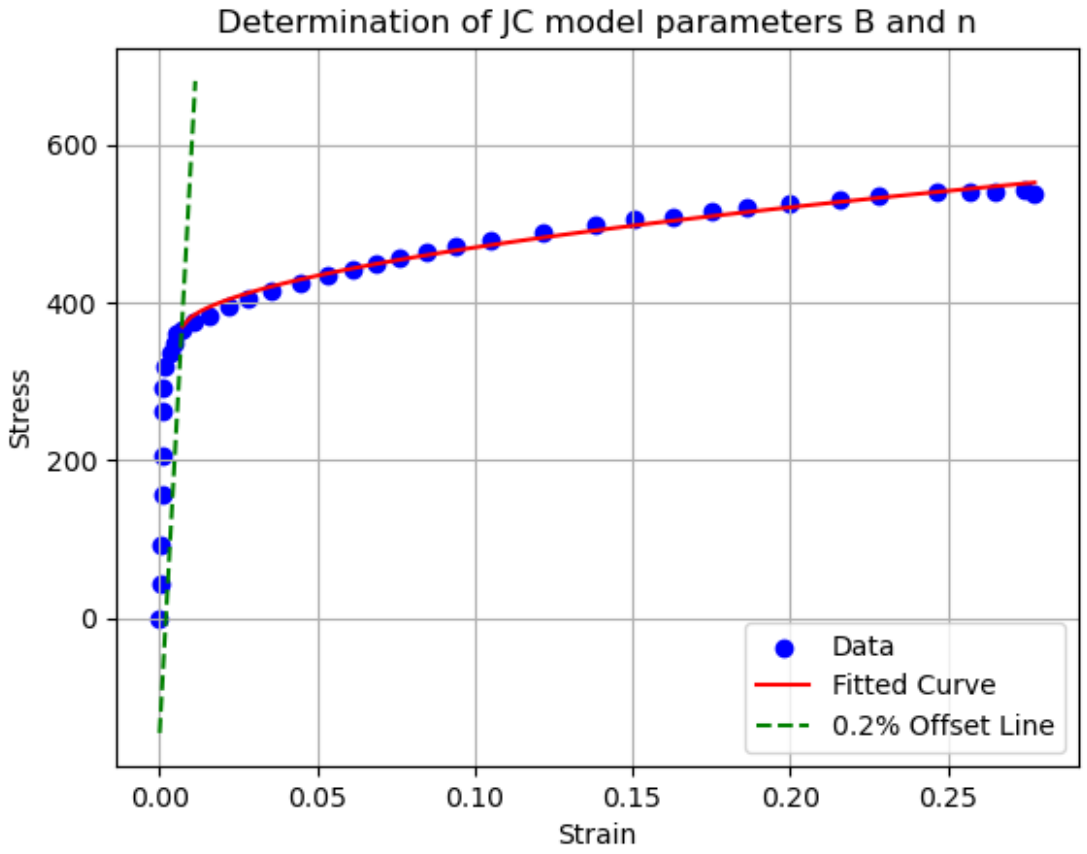
```
plt.xlabel('Strain')
plt.ylabel('Stress')
plt.legend()
plt.grid()
plt.title('Stress-Strain Curve with 0.2% Offset Line')
plt.show()
```



C.3. Determining Parameters B and n

```
#Determining JC model parameters B and n
# Johnson-Cook equation for strain rate effect
def johnson_cook_stress(strain, A, B, n):
    return A + B * strain ** n

# Plastic region: the data after yielding
plastic_region = strain > 0.007
strain_plastic = strain[plastic_region] - eps_yield
stress_plastic = stress[plastic_region]

# Initial guess for B and n
A_blue = sig_yield
initial_guess = [400, 0.6]

# Fit the Johnson-Cook model to the plastic region data, keeping A constant
popt, pcov = curve_fit(
    lambda x, B, n: johnson_cook_stress(x, A_blue, B, n),
    strain_plastic,
```

```
    stress_plastic,
    p0=initial_guess,
    maxfev=10000
)

# Extract B and n
B_blue, n_blue = popt

# Generate fitted curve
x_fit_plastic = np.linspace(min(strain_plastic), max(strain_plastic), 100)
x_fit = x_fit_plastic + eps_yield
y_fit = johnson_cook_stress(x_fit_plastic, A_blue, B_blue, n_blue)



# Plot the data and the fitted curve
plt.scatter(strain, stress, color='blue', label='Data')
plt.plot(x_fit, y_fit, color='red', label='Fitted Curve')
plt.plot(x_straight_line, y_straight_line, color='green', linestyle='--',
label='0.2% Offset Line')
plt.xlabel('Strain')
plt.ylabel('Stress')
plt.legend()
plt.grid()
plt.title("Determination of JC model parameters B and n")
plt.show()

print(f"Fitted parameter: B = {B_blue}")
print(f"Fitted parameter: n = {n_blue}")
```

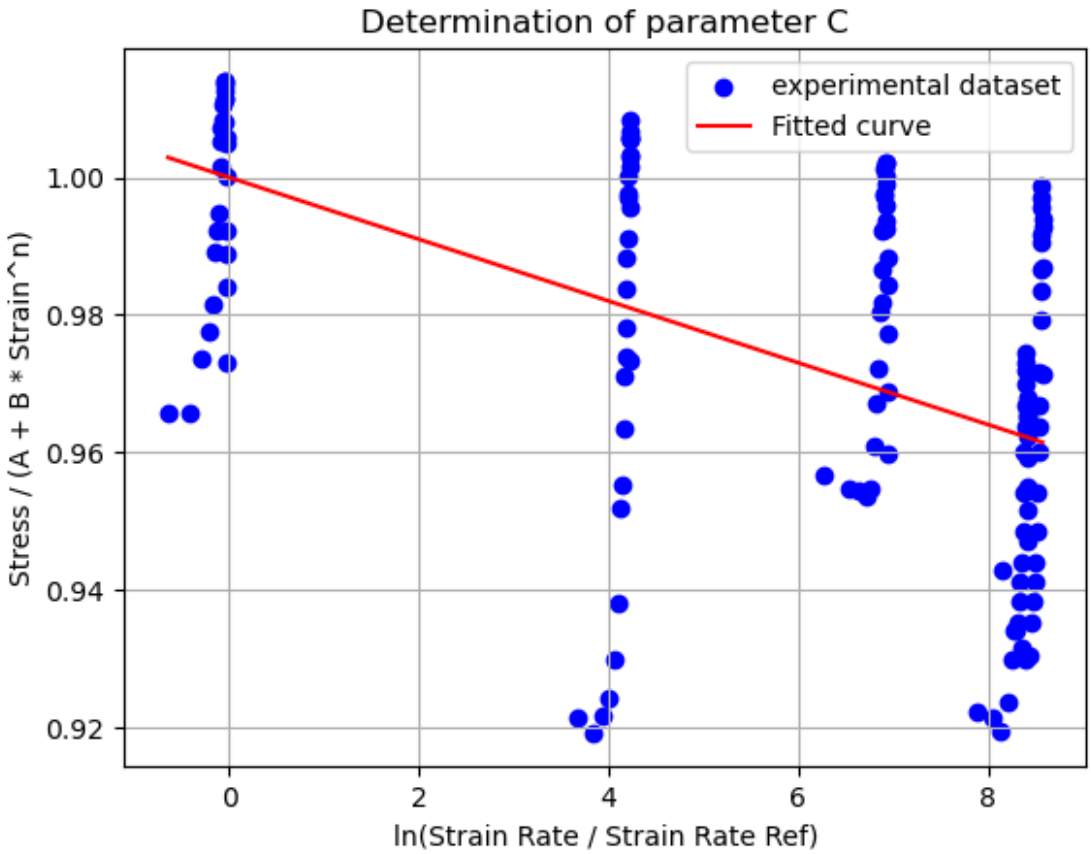



```
Fitted parameter: B = 380.3780416667515
Fitted parameter: n = 0.545031135555984
```

C.4. Determining Parameter C

```
#finding johnson cook material model parameter C
# Given parameters from the quasi-static test (blue curve)
A = A_blue
B = B_blue
n = n_blue

strain_rate_ref = strain_rate_blue = 1.7*10**-4 #s^-1
strain_rate_green = 1.2*10**-2 #s^-1
strain_rate_yellow = 1.8*10**-1 #s^-1
strain_rate_orange = 7.9*10**-1 #s^-1
strain_rate_red = 9.2*10**-1 #s^-1


#strain rate data
strain_rate_data = np.array([strain_rate_blue, strain_rate_green,
strain_rate_yellow, strain_rate_orange, strain_rate_red])  # Example
strain rates

#create lists for plotting data
strain = []
stress = []
strain_rate = []
```

```
for i in range(len(strain_rate_data)):
    for j in range(len(data[i])):
        strain_rate.append(strain_rate_data[i])
        strain.append(data[i][j][0])
        stress.append(data[i][j][1])

#transform lists to numpy array
strain = np.array(strain)
stress = np.array(stress)
strain_rate = np.array(strain_rate)

#reduce the arrays to the plastic region only
plastic_region = strain > 0.01
plastic_stress = stress[plastic_region]
elastic_strain = plastic_stress/E
plastic_strain = strain[plastic_region]-elastic_strain
ratio_plas_strain =plastic_strain/strain[plastic_region]
strain_rate = strain_rate[plastic_region]*ratio_plas_strain

# Calculate data for the graph
x_data = np.log(np.array(strain_rate) / strain_rate_ref)
y_data = plastic_stress / (A + B * plastic_strain ** n)

# Define linear model function to fit y = 1 + C * x
def linear_model(x, C):
    return 1 + C * x

# Fit the linear model to the transformed data
popt, pcov = curve_fit(linear_model, x_data, y_data)
C_fitted = popt[0]

# Generate fitted curve for plotting
x_fit = np.linspace(min(x_data), max(x_data), 100)
y_fit = linear_model(x_fit, C_fitted)

# Plot the data and the fitted line
plt.scatter(x_data, y_data, color='blue', label='experimental dataset')
plt.plot(x_fit, y_fit, color='red', label='Fitted curve')
plt.xlabel('ln(Strain Rate / Strain Rate Ref)')
plt.ylabel('Stress / (A + B * Strain^n)')
plt.legend()
plt.grid()
plt.title("Determination of parameter C")
plt.show()

print(f"Fitted parameter: C = {C_fitted}")
```

