Bachelor's Programme in Science and Technology

Optimization of nanoindentation related crystal plasticity parameters

Youngbin Pyo, Irem Kurt, Mac Teppo, Arturo Martin

Computational Engineering Project
2024

Table of Contents

1. Abstract

This project focuses on the calibration efficiency of the Crystal Plasticity Element Method (CPFEM) through the optimization of nanoindentation and crystal plasticity parameters using machine learning techniques. CPFEM integrates the principles of crystal plasticity with finite element analysis (FEA) and describes the deformation of single crystals and polycrystals assuming crystallographic slip as the main deformation mode. CPFEM is essential for forecasting the mechanical behaviour of our targeted materials, DP1000 and QP1200, under various loading conditions. However, the right calibration of CPFEM is also time consuming as it relies on various curve fitting methods and extensive computational power and resources. This project addresses these limitations by developing an automated workflow that integrates three main areas: nanoindentation experiments, computational simulations, and machine learning algorithms.

Nanoindentation, a technique for assessing mechanical properties of materials at the nanometer scale, provided experimental data on DP1000 and QP1200. This data was then used to calibrate CPFEM parameters such as initial resolved shear stress, hardening rate, and critical shear stress. To optimize CPFEM calibration, many potential machine learning algorithms were explored. An Artificial Neural Network (ANN) was ultimately chosen because of its superior ability to identify correlations present within the data.

Moreover, the proposed workflow enhances predictive modelling capabilities and offers valuable information for material design and manufacturing processes. These results contribute to improving material performance and innovation in sectors like automotive, aerospace, and energy.

2. Introduction

Crystal plasticity modelling is a method used to analyse, and predict a materials properties at its phase crystal levels. It has become increasingly more widespread throughout recent years as a key method to understand material behaviour. Using these CP models, we can learn about how materials deform under different stress loads, which can further be used to predict deformation behaviour and yield locus formation.

The application of crystal plasticity models requires the accurate calibration of the parameters which influence the models mechanical behaviours, and while this allows us to observe precise results, it comes with a set of challenges. The calibration of these models usually relies on experimental data which is obtained via nanoindentation testing, which helps us measure the materials mechanical properties, such as its hardness, at the nanoscale. We are able to obtain localised mechanical responses through this method, and when used in combination with CP models, it allows us to simulate large-scale material behaviours. Parameter calibration for these models, however, is often time-consuming, and complex, requiring the use of large amounts of computational resources, and deep knowledge in material science and programming.

These challenges can be tackled via many different methods, such as through the use of computational techniques, such as the Integrated Computational Materials Engineering (ICME) approach. This approach has transformed into a heavily relied on method to help us address the shortcomings of CP model calibration. ICME helps describe the relationships between a materials microstructure and its mechanical properties, by providing a framework to allow for qualitative and quantitative evaluation of them. Crystal plasticity modelling is an important aspect of this framework, as it allows for the prediction of how various manufacturing processes impact the performance of specific materials. Although there are many advances in addressing such challenges, the parameter calibration process is still the main constrictor of efficient application of CP models.

This paper explores the development of an automated parameter calibration process for crystal plasticity modelling, through the use of nanoindentation testing results. In particular, this project aims to create a workflow to control and optimise the calibration process, via the use of Python. The automation of this calibration process allows for the significant reduction in time used, and effort required to achieve reliable predictions of a materials behaviour. This project, developed on the CSC platform, displays an advancement in the automation process of material characterization workflows, and the optimization of CP models for their real-world application.

Our research not only allows for a deeper understanding of a materials mechanical behaviours under various testing conditions, but also for a more efficient calibration process. This work contributes to the constantly developing pool of knowledge in material science, especially in respect to the topic of advanced manufacturing.

Nanoindentation is a powerful and versatile mechanical testing technique used to investigate the properties of materials at microscopic and nanoscopic scales. The method involves pressing a sharp indenter, typically made of a hard material like diamond, into the surface of a test specimen while monitoring the applied force and the resulting penetration depth. The load-displacement curve generated during this process serves as the primary data source for determining key material properties such as hardness, elastic modulus, and, in certain cases, time-dependent viscoelastic or creep behavior.

Unlike traditional mechanical testing methods, nanoindentation excels in scenarios where only a small volume of material is available, such as thin films, coatings, and small-scale structures. Its precision and reliability stem from its ability to measure very small loads and displacements, often at the nanonewton and nanometer scale, respectively. Furthermore, nanoindentation is a minimally destructive method, which means that valuable or limited samples can often be preserved for additional testing. In research fields like materials science, biomechanics, and nanotechnology, nanoindentation is indispensable for probing localized mechanical responses that govern a material's overall behavior.

Future Implications of Nanoindentation Results

Nanoindentation research offers significant benefits across various scientific and industrial fields. In materials engineering, it provides critical data to connect a material's microstructural characteristics to its macroscopic performance. By examining how different phases within a material respond to localized stresses, it becomes possible to design new alloys with improved properties such as higher strength, better ductility, or enhanced wear resistance. This understanding directly supports advancements in key industries like automotive, aerospace, and energy.

Furthermore, nanoindentation results play a vital role in the development of predictive modeling techniques like the Crystal Plasticity Finite Element Method (CPFEM). These models simulate material behavior under specific conditions, such as forming operations or high-stress environments, with a high degree of accuracy. By reducing reliance on extensive physical testing, these simulations save both time and resources, accelerating the innovation pipeline in material design.

In manufacturing, nanoindentation has practical implications for quality control. It provides a fast and reliable way to assess mechanical properties, ensuring that materials meet required standards before they are integrated into critical applications. Moreover, the insights from nanoindentation can guide the development of more sustainable materials by identifying opportunities to reduce energy usage or waste during production without compromising performance. This is particularly important as industries strive to balance performance requirements with environmental considerations.

In manufacturing, nanoindentation can improve quality control protocols by providing rapid and precise assessments of mechanical properties. This ensures that materials meet stringent specifications before they are incorporated into critical components. Furthermore, nanoindentation studies can inform the development of sustainable materials by identifying ways to reduce the environmental impact of production processes while maintaining or improving performance.

Rationale Behind the Nanoindenter Selection

The accuracy of nanoindentation experiments is heavily reliant on the choice of instrument. For this project, a precision nanoindenter capable of supporting both conical and cube-corner indenters was selected to provide flexibility and ensure reliable data collection. This choice reflects the need to explore different aspects of mechanical behavior, such as plastic deformation and elastic recovery, under varying test conditions.

The selected nanoindenter offers exceptional precision, achieving sub-nanometer displacement resolution and nanonewton force resolution. These features are crucial for accurately measuring the behavior of small material volumes, which is essential in studies like this. The ability to switch between multiple indenter geometries allows experiments to be tailored to specific objectives. For instance:

- Cube-corner indenters provide sharper contact points, making them ideal for studying material yield and plastic deformation.
- Conical indenters are suited for capturing elastic recovery and stress distribution.

Additionally, the instrument's advanced automation capabilities minimize operator involvement, reducing variability and ensuring repeatability in results. Its ability to conduct high-throughput tests enhances efficiency, allowing for larger datasets to be collected in shorter timeframes. Furthermore, seamless integration with computational platforms ensures that experimental data can be directly incorporated into simulations like CPFEM, enabling streamlined workflows for calibrating material models and optimizing parameters. The combination of precision, versatility, and automation makes this nanoindenter an ideal tool for the project's goals of refining material characterization techniques and improving predictive modeling.

3. Literature Review

3.1 Nanoindentation 4p

Nanoindentation is the most popular technique for assessing mechanical properties of materials at the nanometer scale. The main goal is to calculate the elastic

modulus and hardness of the targeted material based on readings of load and depth of penetration. The principal elements used with this technique are the test material, the mechanical load, and the indenter displacement (Yao Yao, p.4). Nanoindentation works by pressing a hard, sharp indenter into the surface of a material while recording the force and displacement during both the loading and unloading phases. The data obtained is then utilized to calculate properties such as hardness, elastic modulus, and creep behavior. Moreover, as the penetration load and depth are relatively small, which is non-destructive to the surface, nanoindentation tests have been widely used in bulk materials and thin films (Yao Yao, p2).

The first diagram (a) shows how an indenter applies a controlled load onto the surface of a material to a maximum depth represented by h_max. The second diagram (b) displays the load-displacement curve and its different phases: During the loading phase (1), the indenter penetrates the material, followed by a hold phase (2) where the load is constant. The unloading phase (3) reflects the material's elastic recovery as the load is removed. The slope of the unloading curve ‘S’ is used to calculate the elastic modulus, while the maximum load and depth help to determine hardness.