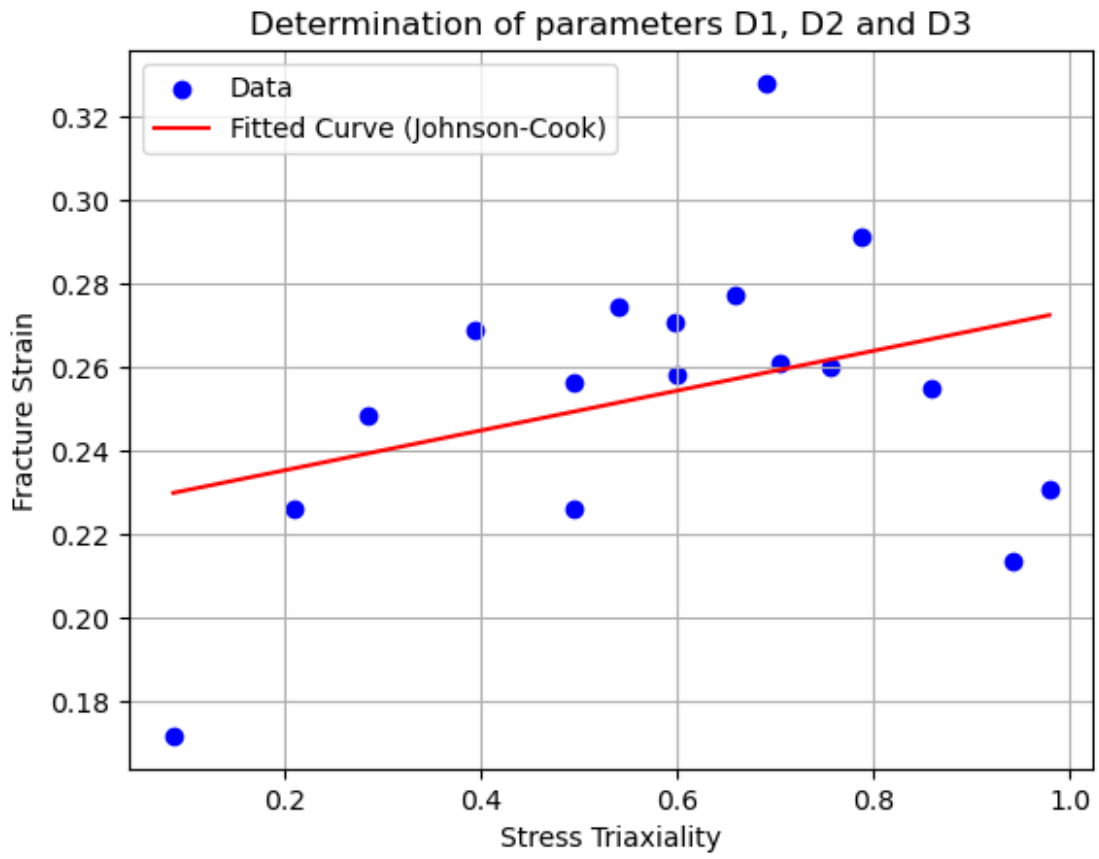



```
Fitted parameter: C = -0.004494194034623321
```

C.5. Determining Damage Model Parameters D1, D2 and D3

```
#calculation of damage parameters D1, D2, D3
#file directories
directory = r"D:\OneDrive - Delft University of
Technology\thesis\literature study\material&damage modelling\triaxiality-
vs-fracturestraincurve"
file_name = "triaxiality-vs-fracturestraincurve.txt"

# Construct the full file path
file_path = os.path.join(directory, file_name)

# Load the data
data_curve = np.loadtxt(file_path)

# Separate x and y values
triaxiality_data = data_curve[:, 0]
fracturestrain_data = data_curve[:, 1]

# Define the Johnson Cook damage model function
def JC_D1D2D3(triaxiality, D1, D2, D3):
    ef = D1 + D2 * np.e**(D3*triaxiality)
    return ef

#curve fitting for D1, D2 and D3
popt, pcov = curve_fit(JC_D1D2D3, triaxiality_data, fracturestrain_data,
```

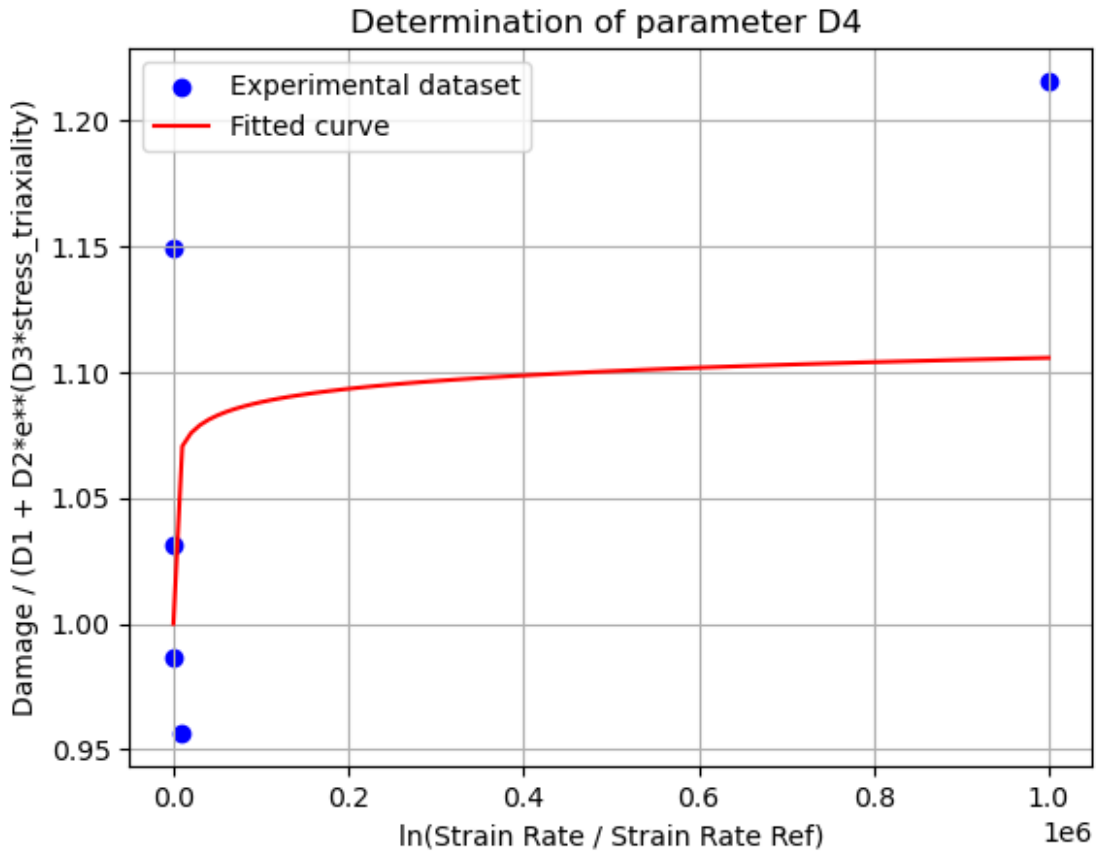
```
maxfev=50000)

# Retrieve the fitted parameters
D1, D2, D3 = popt

# Generate fitted curve for plotting
x_fit = np.linspace(min(triaxiality_data), max(triaxiality_data), 100)
y_fit = JC_D1D2D3(x_fit, D1, D2, D3)

# Plot the data and the fitted curve
plt.scatter(triaxiality_data, fracturestrain_data, color='blue',
label='Data')
plt.plot(x_fit, y_fit, color='red', label='Fitted Curve (Johnson-Cook)')
plt.xlabel('Stress Triaxiality')
plt.ylabel('Fracture Strain')
plt.legend()
plt.grid()
plt.title("Determination of parameters D1, D2 and D3")
plt.show()

print(f"Fitted parameters: D1 = {D1}, D2 = {D2}, D3 = {D3}")
```




```
Fitted parameters: D1 = -132.7438252565593, D2 = 132.9696080490151,
D3 = 0.00035914687671983255
```

C.6. Determining Damage Model Parameter D4

```
# Finding the damage parameters D4
# Fracture strain values from the end points of respective curves
damage_data = np.array([
    data_blue_curve[-1][0],
    data_green_curve[-1][0],
    data_yellow_curve[-1][0],
    data_orange_curve[-1][0],
    data_red_curve[-1][0]
])

# Strain rate data, excluding the outlier
strain_rate_data_d4 = strain_rate_data#[:-1]

# Reference stress triaxiality
stress_triaxiality = 0.33  # Given stress triaxiality value for tensile
loaded specimen

# Normalize strain rate data (x_data) and damage data (y_data)
x_data = strain_rate_data_d4 / strain_rate_ref
y_data = damage_data / (D1 + D2 * np.e**(D3*stress_triaxiality))

# Define strain rate dependent function
def JC_D4(x, D4):
    return 1 + D4 * np.log(x)

# Fit the extended linear model to the transformed data
popt, pcov = curve_fit(JC_D4, x_data, y_data, maxfev=50000)

# Extract the fitted parameter
D4_fitted = popt

# Generate fitted curve data for plotting
x_fit = np.linspace(min(x_data), max(x_data), 100)
y_fit = JC_D4(x_fit, D4_fitted)

# Plot the experimental data and the fitted curve
plt.scatter(x_data, y_data, color='blue', label='Experimental dataset')
plt.plot(x_fit, y_fit, color='red', label='Fitted curve')
plt.xlabel('ln(Strain Rate / Strain Rate Ref)')
plt.ylabel('Damage / (D1 + D2*e**(D3*stress_triaxiality)')
plt.legend()
plt.grid()
plt.title("Determination of parameter D4 ")
plt.show()

# Print the fitted parameters
print(f"Fitted parameters: D4 = {D4_fitted}")
```

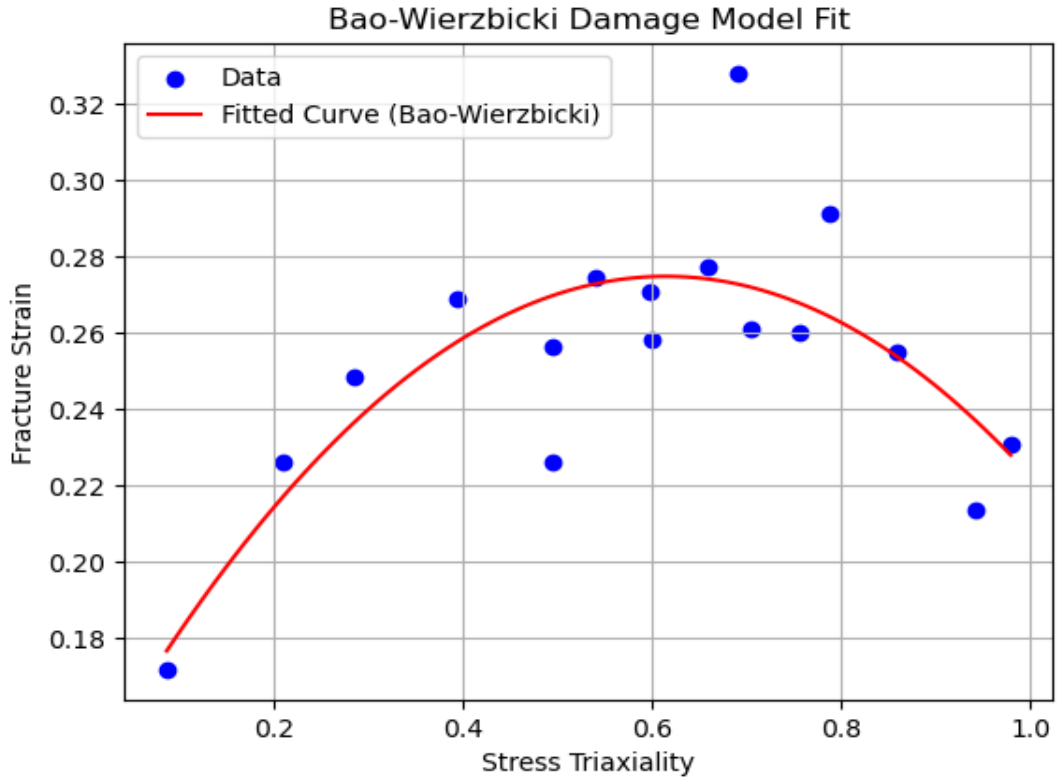



```
Fitted parameters: D4 = [0.0076463]
```

C.7. Determining Damage Model Parameters D1, D2 and D3 - updated

```
#calculation of damage parameters D1, D2, D3
#file directories
directory = r"D:\OneDrive - Delft University of
Technology\thesis\literature study\material&damage modelling\triaxiality-
vs-fracturestraincurve"
file_name = "triaxiality-vs-fracturestraincurve.txt"

# Construct the full file path
file_path = os.path.join(directory, file_name)

# Load the data
data_curve = np.loadtxt(file_path)

# Separate x and y values
triaxiality_data = data_curve[:, 0]
fracturestrain_data = data_curve[:, 1]

# Define the Bao-Wierzbicki damage model function
def bao_wierzbicki_damage(triaxiality, d1, d2, d3):
    ef = d1 + d2 * triaxiality + d3 * triaxiality**2
    return ef
```

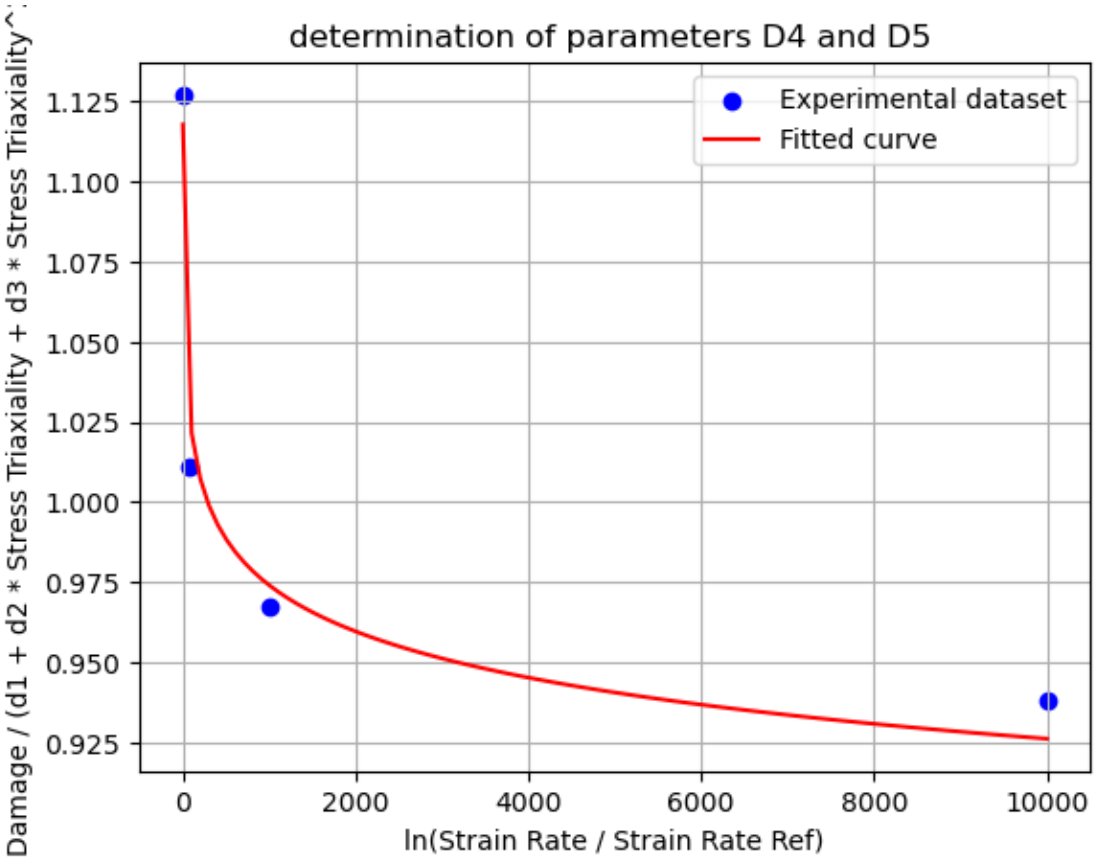
```
popt, pcov = curve_fit(bao_wierzbicki_damage, triaxiality_data,
fracturestrain_data, maxfev=50000)

# Retrieve the fitted parameters
d1, d2, d3 = popt

# Generate fitted curve for plotting
x_fit = np.linspace(min(triaxiality_data), max(triaxiality_data), 100)
y_fit = bao_wierzbicki_damage(x_fit, d1, d2, d3)

# Plot the data and the fitted curve
plt.scatter(triaxiality_data, fracturestrain_data, color='blue',
label='Data')
plt.plot(x_fit, y_fit, color='red', label='Fitted Curve (Bao-Wierzbicki)')
plt.xlabel('Stress Triaxiality')
plt.ylabel('Fracture Strain')
plt.legend()
plt.grid()
plt.title("determination of parameters D1, D2 and D3")
plt.show()

print(f"Fitted parameters: d1 = {d1}, d2 = {d2}, d3 = {d3}")
```




```
Fitted parameters: d1 = 0.14149148940242626, d2 = 0.4339760367954482, d3 =
-0.352962243979643
```

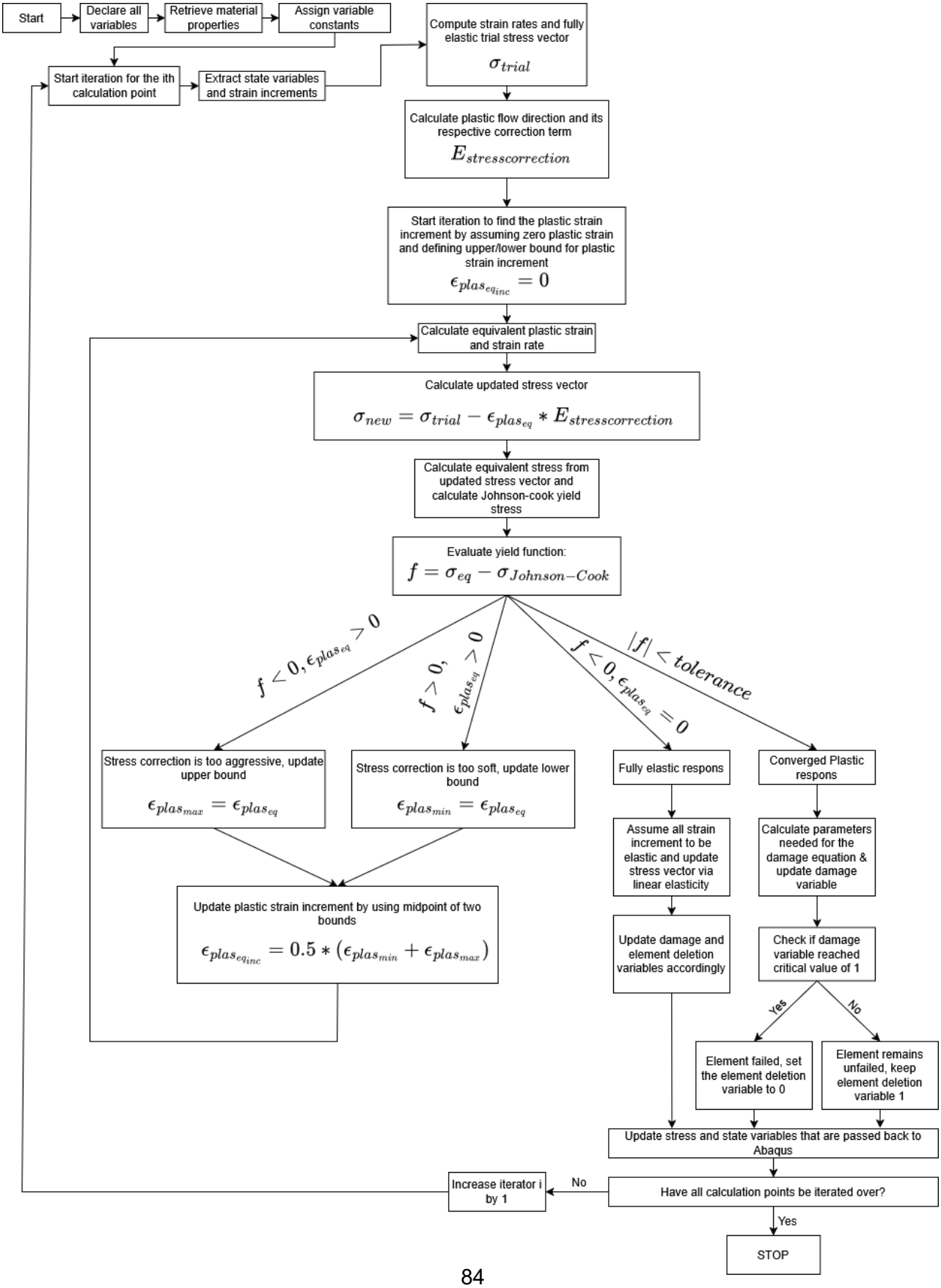
C.8. Determining Damage Model Parameters D4 and D5 - updated

```
# Finding the damage parameters d4 and d5
# Fracture strain values from the end points of respective curves
damage_data = np.array([
    data_blue_curve[-1][0],
    data_green_curve[-1][0],
    data_yellow_curve[-1][0],
    data_orange_curve[-1][0]
])

# Strain rate data for damage parameters, excluding the outlier
strain_rate_data_d4 = strain_rate_data[:-1]

# Reference strain rate (assumed quasi-static) and stress triaxiality
stress_triaxiality = 0.33  # Given stress triaxiality value for tensile
loaded specimen

# Normalize strain rate data (x_data) and damage data (y_data)
x_data = strain_rate_data_d4 / strain_rate_ref
y_data = damage_data / (d1 + d2 * stress_triaxiality + d3 *
stress_triaxiality**2)

# Define the extended linear model: y = d4 * ln(d5 * x)
def linear_model(x, d4, d5):
    return 1 + d4 * np.log(d5 * x)

# Fit the extended linear model to the transformed data
popt, pcov = curve_fit(linear_model, x_data, y_data, maxfev=50000)

# Extract the fitted parameters
d4_fitted, d5_fitted = popt

# Generate fitted curve data for plotting
x_fit = np.linspace(min(x_data), max(x_data), 100)
y_fit = linear_model(x_fit, d4_fitted, d5_fitted)

# Plot the experimental data and the fitted curve
plt.scatter(x_data, y_data, color='blue', label='Experimental dataset')
plt.plot(x_fit, y_fit, color='red', label='Fitted curve')
plt.xlabel('ln(Strain Rate / Strain Rate Ref)')
plt.ylabel('Damage / (d1 + d2 * Stress Triaxiality + d3 * Stress
Triaxiality^2)')
plt.legend()
plt.grid()
plt.title("determination of parameters D4 and D5 ")
plt.show()

# Print the fitted parameters
print(f"Fitted parameters: d4 = {d4_fitted}, d5 = {d5_fitted}")
```