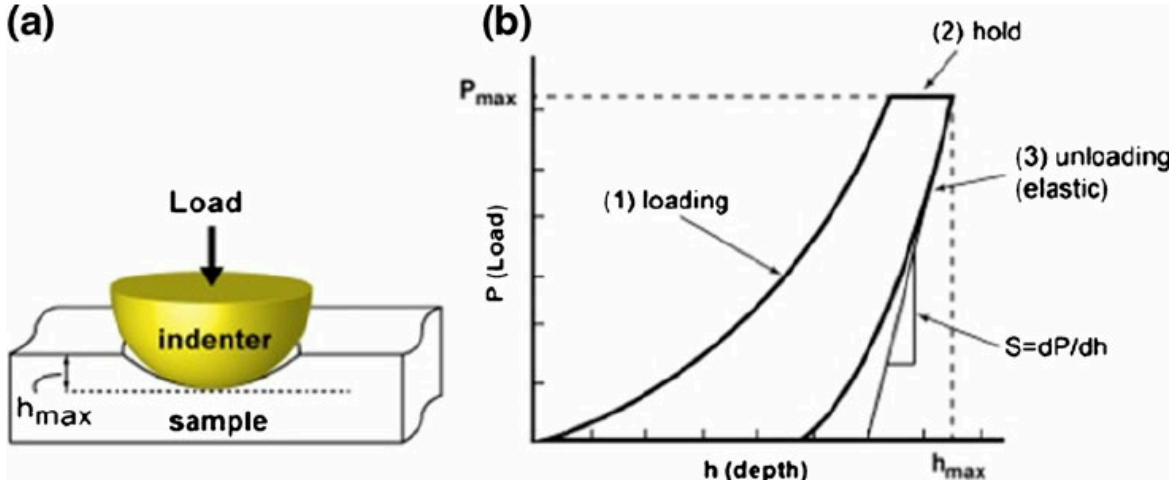


Figure 1: ref

The technique operates by recording the load and displacement of an indenter as it penetrates a material's surface. The resultant load-displacement curve serves as a fingerprint of the material’s response, from which key properties are extracted. The

slope of the unloading curve indicates the elastic modulus, while the area under the loading curve relates to hardness (Wang et al., 2018). Such precision, coupled with its non-destructive nature, allows nanoindentation to complement computational models like the Crystal Plasticity Finite Element Method (CPFEM), where experimental data is used to calibrate simulations (Nguyen et al., 2022).

In the context of material modelling, nanoindentation bridges experimental and computational domains. It informs CPFEM by providing essential boundary conditions and calibration parameters derived from real-world tests. This integration is especially crucial for materials with complex microstructures, such as dual-phase steels or composites, where phase-specific mechanical properties are necessary for accurate modelling (Liu et al., 2020). Nanoindentation tests use different types of indenters, with Berkovich, spherical, and cube-corner geometries being the most common. Indenters are chosen based on their shape and size to match the necessary force to affect the mechanical response of the material tested.

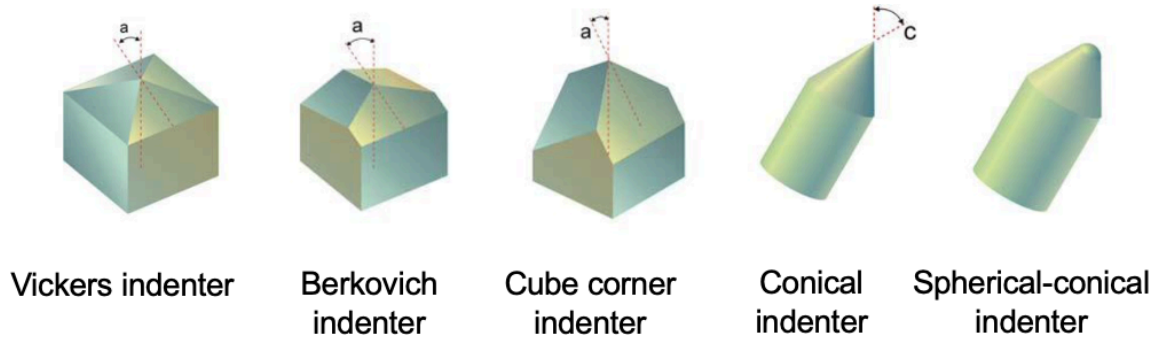


Figure 2: Different types of indenters ()

3.1.1 Indentation Size Effect

The indentation size effect (ISE) is a well-documented phenomenon in nanoindentation, where the measured hardness of a material increases as the indentation depth decreases. This size-dependent behavior has significant implications for the interpretation of nanoindentation results, particularly in materials with fine microstructures or small characteristic dimensions (Liu et al., 2015; Moreno & Hähner, 2018).

The following graph illustrates the ISE for both Berkovich and Spherical indenters. At low depths, hardness rises due to strain gradients and geometrically necessary dislocations. Also, we can observe how sharper indenters, such as the Berkovich one, exhibit a greater ISE compared to the Spherical indenter because of higher localized stress and strain gradients.

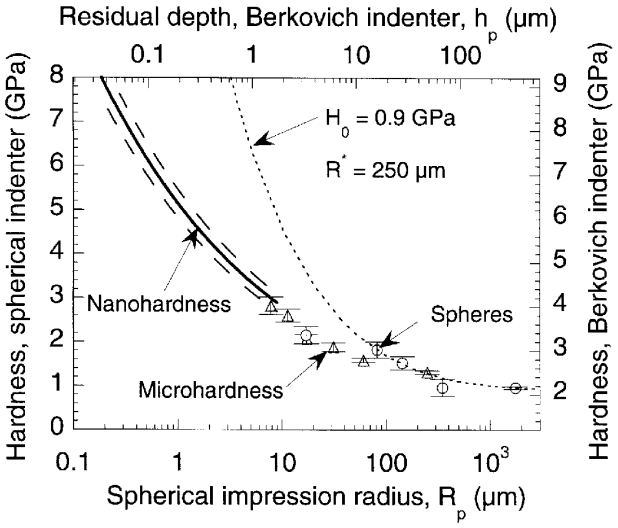


Figure 3: Correlation of the indentation size effect with different shapes indenters (J.G. Swadener, 2002)

ISE arises due to various mechanisms, the most prominent of which is the strain gradient plasticity theory. At smaller scales, the material's plastic deformation is dominated by geometrically necessary dislocations (GNDs) required to accommodate non-uniform strain. The increased density of GNDs leads to higher apparent hardness at smaller indentation depths. Other factors contributing to ISE include dislocation pileups, work hardening, and lattice curvature effects (Moreno & Hähner, 2018; Zhang et al., 2018).

For CPFEM and related modeling techniques, accounting for ISE is critical to achieving realistic simulations. The strain gradient plasticity framework incorporates an intrinsic length scale that captures the additional strengthening observed at small scales (Wang et al., 2018). Modified models, such as the Nix-Gao approach, have been employed to refine predictions of hardness and elastic modulus at varying depths. For example, Liu et al. (2015) demonstrate how CPFEM simulations incorporating ISE provide accurate load-displacement predictions compared to experimental results.

ISE also influences nanoindentation's applicability to multi-phase materials. Different phases, such as ferrite and martensite in steels, may exhibit varying hardness values influenced by their microstructural length scales. Correctly interpreting these variations is essential for using nanoindentation data to calibrate CPFEM simulations accurately (Liu et al., 2020).

3.1.2 Significance for Material Modelling

Nanoindentation plays a pivotal role in material modeling, particularly in calibrating and validating computational methods like CPFEM. Its high spatial resolution and ability to target specific regions within a material make it ideal for studying heterogeneous materials, such as composites or alloys. Nanoindentation can differentiate the mechanical behavior of individual phases, providing phase-specific parameters necessary for CPFEM simulations (Nguyen et al., 2022).

For example, in dual-phase steels, nanoindentation can independently assess the ferrite and martensite phases, enabling researchers to model their combined mechanical behavior more accurately. Such data is critical for CPFEM, which relies on accurate phase-specific properties to predict the overall stress-strain response of the material. Studies like those by Zhang et al. (2018) highlight how CPFEM uses nanoindentation results to simulate multi-scale mechanical responses.

Nanoindentation’s ability to measure strain-rate sensitivity further enhances its utility. By varying the rate of indentation, researchers can study viscoelastic and time-dependent plasticity behaviors, which are integral to materials operating under dynamic conditions. CPFEM integrates these insights to simulate real-world applications, such as high-strain-rate forming or impact scenarios (Wang et al., 2018).

Moreover, nanoindentation complements advanced characterization techniques like electron backscatter diffraction (EBSD) or transmission electron microscopy (TEM). These combined methods provide comprehensive data on crystallographic

orientations, phase distributions, and defect structures, all of which are crucial inputs for CPFEM (Liu et al., 2020).

3.2 Crystal Plasticity Finite Element Method (CPFEM) 4p

The Crystal Plasticity Finite Element Method (CPFEM) is a computational technique that integrates the principles of crystal plasticity with finite element analysis (FEA). According to Liu et al. (2015), the method is a design to simulate the microscopic mechanical behavior of crystalline materials, including their anisotropic nature, deformations, and grain orientation. By analyzing all these microstructural features, CPFEM offer an accurate and predictive representation compared to traditional approaches such as continuum mechanics models.

The greater benefit of CPFEM is its ability to model the deformation processes, including dislocation slips and twinning, which govern the plasticity of individual crystals. In contrast to macroscopic models that presume uniform material behavior, CPFEM specifically considers the heterogeneity of materials caused by their microstructures. This element makes CPFEM ideal model studying complicated materials, including phase alloys, composites, and polycrystalline metals.

One of the most significant features of CPFEM is its ability to adapt to different loading scenarios, from simple uniaxial tension to complex multi-axial cyclic loading. This is because CPFEM can provide information about the mechanical performance of materials while capturing texture evolution, stress distributions, and strain localization at the grain level (Zhang et al., 2018). This characteristic is why CPFEM is preferred for experiments requiring detailed predictions of material behavior. The model is commonly used in aerospace alloys, automotive steels, and microelectronics.

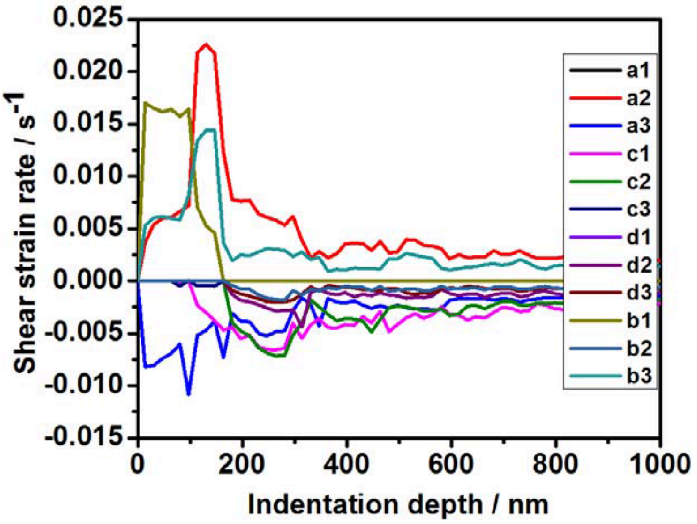


Figure 4: Crystal plasticity finite element method size effect (M Liu, 2015)

3.2.1 Models

Constitutive models form the core of CPFEM, describing how individual grains deform under applied stresses. These models can be broadly classified into two categories: phenomenological models and dislocation-based models.

Phenomenological models use empirical relationships to describe the stress-strain behavior of materials. They rely on mathematical formulations that capture the macroscopic response of crystals without explicitly modeling the underlying deformation mechanisms. While phenomenological models are computationally efficient, they may lack the precision needed for capturing complex behaviors like strain localization or gradient-dependent phenomena.

Dislocation-based models explicitly incorporate the motion and interaction of dislocations, which are line defects in the crystal lattice responsible for plastic deformation. Dislocation-based models provide a more detailed and physically accurate representation of crystal plasticity by linking stress and strain to microstructural features such as dislocation density, slip system activation, and grain boundary interactions (Moreno & Hähner, 2018).

3.3 Integration of Nanoindentation and CPFEM

The integration of nanoindentation and Crystal Plasticity Finite Element Method (CPFEM) is a critical step in accurately modeling the mechanical behavior of materials. Nanoindentation provides experimental data in the form of force-displacement (FD) curves, which encapsulate key material properties, such as hardness and elastic modulus, at localized regions. CPFEM, on the other hand, offers a computational framework to simulate material responses at the microstructural level, enabling detailed insights into phase-specific behavior and deformation mechanisms.

Nanoindentation and CPFEM complement each other by bridging experimental observations and computational modeling. Nanoindentation tests generate high-resolution data for specific phases, such as Ferrite and Martensite, which are then used to calibrate CPFEM parameters. CPFEM employs these parameters to model stress-strain relationships, dislocation mechanisms, and grain-level interactions under various loading conditions.

In this project, nanoindentation was employed to characterize the behavior of DP1000 and QP1200 steels. The experimental FD curves served as benchmarks to calibrate CPFEM parameters, including initial resolved shear stress (tau0), hardening rate (h0), critical shear stress (tcs), and hardening exponent (a). By iteratively tuning these parameters, CPFEM simulations were aligned with the experimental results, enhancing the predictive accuracy of the model. This integration ensures that the simulated mechanical response reflects the intrinsic properties of the materials, making it a robust method for studying deformation behaviors.

A key challenge in this process is the nonlinear and anisotropic nature of the materials, which demands precise parameter calibration. To address this, the project utilized both traditional curve-fitting techniques and machine learning (ML) algorithms to optimize parameter calibration. By leveraging nanoindentation data and CPFEM simulations, the integration has provided a comprehensive approach to studying localized mechanical properties and their impact on material performance.

3.4 Machine Learning in Parameter Calibration and CPFEM Modeling

Machine learning (ML) was an important part of our project. We used it to optimize the efficiency of CPFEM parameter calibration, which is a manual and time-consuming process. By the use of machine learning models, we analyzed nanoindentation data. As a whole, our project aimed to optimize the parameter calibration process, making it more efficient and accurate.

We used a range of ML models to identify the best approach for predicting CPFEM parameters:

- Artificial Neural Networks (ANNs):

We chose to use ANN for its ability to model complex, nonlinear relationships. The network architecture included multiple layers with ReLU activation functions, enabling the model to capture subtle variations in the FD curves. ANNs showed high accuracy in the parameter prediction process such as tau0 and h0. This made them the most effective model for this project, and was the machine learning algorithm we chose to proceed with.

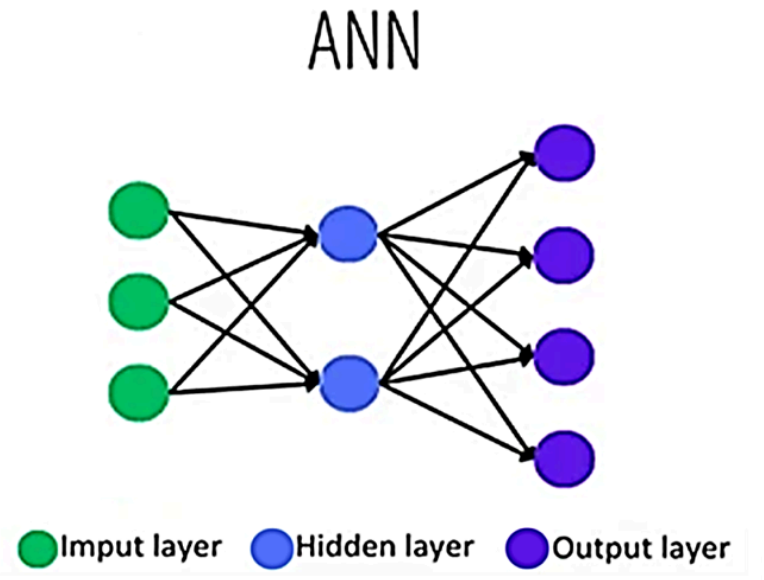


Figure 3.4.1 ANN Diagram

- Long Short-Term Memory Networks (LSTMs):

We also decided to try out LSTMs because of their ability to handle sequential or time-series data, which aligns well with the sequential nature of FD curves. LSTMS did a good job at observing temporal dependencies well, but their downside was that their computational demands were higher. The increase in performance was not pronounced enough to justify using it instead of ANNs for our application.

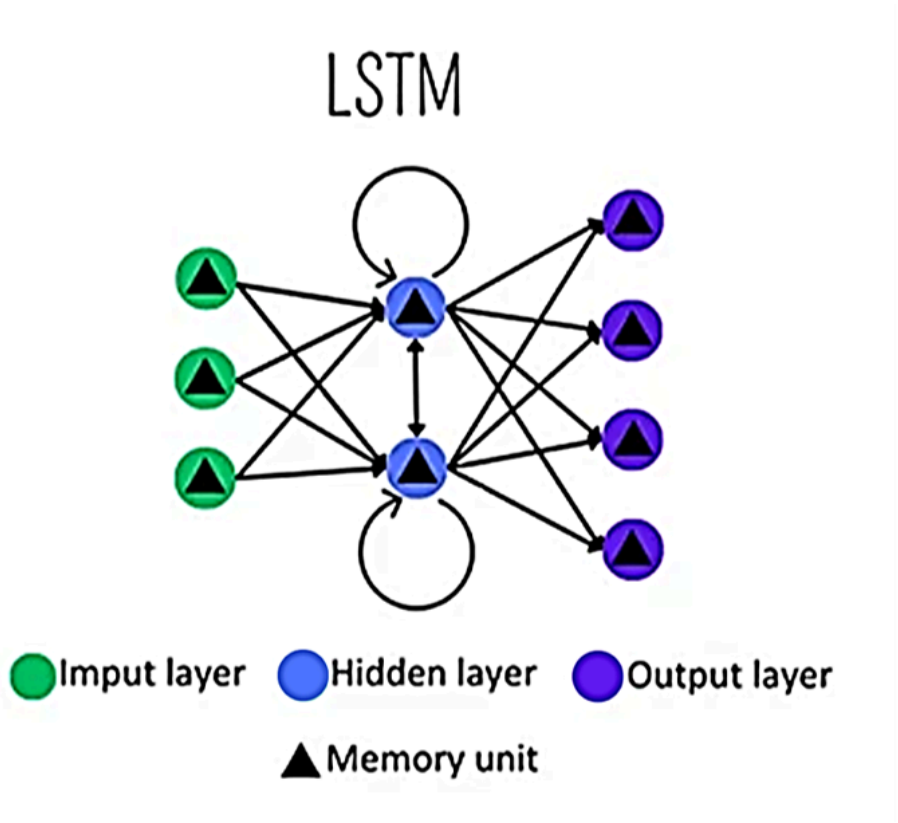


Figure 3.4.2 LSTM Diagram

- Convolutional Neural Networks (CNNs):

We also tried out CNNs with 1D convolutions to observe localized features within the FD curves. As a whole, they did a good job at observing large changes and critical points in the data, however, they performed worse than ANNs as a whole due to the somewhat simple structure of the FD curves.