Fitted parameters: d4 = -0.02081453900375374, d5 = 0.0034684695061959403

D. VUMAT Flowchart



E. VUMAT Johnson-Cook

```
subroutine vumat(nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,&
      stepTime, totalTime, dt, cmname, coordMp, charLength,&
      props, density, strainInc, relSpinInc,&
      tempOld, stretchOld, defgradOld, fieldOld,&
      stressOld, stateOld, enerInternOld, enerInelasOld,&
      tempNew, stretchNew, defgradNew, fieldNew,&
      ! Write only (modifiable) variables -
      stressNew, stateNew, enerInternNew, enerInelasNew )

      implicit none     !no assumptions can be made about variable types

      !double variables
      double precision :: props(nprops), tempOld(nblock), tempNew(nblock)
      double precision :: stateOld(nblock,nstatev), strainInc(nblock,ndir+nshr), stressOld(nblock,ndir+nshr)
      double precision :: stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev)
      double precision :: coordMp(nblock,*), relSpinInc(nblock,nshr), stretchOld(nblock,ndir+nshr)
      double precision :: defgradOld(nblock,ndir+nshr+nshr), fieldOld(nblock,nfieldv)
      double precision :: stretchNew(nblock,ndir+nshr), defgradNew(nblock,ndir+nshr+nshr), fieldNew(nblock,nfieldv)
      double precision :: enerInternOld(nblock), enerInelasOld(nblock), enerInternNew(nblock), enerInelasNew(nblock)
      double precision :: dt, stepTime, totalTime, density, charLength(nblock)
      double precision :: A, B, n, C, m, ref_ep_dot, Tm, Tr, E, G, T_star, e_dot_star
      double precision :: D1, D2, D3, D4, D5
      double precision :: e_tot(ndir+nshr)
      double precision :: e_inc(ndir+nshr), e_dot(ndir+nshr)
      double precision :: e_plas_inc_eq, e_plas_inc_eq_old
      double precision :: e_plas_tot_eq, e_plas_tot_new_eq, e_plas_dot_eq , e_plas_inc_min, e_plas_inc_max
      double precision :: sig_old(ndir+nshr), sig_new(ndir+nshr), sig_eq, sig_johnson_cook, sig_johnson_cook_old
      double precision :: sig_deviatoric(ndir+nshr), plastic_flow_dir(ndir+nshr), stress_correction_term(ndir+nshr)
      double precision ::  triaxiality, e_failure, damage_old, damage_new, element_deletion_new
      double precision :: sig_trial(ndir+nshr), C_mat(6,6), error, sig_hydrostatic
      double precision :: tolerance, lambda_lame, mu_lame, nu

      ! Integer variables
      integer :: i, nblock, nstatev, nfieldv, nprops, lanneal
      integer :: ndir, nshr
      integer :: iter, max_iter, debug_iter

      !string variables
      CHARACTER(len=*) :: cmname

      !material related input parameters
      A = props(1)            !Johnson cook material parameter
      B = props(2)            !idem
      n = props(3)            !idem
      C = props(4)            !idem
      m = props(5)            !idem
      ref_ep_dot = props(6)   !reference strain rate
      Tm = props(7)           !melting temperature
      Tr = props(8)           !reference temperature
      E = props(9)            !youngs modulus
      G = props(10)           !shear modulus
      nu = props(11)          !poisson ratio
      D1 = props(12)          !Damage model parameter
      D2 = props(13)          !idem
      D3 = props(14)          !idem
      D4 = props(15)          !idem
      D5 = props(16)          !idem

      !elasticity matrix C
      lambda_lame = E*nu/((1+nu) * (1-2*nu))
      mu_lame = E/(2*(1+nu))
      C_mat(1, 1:6) = [lambda_lame + 2 * mu_lame, lambda_lame, lambda_lame, 0.0d0, 0.0d0, 0.0d0]
      C_mat(2, 1:6) = [lambda_lame, lambda_lame + 2 * mu_lame, lambda_lame, 0.0d0, 0.0d0, 0.0d0]
      C_mat(3, 1:6) = [lambda_lame, lambda_lame, lambda_lame + 2 * mu_lame, 0.0d0, 0.0d0, 0.0d0]
      C_mat(4, 1:6) = [0.0d0, 0.0d0, 0.0d0, mu_lame, 0.0d0, 0.0d0]
      C_mat(5, 1:6) = [0.0d0, 0.0d0, 0.0d0, 0.0d0, mu_lame, 0.0d0]
      C_mat(6, 1:6) = [0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, mu_lame]

      !Newton-raphson iteration parameters
      max_iter = 100
      tolerance = 1.0E-9

      !Looping over all the integration points
      DO i = 1, nblock
```

```
!updating state variables
e_tot = stateOld(i, 1:6)
e_plas_inc_eq_old = stateOld(i, 7)
e_plas_tot_eq = stateOld(i, 8)
sig_johnson_cook_old = stateOld(i, 9)
damage_old = stateOld(i, 10)
element_deletion_new = stateOld(i,11)
IF (stepTime == 0.0) then
      element_deletion_new = 1.0
END IF

!strain increment vector
e_inc = strainInc(i, 1:6)

!strain rate vector
e_dot = e_inc / dt

!stress vector at previous increment
sig_old = stressOld(i, 1:6)

sig_trial = sig_old + matmul(C_mat, e_inc)  ! Compute trial stress using the elastic modulus matrix

! Compute hydrostatic & deviatoric stress vector
sig_hydrostatic = sum(sig_trial(1:ndir)) / 3.0
sig_deviatoric(1:ndir) = sig_trial(1:ndir) - sig_hydrostatic
sig_deviatoric(ndir+1:ndir+nshr) = sig_trial(ndir+1:ndir+nshr)
sig_eq = sqrt(sig_deviatoric(1)**2 + sig_deviatoric(2)**2 + &
      sig_deviatoric(3)**2 + sig_deviatoric(4)**2 + sig_deviatoric(5)**2 + sig_deviatoric(6)**2)

if (sig_eq > 0.0d0) then
      plastic_flow_dir = (3.0*sig_deviatoric) / (2.0*sig_eq) ! Compute plastic flow direction
else
      plastic_flow_dir = 0.0d0
end if

stress_correction_term = matmul(C_mat, plastic_flow_dir)

e_plas_inc_eq = 0.0             ! Initial plastic strain increment
e_plas_inc_min = 0.0            ! Lower bound for plastic strain increment
e_plas_inc_max = sig_eq / (2.0d0 * G)  ! Upper bound

iter = 0
do while (iter < max_iter)
      iter = iter + 1
      if (iter > max_iter - 1) then
            debug_iter = debug_iter + 1
            print*, "Error: Too many iterations, the effective produced error remains at = ", error
      end if

      ! Compute updated plastic strain parameters
      e_plas_dot_eq = e_plas_inc_eq / dt
      e_plas_tot_new_eq = e_plas_tot_eq + e_plas_inc_eq

      ! Compute updated elastic stress respons
      sig_new = sig_trial - e_plas_inc_eq*stress_correction_term
      sig_hydrostatic = sum(sig_new(1:ndir)) / 3.0
      sig_deviatoric(1:ndir) = sig_new(1:ndir) - sig_hydrostatic
      sig_deviatoric(ndir+1:ndir+nshr) = sig_new(ndir+1:ndir+nshr)
      sig_eq = sqrt((1.0/2.0)*((sig_new(1)-sig_new(2))**2 + (sig_new(2)-sig_new(3))**2 + &
            (sig_new(3)-sig_new(1))**2) + 3.0*(sig_new(4)**2 + sig_new(5)**2 + sig_new(6)**2))

      !Compute Johnson-Cook yield stress
      tempOld(i) = Tr
      T_star = (tempOld(i) - Tr) / (Tm - Tr)
      e_dot_star = e_plas_dot_eq / ref_ep_dot
      if (e_dot_star < 1.0d0) then
            e_dot_star = 1.0d0
      end if
      sig_johnson_cook = (A + B * (e_plas_tot_new_eq**n)) * (1.0d0 + C * log(e_dot_star)) &
      * (1.0d0 - T_star**m)

      ! Ensure yield stress does not decrease
      if (sig_johnson_cook < sig_johnson_cook_old) then
            sig_johnson_cook = sig_johnson_cook_old
      end if
```

```
151
152                 !yield function
153                 error = sig_eq - sig_johnson_cook
154
155                 !Elastic respons exit
156                 if (e_plas_inc_eq == 0.0d0 .and. error < 0.0d0) then
157                     !debug_iter = debug_iter + 1
158                     sig_new = sig_old + matmul(C_mat, e_inc)
159                     damage_new = damage_old
160                     element_deletion_new = 1.0
161                     exit
162                 end if
163
164                 !plastic respons exit
165                 if (abs(error) < tolerance) then
166                     debug_iter = debug_iter + 1
167                     sig_hydrostatic = sum(sig_new(1:ndir)) / 3.0
168                     triaxiality = sig_hydrostatic/sig_eq
169                     e_failure = (D1 + D2*triaxiality + D3*triaxiality**2.0)* (1.0 + D4*log(D5*(e_plas_dot_eq / ref_ep_dot)))
170                     damage_new = damage_old + sqrt((e_plas_inc_eq/e_failure)**2)
171                     !determine whether failure is reached (deactivate element from the model)
172                     IF (damage_new >= 1.0) then
173                         element_deletion_new = 0.0
174                         print*, "element failed at steptime:", stepTime
175                     ELSE IF (damage_new < 1.0) then
176                         element_deletion_new = 1.0
177                     END IF
178                     exit
179                 end if
180
181                 ! Update the bounds for undershooting
182                 if (error >= 0.0d0 .and. e_plas_inc_eq /= 0.0d0) then
183                     e_plas_inc_min = e_plas_inc_eq
184                 end if
185
186                 !update the bounds for overshooting
187                 if (error < 0.0 .and. e_plas_inc_eq /= 0.0d0) then
188                     e_plas_inc_max = e_plas_inc_eq
189                 end if
190
191                 ! Update plastic strain increment using the midpoint of bounds
192                 e_plas_inc_eq = 0.5d0 * (e_plas_inc_max + e_plas_inc_min)
193             end do
194
195             !stress update
196             stressNew(i, 1:6) = sig_new
197
198             ! Update state variables
199             stateNew(i, 1:6) = e_tot + e_inc
200             stateNew(i, 7) = e_plas_inc_eq
201             stateNew(i, 8) = e_plas_tot_new_eq
202             stateNew(i, 9) = sig_johnson_cook
203             stateNew(i, 10) = damage_new
204             stateNew(i, 11) = element_deletion_new
205         END DO
206         return
207     end subroutine vumat
208
209
210
211
212
213
```