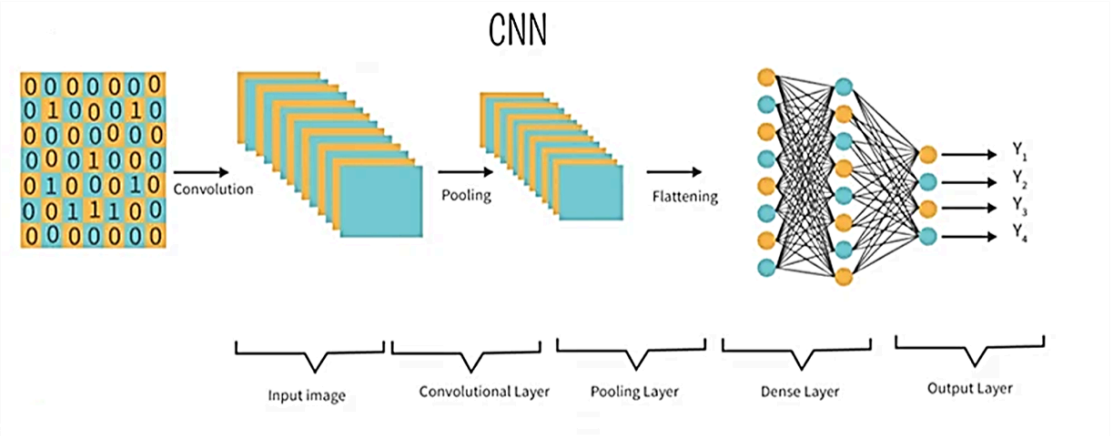


Figure 3.4.2 CNN Diagram

- Random Forest Regressors (RFs):

We utilised RFs as a baseline model for structured data. They provided good results when regarding linear data, and had easy interpretability and were robust to noise. The downside was that RFs struggled to capture the nonlinear dependencies between features and outputs. In total, when compared to the other machine learning models we tried, RFs had higher prediction errors.

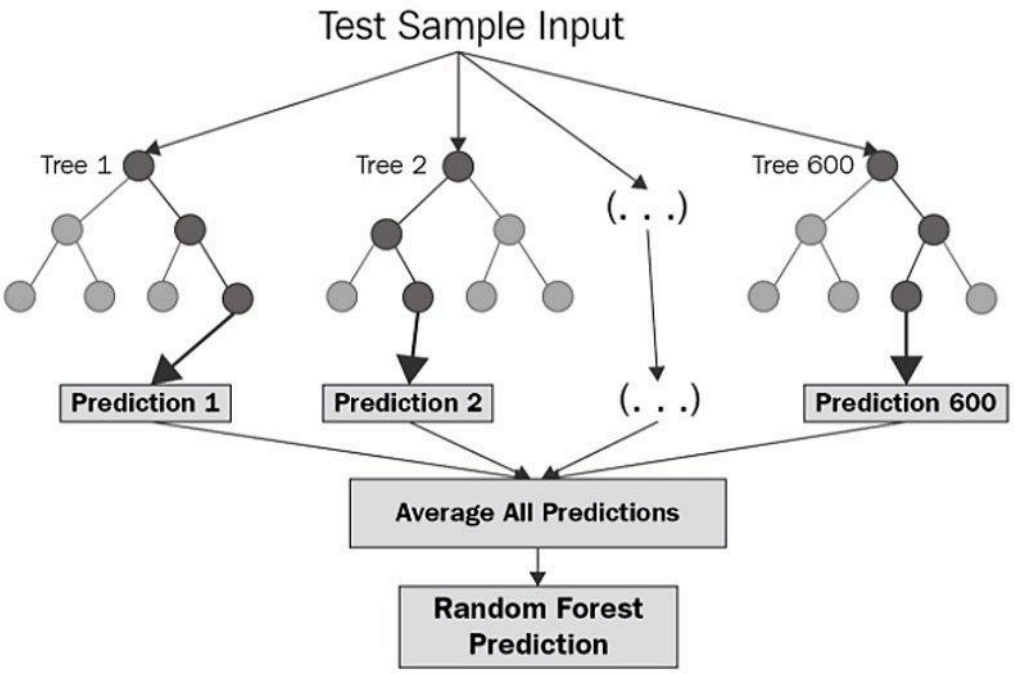


Figure 3.4.2 RF Diagram

We evaluated these models using multiple different metrics including R-squared (R^2) and Root Mean Square Error (RMSE). Among all tested models, ANNs achieved the best balance between accuracy and computational efficiency. For this reason, we were inclined to use them as our primary choice for parameter calibration.

We started our machine learning workflow with preprocessing of the data. This included both normalization and outlier removal, to ensure the consistency and quality of the training data. We also simplified our dataset using feature engineering by selecting key parameters and critical points, such as peak and drop points, and target outputs. We then used our trained ML models to predict different materials' CPFEM parameters, such as DP1000 and QP1200, and their predictions were validated against experimental FD curves.

Our project provides a significant advancement in the automation of the parameter calibration process via the integration of machine learning into CPFEM modeling. By reducing human intervention and improving efficiency, our project establishes a scalable framework for material characterization and modeling.

4. Experiment Setup/Parametric Studies

4.1 Experiment Setup

The primary method for evaluating the material properties in our experiment was stress-strain curves. Such curves allow observers to see the deformation behavior of a material when applied with force. In our experiment, the forces are used on a nano-scale. Two materials were studied: Dual Phase steel DP1000 (Ferrite) and Q&P steel QP1200 (Ferrite and Martensite). Blue Grain with the [111] slip system was the grain used for DP1000.

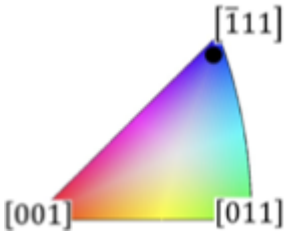


Figure 4.1.1 Standard stereographic triangle and grain (Liu et al., 2019c)

The Indenters used were a cubecorner indenter with a diameter of 1um for DP1000 and a spherical-conical indenter with a diameter of 1.5um for QP1200. The strain rate and displacement control loadings were 0.01nm/s and 175nm for DP1000 and 1nm/s and 150nm for QP1200. The thickness of each material was 1mm and 1.5mm for DP1000 and QP1200.

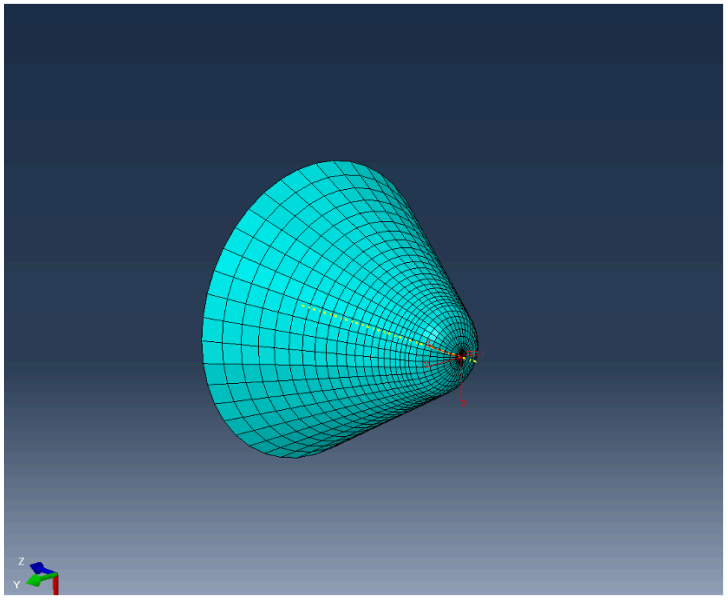


Figure 4.1.2 Spherical-conical Indenter in Abaqus

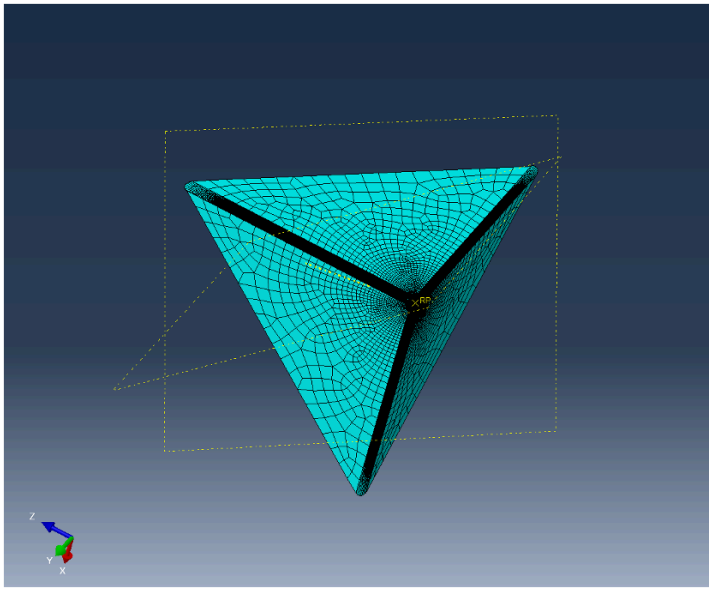


Figure 4.1.3 Cubecorner Indenter in Abaqus

The values of these variables were set using load-displacement curves, load-time curves, and depth-time curves from experimental data to build our simulation models. The following figures gives the load vs depth curves, depth vs time curves, and load vs time curves for both materials.