F. Dummy Program

```
PROGRAM vumat_test
    !control variables
    double precision :: strain_inc = 5e-4
    double precision :: dt = 1e-3


    ! Dummy values for testing
    integer :: nblock = 1
    integer :: ndir = 3
    integer :: nshr = 3
    integer :: nstatev = 11
    integer :: nfieldv = 3
    integer :: nprops = 16
    integer :: lanneal = 0
    double precision :: stepTime = 0.0d0
    double precision :: totalTime = 0.0d0

    !define variable dimensions
    character(len=80) :: cmname ='Test material'
    double precision, dimension(1, 3) :: coordMp
    double precision, dimension(1, 1) :: charLength
    double precision, dimension(1, 16) :: props
    double precision, dimension(1, 1) :: density
    double precision, dimension(1, 6) :: strainInc
    double precision, dimension(1, 1) :: relSpinInc
    double precision, dimension(1, 1) :: tempOld
    double precision, dimension(1, 6) :: stretchOld
    double precision, dimension(1, 9) :: defgradOld
    double precision, dimension(1, 1) :: fieldOld
    double precision, dimension(1, 6) :: stressOld
    double precision, dimension(1, 11) :: stateOld
    double precision, dimension(1, 1) :: enerInternOld
    double precision, dimension(1, 1) :: enerInelasOld
    double precision, dimension(1, 1) :: tempNew
    double precision, dimension(1, 6) :: stretchNew
    double precision, dimension(1, 9) :: defgradNew
    double precision, dimension(1, 1) :: fieldNew
    double precision, dimension(1, 6) :: stressNew
    double precision, dimension(1, 11) :: stateNew
    double precision, dimension(1, 1) :: enerInternNew
    double precision, dimension(1, 1) :: enerInelasNew

    !define variable dummy values
    coordMp = reshape([0.0d0, 0.0d0, 0.0d0], [1, 3])
    charLength = reshape([1.0d0], [1,1])
    props = reshape([265.0d0, 380.0d0, 0.54d0, -0.0043d0, 1.0d0, 1.0d0, 783.0d0, 293.0d0, 73100.0d0, 28000.0d0, &
    0.3d0, 0.141d0, 0.434d0, -0.353d0, -0.028d0, 0.00346d0], [1, 16])   !important
    density = reshape([1.0d0], [1, 1])
    strainInc = reshape([strain_inc, -0.3d0* strain_inc, -0.3d0*strain_inc, &
                         0.0d0* strain_inc, 0.0d0* strain_inc, 0.0d0* strain_inc], [1, 6])    !important
    relSpinInc = reshape([0.0d0], [1, 1])
    tempOld = reshape([293.0d0], [1, 1])
    stretchOld = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 6])
    defgradOld = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 9])
    fieldOld = reshape([0.0d0], [1, 1])
    stressOld = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 6])    !important
    stateOld = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, &
                        0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 11])   !important
    enerInternOld = reshape([0.0d0], [1, 1])
    enerInelasOld = reshape([0.0d0], [1, 1])
    tempNew = reshape([0.0d0], [1, 1])
    stretchNew = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 6])
    defgradNew = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 9])
    fieldNew = reshape([0.0d0], [1, 1])
    stressNew = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 6])   !important
    stateNew = reshape([0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0, &
                        0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0], [1, 11])                !important
    enerInternNew = reshape([0.0d0], [1, 1])
    enerInelasNew = reshape([0.0d0], [1, 1])

```

```
71      ! Iterative loop for incremental loading
72      do
73          call vumat(nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal, &
74                     stepTime, totalTime, dt, cmname, coordMp, charLength, &
75                     props, density, strainInc, relSpinInc, &
76                     tempOld, stretchOld, defgradOld, fieldOld, &
77                     stressOld, stateOld, enerInternOld, enerInelasOld, &
78                     tempNew, stretchNew, defgradNew, fieldNew, &
79                     stressNew, stateNew, enerInternNew, enerInelasNew)
80
81          !termination conditions
82          if (stateNew(1, 1) >= 0.4d0) exit !strain in 11 direction exceeds 0.4
83
84          !pass on variable values of last iteratin to the next iteration
85          stressOld = stressNew
86          stateOld = stateNew
87          enerInternOld = enerInternNew
88          enerInelasOld = enerInelasNew
89      end do
90
91  END PROGRAM vumat_test
92
```

G. Abaqus Setup

The following steps outline the key procedures involved in creating a complete Abaqus simulation model of the dogbone specimen. The process assumes the base units listed in Table 4.

Table 4: Applied base units

Quantity	Unit Name	Unit Symbol
$Length$	$millimeter$	mm
$Mass$	$tonne$	t
$Time$	$second$	s
$Temperature$	$kelvin$	K
$Energy$	$Joule$	J
$Force$	$Newton$	N

Step 1: Part Design

To create the part, a 3D deformable solid was defined in the part module. To create the geometry, a 2D sketch was created on the XZ plane, representing the cross-section of the desired shape. This sketch can be seen in Figure 40.

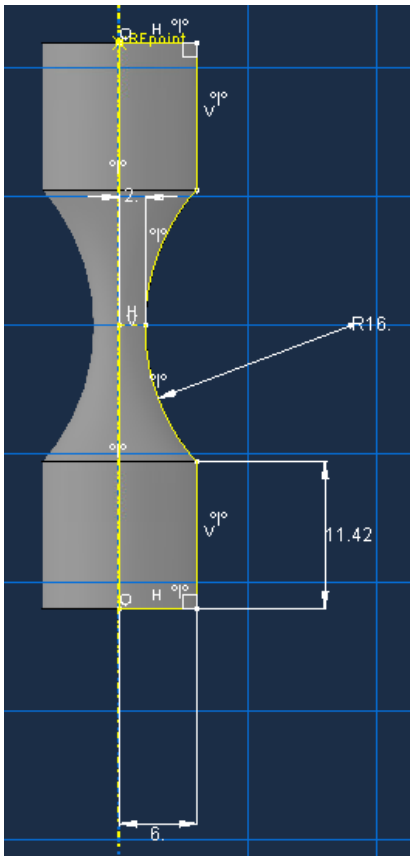


Figure 40: Cross-section dimensions

After completing the sketch, the revolve feature was used to rotate the profile around its central vertical axis. This operation generated a full 3D solid model.

Step 2: Material Implementation and Assembly

The material properties were defined in the material module. Since a user-defined material model was required, the necessary parameters were specified under the General tab.

First, the Density option is used to assign the material density of $2.77 \cdot 10^{-9}$. Second, the Depvar option was selected, and 11 solution-dependent state variables were defined for the user material model. Lastly, the User Material (User) option was enabled to indicate that the material behavior would be controlled through a user-defined subroutine (VUMAT).

After defining the material, it must be assigned to the part. In the section module, a section is created and linked to the appropriate material properties. Then, in the “Assign Section” option, this section is applied to the part created in step 1.

Once the material definition is fully completed, the part can be loaded into an assembly within the assembly model.

Step 3: Creating a Step and Requesting Output Data

Within the step module, a step needs to be defined. This step should be a dynamic explicit step to invoke the explicit solver. Within the newly created step window, the following details are specifically defined:

- Basic- time period: Must be set accordingly to accommodate a certain loading rate.
- Incrementation – type: Fixed
- Incrementation – user-defined time increment: Must be set appropriately with respect to time period
- Mass scaling – use definitions below: This option is to give the freedom to choose the time increment size

Within the mass scaling definition, the settings within Figure 41 are used. Within these settings, the value of the target time increment can be assumed variable depending on the exact simulation that needs to be run.