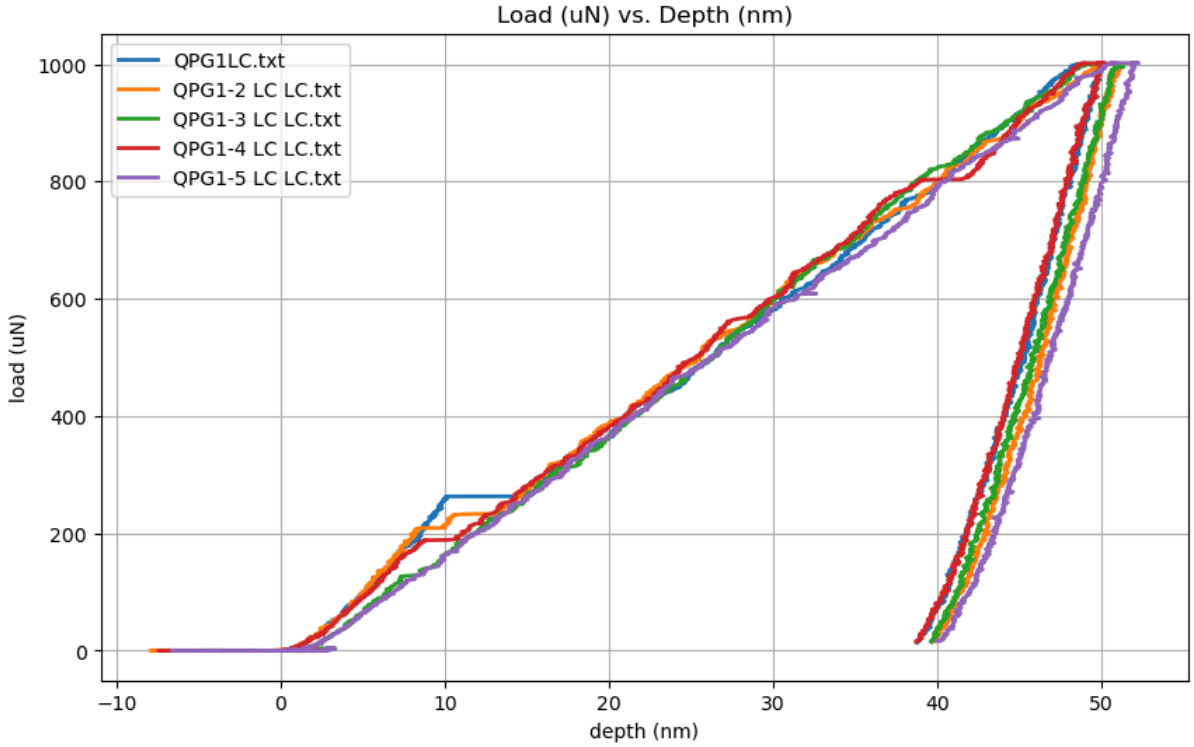


Figure 4.1.4 QP1200 Load vs Depth Curve Ferrite

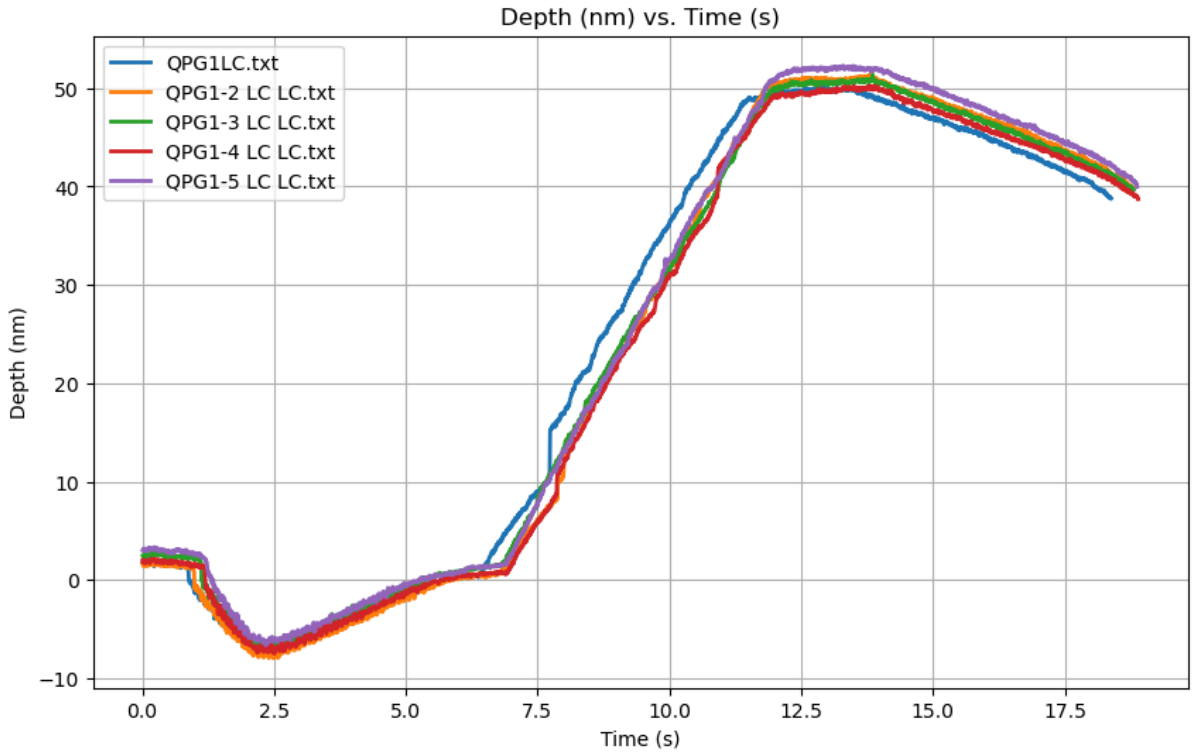


Figure 4.1.5 QP1200 Depth vs Time curve Ferrite

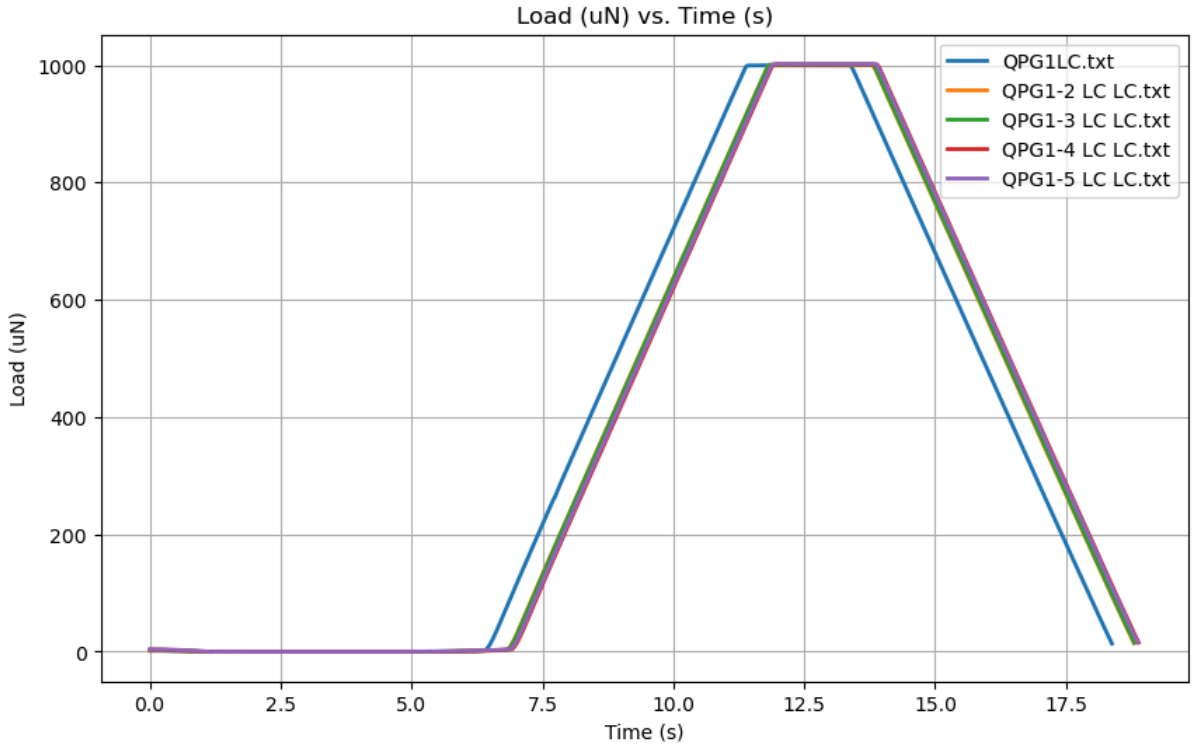


Figure 4.1.6 QP1200 Load vs Time curve Ferrite

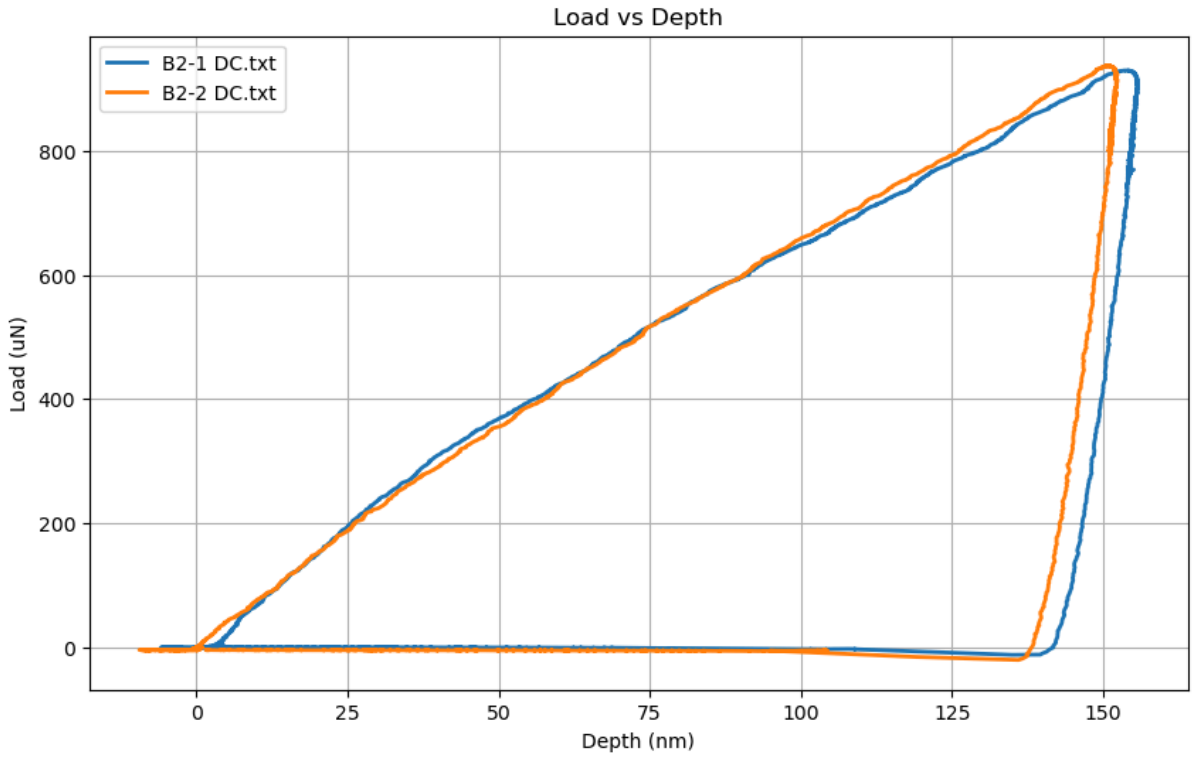


Figure 4.1.7 QP1200 Load vs Depth Curve Martensite

Out of the 5 experimental samples for QP1200 Ferrite, QPG1-2 was chosen due to its best correlation with other experimental data and smoothness. Out of the 2 experimental samples for QP1200 Martensite, B2-2 was chosen due to its max depth value closer to 150nm which was the desired value in the material thickness.