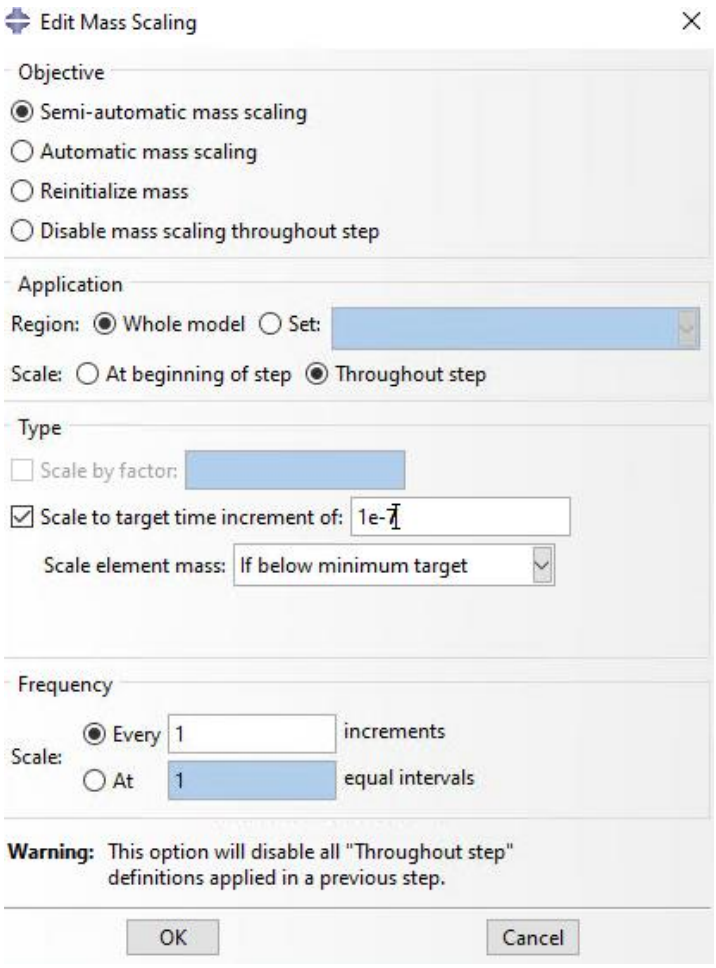


Figure 41: Mass scaling settings

Next, the appropriate outputs need to be requested. For the history output request, the output shown in Figure 42 is requested. This output is requested for the reference point, which is located in the location depicted in Figure 43.

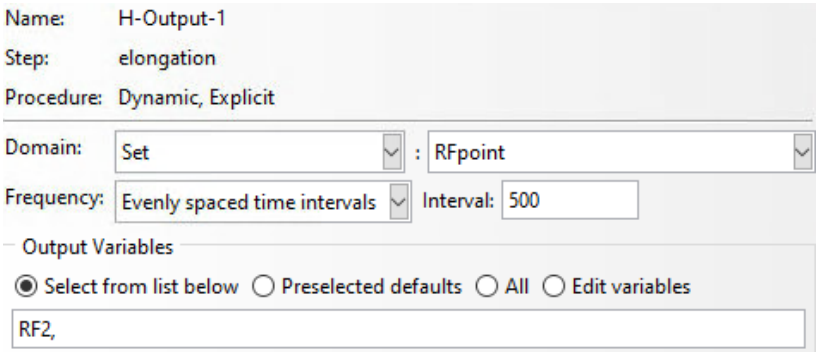


Figure 42: History output request

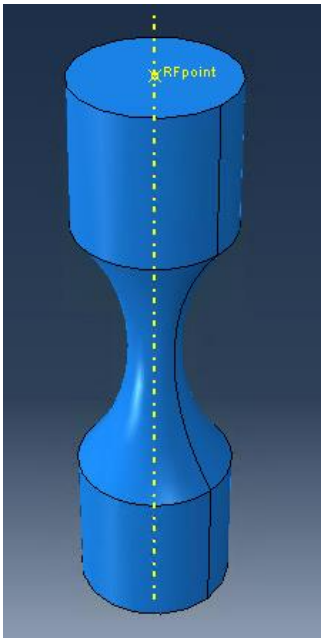


Figure 43: Location of reference point

For the field output request, the outputs given in Figure 44 are requested.

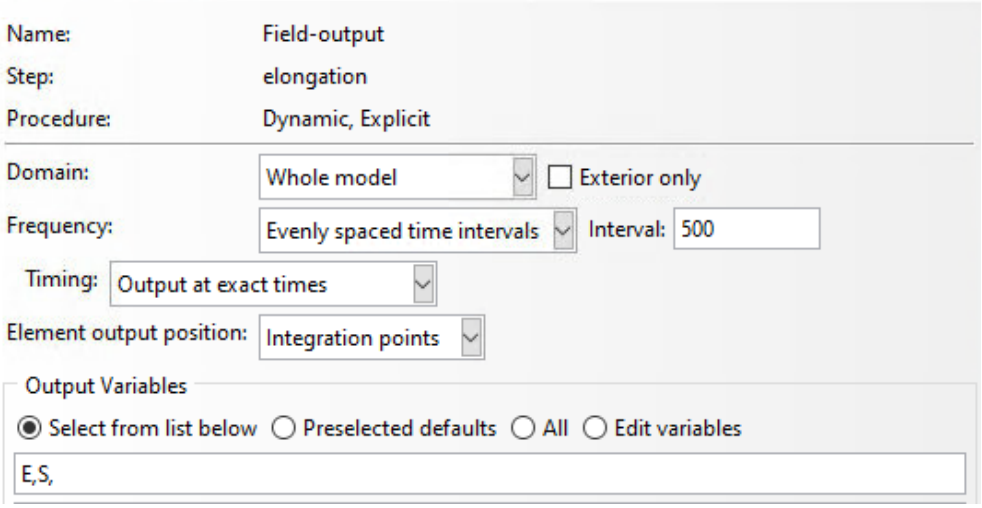


Figure 44: Field output request

Step 4: Apply Loading

To apply loading to the specimen, two boundary conditions are defined. The first boundary condition effectively clamps the lower part of the specimen, as can be seen within Figure 45.

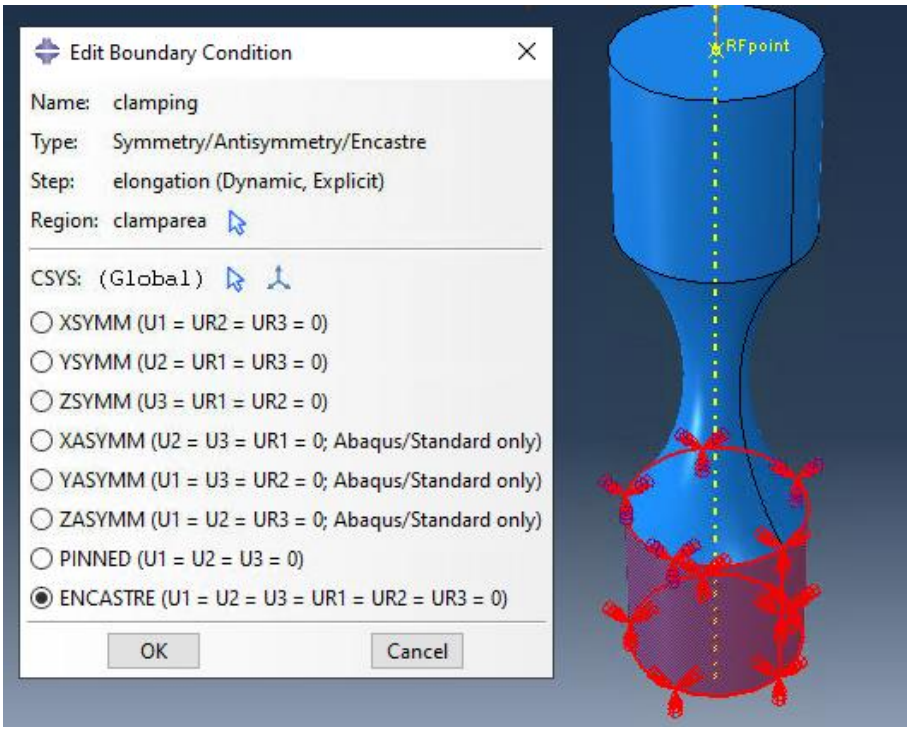


Figure 45: Clamping boundary condition

The second boundary condition is a displacement boundary condition that is applied to the reference point, as can be seen within Figure 46.

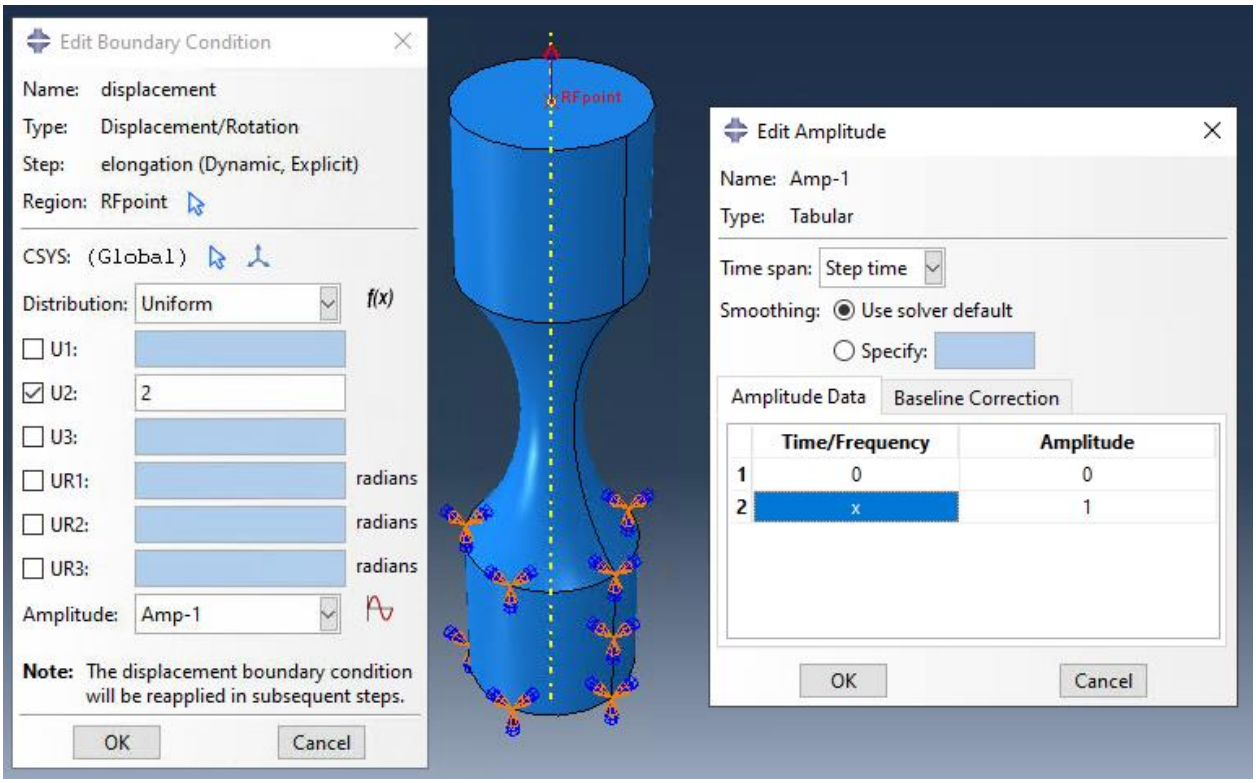


Figure 46: Displacement boundary condition

Within the amplitude specification window, the final time (marked by an x) is variable depending on the displacement rate that is required.