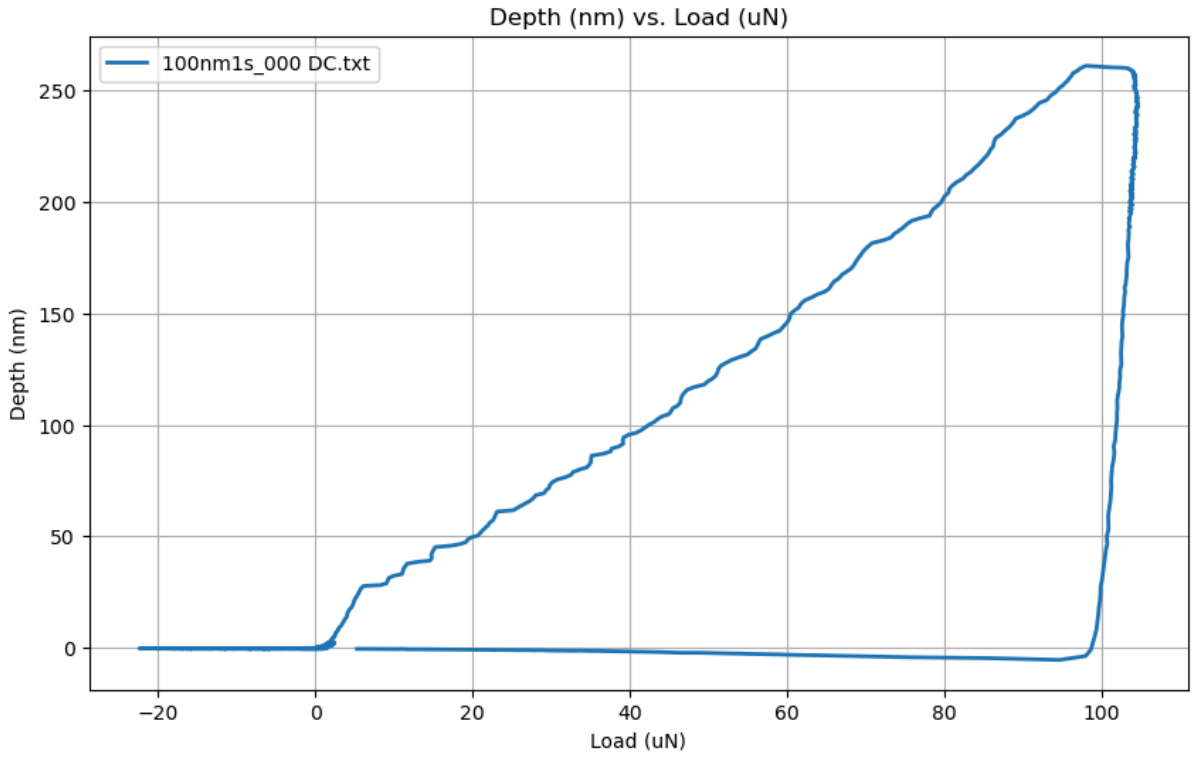


Figure 4.1.7 DP1000 Load vs Depth curve

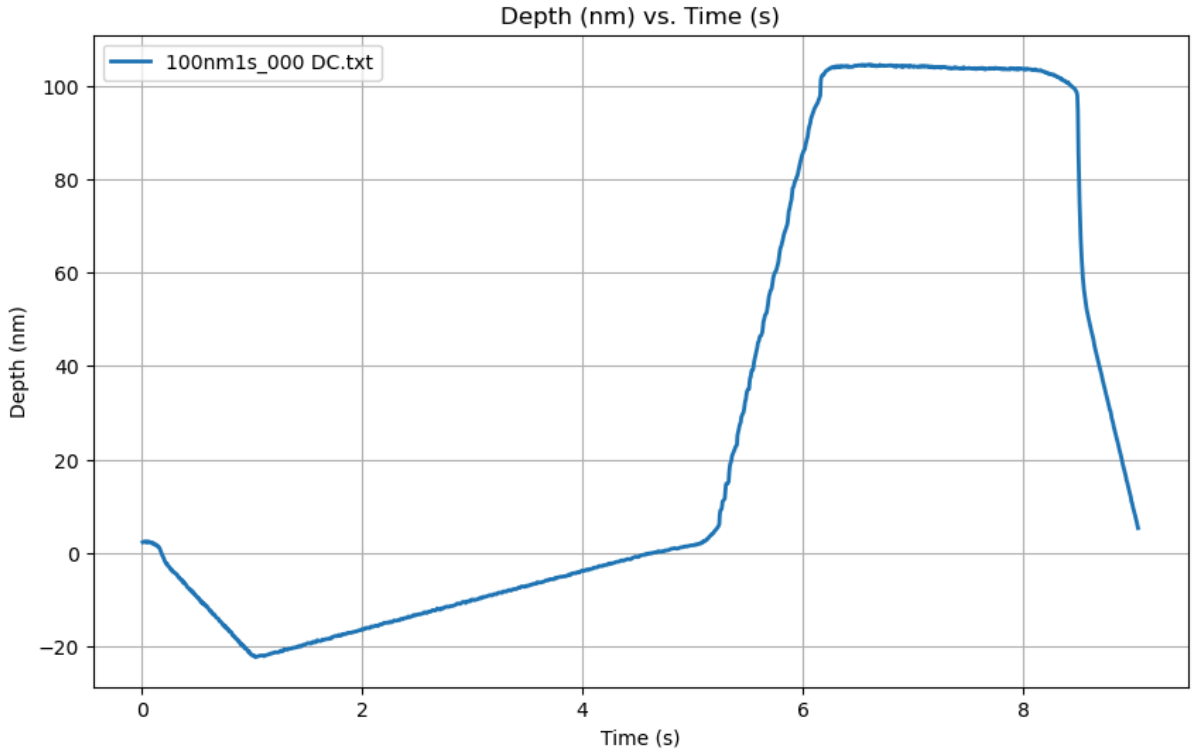


Figure 4.1.8 DP1000 Depth vs Time curve

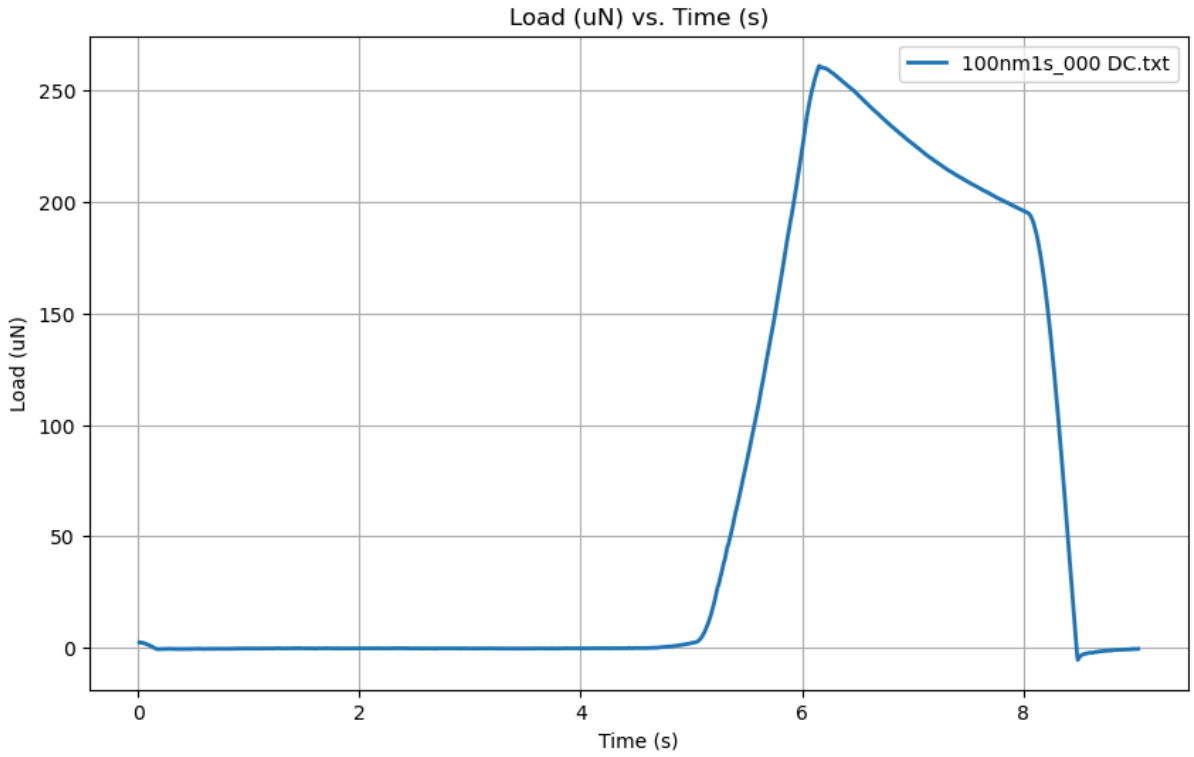


Figure 4.1.9 DP1000 Load vs Time curve

4.2 Parametric Studies Background

Nanoindentation experiments provide detailed information about the mechanical properties of materials at small scales, such as hardness, elastic modulus, and yield strength. CPFEM allows one to simulate these properties by modeling the crystal plasticity, which captures the material's microstructural response under different loading conditions. Parametric studies help understand how various parameters (e.g., hardening, slip system properties, grain orientations) affect the material's behavior during indentation.

Predictions about how a material might behave under different conditions, such as varying loads, temperatures, or strain rates, can be made by studying how various parameters affect nanoindentation responses. This predictive capability is valuable for designing materials with specific properties or for understanding how microstructural features influence macroscopic behavior. In our experiments, we carry out work by testing hardening parameters. Table 4.2.1 below describes key parameters used in this experiment.

$h\_\{\alpha\,\beta\} = q\_\{\alpha\,\beta\}\,\left[\, h\_0\,\left(1 - \frac\{\tau\_\beta\^c\}\{\tau\_s\^c\}\right)$

$\dot{\gamma}_\alpha = \dot{\gamma}_0 \left|\frac{\tau_\alpha}{\tau_\alpha^c}\right|^m$

Table 4.2.1 CP Parameter formulas and function description (try rendering the latex code and leave it empty for now if it doesn't work)

Parameter	class	Function
a	hardening	hardening parameter