The reference point, to which the displacement boundary condition is applied, needs to be connected with the appropriate regions via a constraint. This constraint is specified within Figure 47.

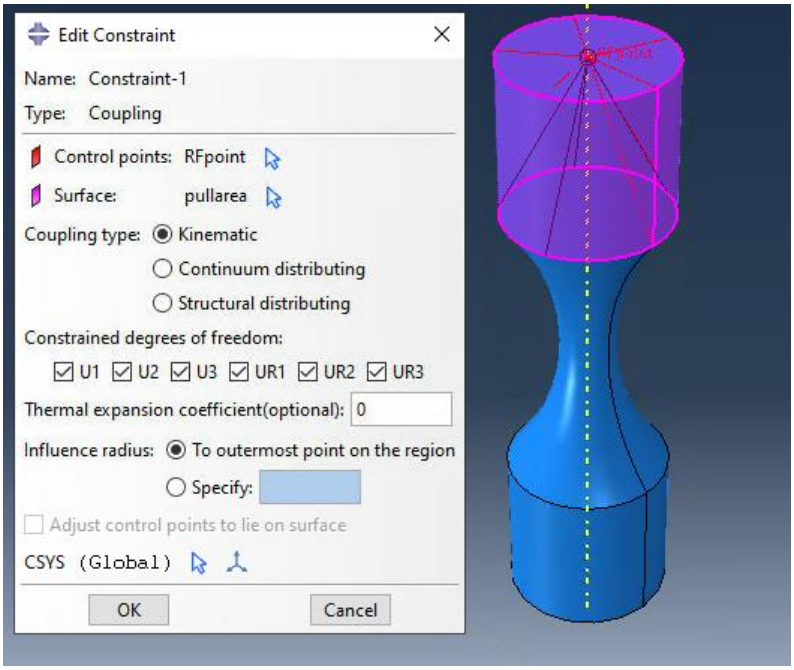


Figure 47: Coupling constraint of the reference point

Step 5: Apply Mesh

Within the mesh module, a mesh is applied to the specimen. This mesh is created by applying local seeds and meshing the entire part as shown in Figure 48.

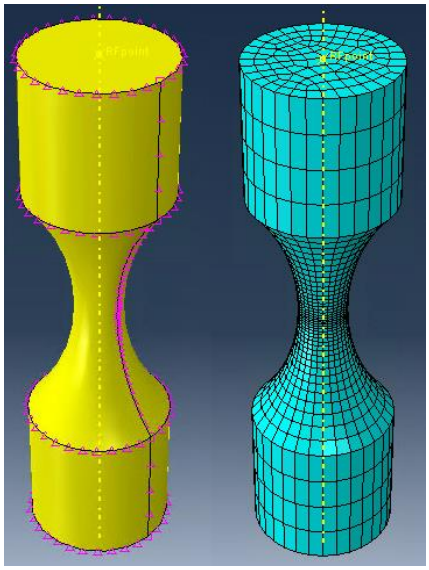


Figure 48: Meshing of specimen

After applying the mesh, the element type must be specified by the element definition given in Figure 49 and applied to the entire specimen mesh.

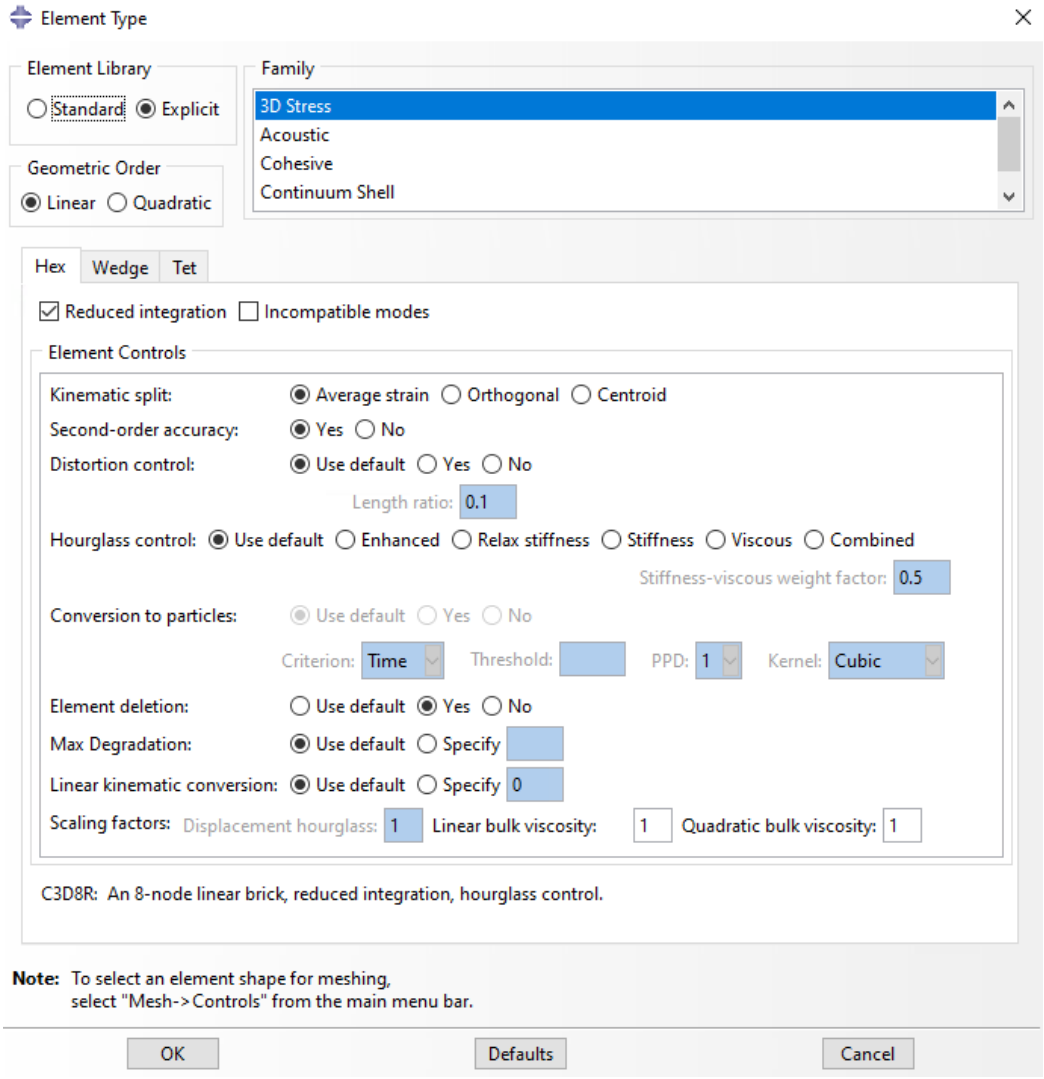


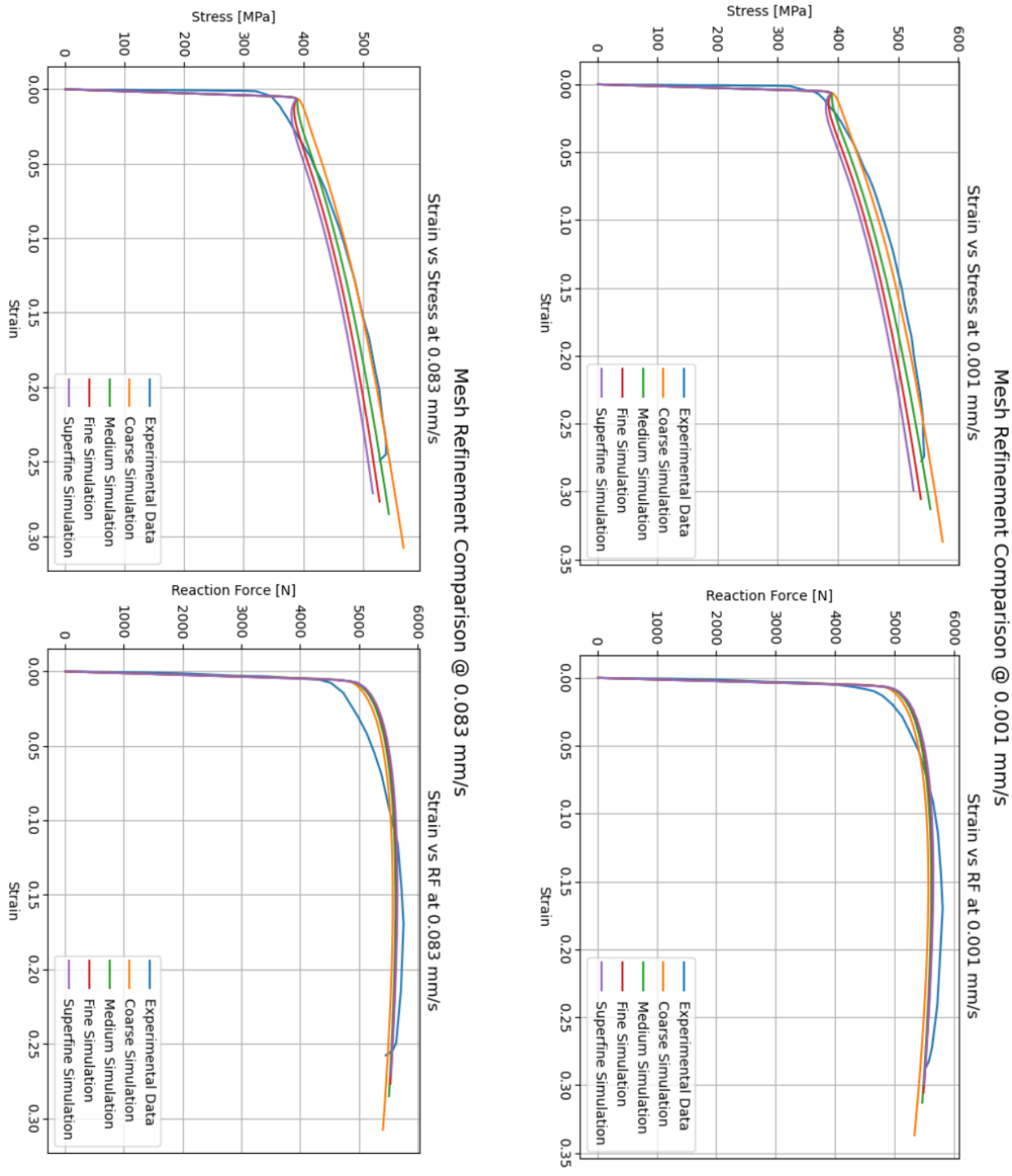
Figure 49: Element type definition

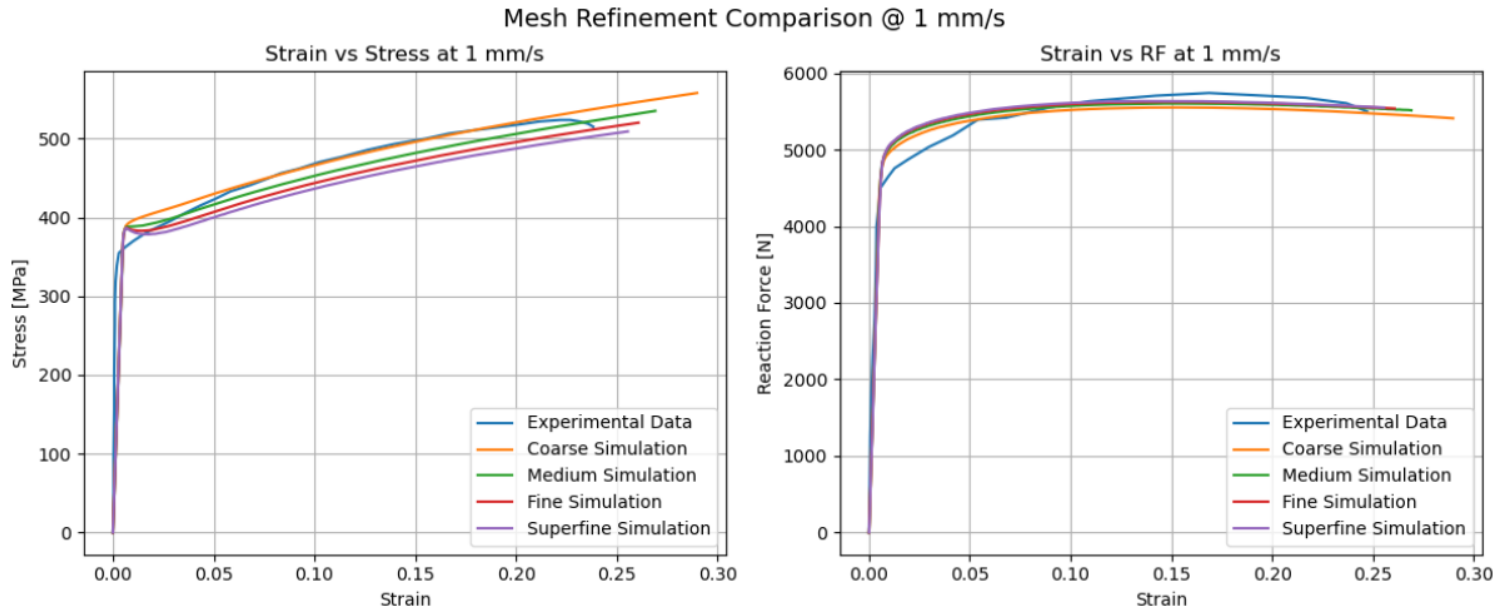
Step 6: Create a Job

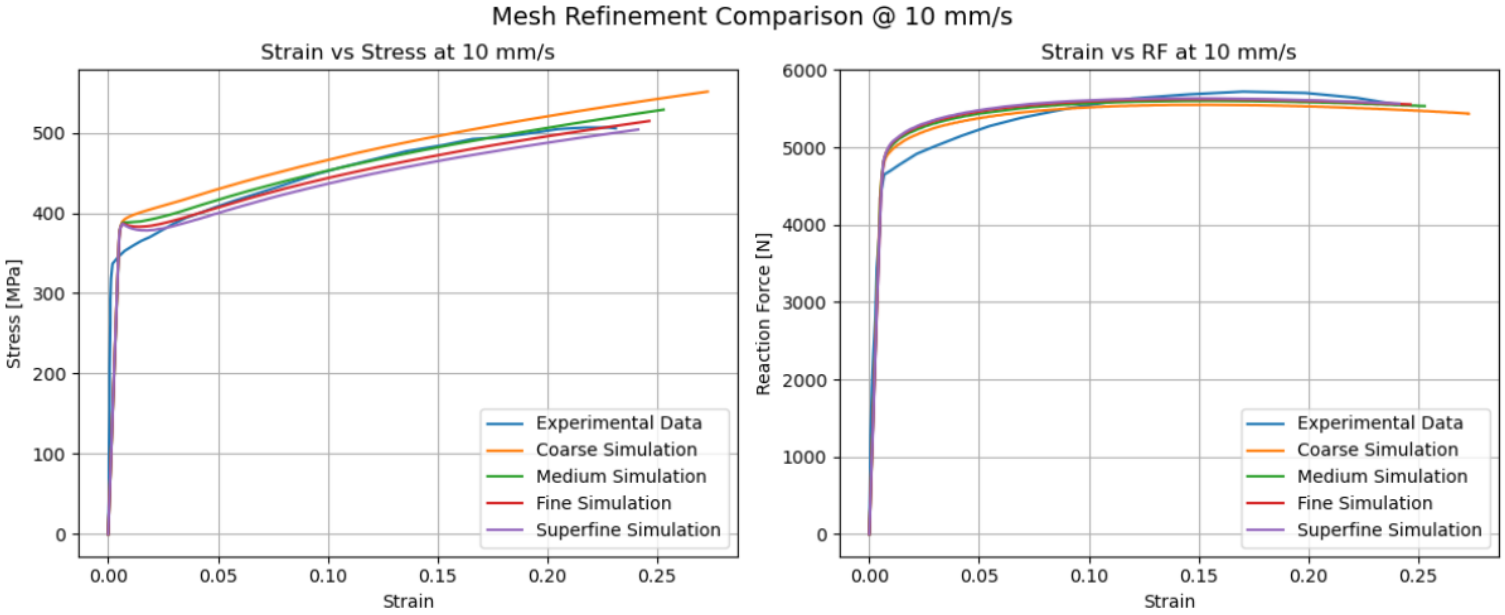
As a last step, within the job module, a job needs to be created. Within the creation of this job, the following input parameters are important:

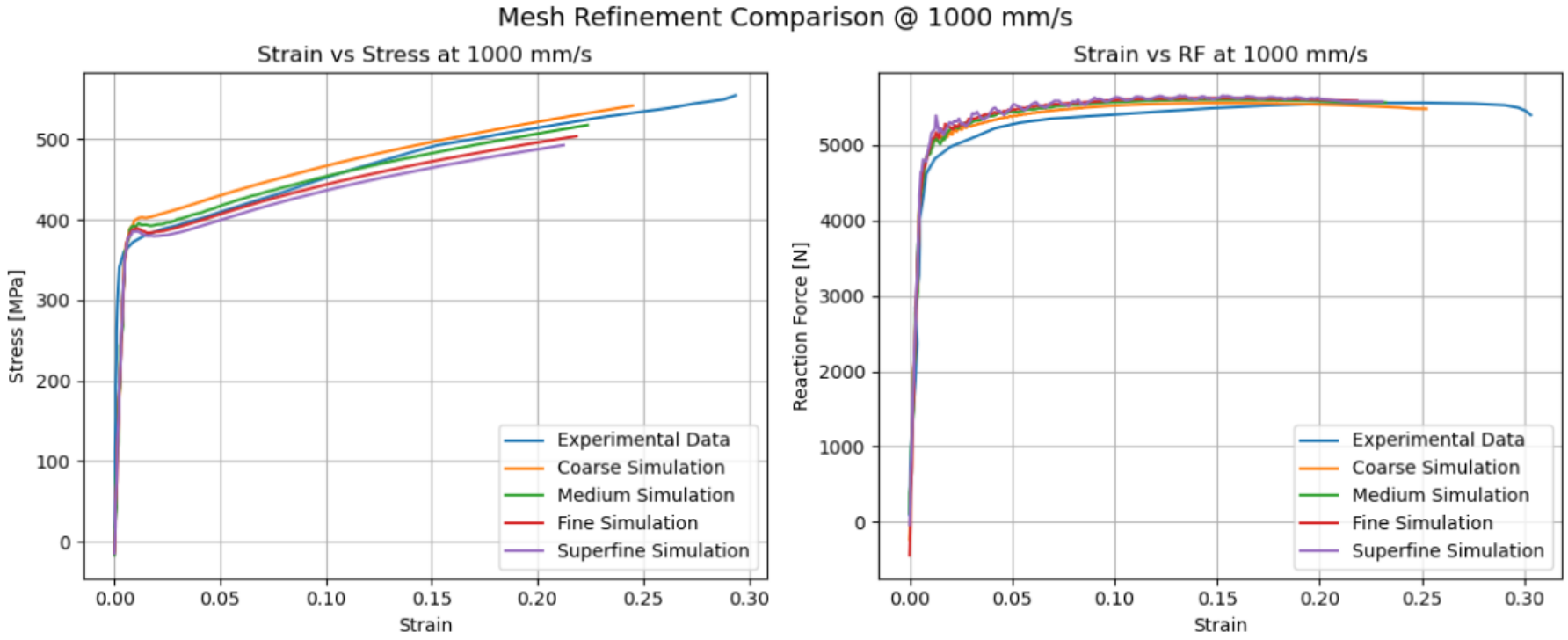
- Submission - Job type: Full analysis
- Precision – abaqus/explicit precision: Double precision to avoid rounding errors

H. Mesh Convergence Results




Mesh Refinement Comparison @ 1 mm/s
Strain vs Stress at 1 mm/s
Stress [MPa]
Strain
Strain vs RF at 1 mm/s
Reaction Force [N]
Strain
Experimental Data
Coarse Simulation
Medium Simulation
Fine Simulation
Superfine Simulation


Mesh Refinement Comparison @ 10 mm/s
Strain vs Stress at 10 mm/s
Stress [MPa]
Strain
Strain vs RF at 10 mm/s
Reaction Force [N]
Strain
Experimental Data
Coarse Simulation
Medium Simulation
Fine Simulation
Superfine Simulation


Mesh Refinement Comparison @ 1000 mm/s
Strain vs Stress at 1000 mm/s
Stress [MPa]
Strain
0
100
200
300
400
500
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Experimental Data
Coarse Simulation
Medium Simulation
Fine Simulation
Superfine Simulation
Strain vs RF at 1000 mm/s
Reaction Force [N]
Strain
0
1000
2000
3000
4000
5000
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Experimental Data
Coarse Simulation
Medium Simulation
Fine Simulation
Superfine Simulation