h_0	hardening	hardening parameter
τ_{cs}	hardening	hardening parameter
τ_0	yielding	initial resolved shear strength
n	yielding	strain rate parameters
$\dot{\gamma}_0$	yielding	strain rate parameters

The following section presents the results from parametric studies on hardening parameters. The range of values used in our experiment is listed in Table 4.2.2. We have set 5 different boundaries of our parameter values to better fit the target curve later in the machine learning training.

Table 4.2.2

CP Parameters Range					
parameter	Min	25%	Median	75%	Max
a	0.5	1.0	1.5	2.0	2.5
h_0	2000	3000	4000	5000	6000
τ_{cs}	175	275	375	475	575
τ_0	50	125	200	275	350

Parametric Studies are required for the following reasons:

- A general range of parameters will reduce the total time for testing since it will suggest the acceptable range where the models do not break
- Obtaining a reasonable shape of stress-strain curves can be aided by an appropriate range of parameters

- CP parameters allow one to observe how each parameter affects the shape of force-displacement curves and distinguish between hardening and yielding parameters.
- Although a general range of parameters may have minor differences between materials and grains, it gives similar boundaries among common material families.

Based on these reasons, methods for identifying a general range of CP parameters can be:

- Consulting earlier literature on CP parameter testing
- Run initial simulations with increased/decreased target parameter values while fixing the others as constants.

4.3 PH law parameter study

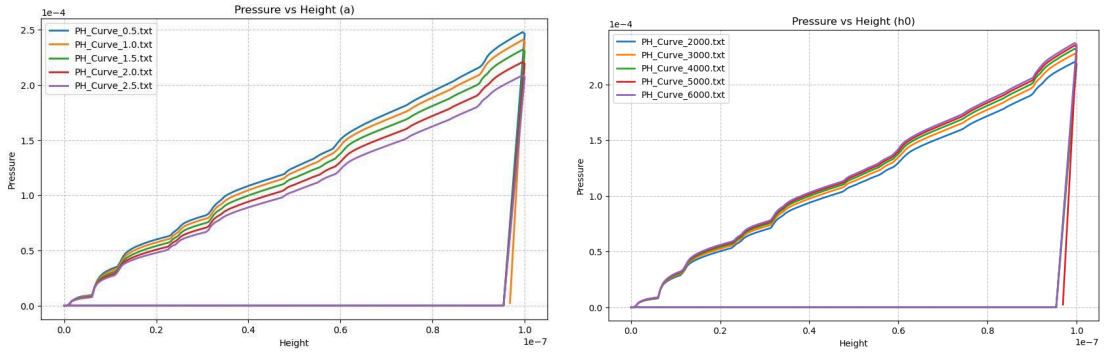


Figure 4.3.1 Parametric studies for hardening exponent a and h_0 parameters.

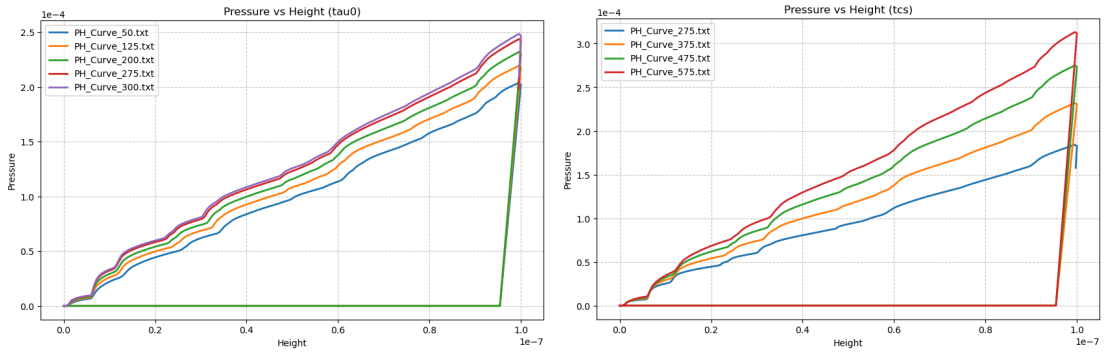


Figure 4.3.2 Parametric studies for hardening exponent τ_0 and τ_{sat} parameters.

The results above show the different behaviors of different parameters. The pressure values rise with the increase of the parameter values for h_0 ,τ_0,and τ_{sat}. Conversely,

the pressure value drops with with the increase of parameters for a. Based on these behaviors, we gave rough estimates of what the parameters may be for each material. The following tables 4.3.1, 4.3.2, and 4.3.3 have the values used in our experiments (Liu et al., 2019c).

Table 4.3.1 Initial parameters for DP1000 Ferrite

τ_0	τ_{sat}	h_0	a	(γ_0)	n	$q_{\alpha\beta}$	Q
200	375	4000	1.5	0.001	100	1.4	0.1

Table 4.3.2 Initial parameters for QP1200 Ferrite

τ_0	τ_{sat}	h_0	a	(γ_0)	n	$q_{\alpha\beta}$	Q
350	550	5000	0.45	0.0002778	20	1.4	0.1

Table 4.3.3 Initial parameters for QP1200 Martensite

τ_0	τ_{sat}	h_0	a	(γ_0)	n	$q_{\alpha\beta}$	Q
350	700	5000	0.45	0.0002778	20	1.4	0.1

4.4 Abaqus Mesh

Building the mesh were done with the following properties:

Table 4.4.1 Mesh properties for DP1000 (left) and QP1200 (right)

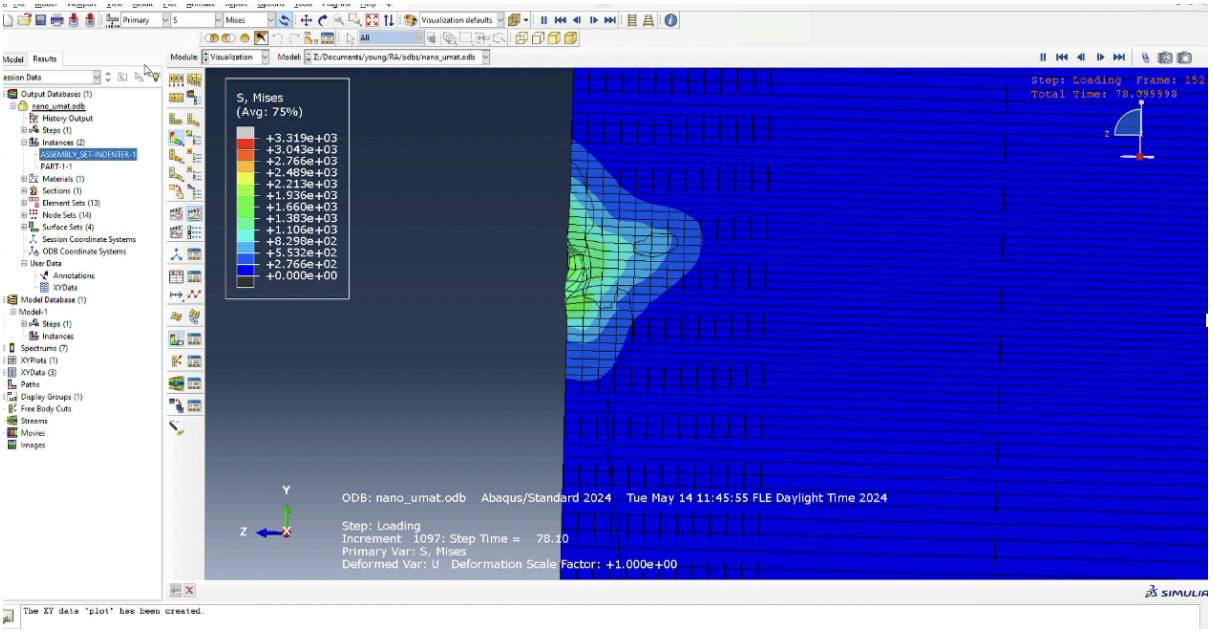
Element	C3D8
Loading Time	100
Holding Time	5
Displacement	100nm
Max Force	1000uN

Element	C3D8
Loading Time	4.985
Holding Time	1.32
Displacement	150nm
Max Force	600uN

The values have been set according to its material properties discussed earlier. One important task for the project was to build a smooth surface on the mesh to obtain a

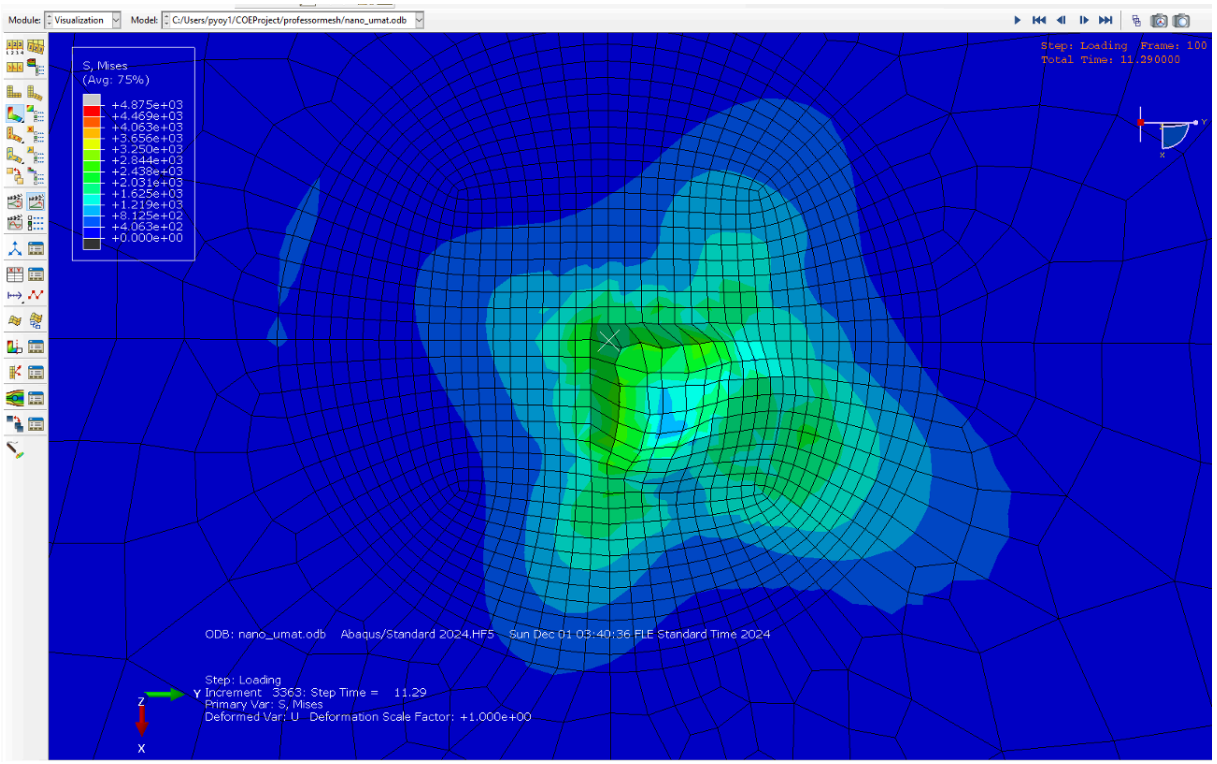
smooth curve. The cubecorner model in Abaqus remained quite coarse, therefore, did not produce a smooth force-displacement curve.

Figure 4.4.1 Cubecorner mesh results



As the above image shows, the mesh is coarse, and results do not provide a smooth mesh. However, in the spherical-conical mesh, we were able to build a smooth mesh as shown in Figure 4.4.2.

Figure 4.4.2 Mesh after indentation for Spherical-conical



4.5 Optimization and ML workflow

The workflow built in this project is separated into two parts. The first part shown in Figure 4.5.1 is summarized into processing information and preparing the required data for the machine learning training. Stage 0 reads and processes the information. Stage 1 turns the experimental data into useable forms of code. Then, in stage 2, 40 initial guesses were run to prepare the training data. If previous iterations exist, we merged them with it to increase the number of training data. If not, the initial simulations are set as the initial guess. The original workflow that was built for the project is labled in stage 3-2 which is directly applying a regression model to iteratively calibrate the parameters. However, we propose a new method in stage 3-1 that implements a loss function to predict the peak and dropping point in the force-displacement curve. Our initial results did not perform the best with the new method, therefore, further refinement was needed as shown in Figure 4.5.2

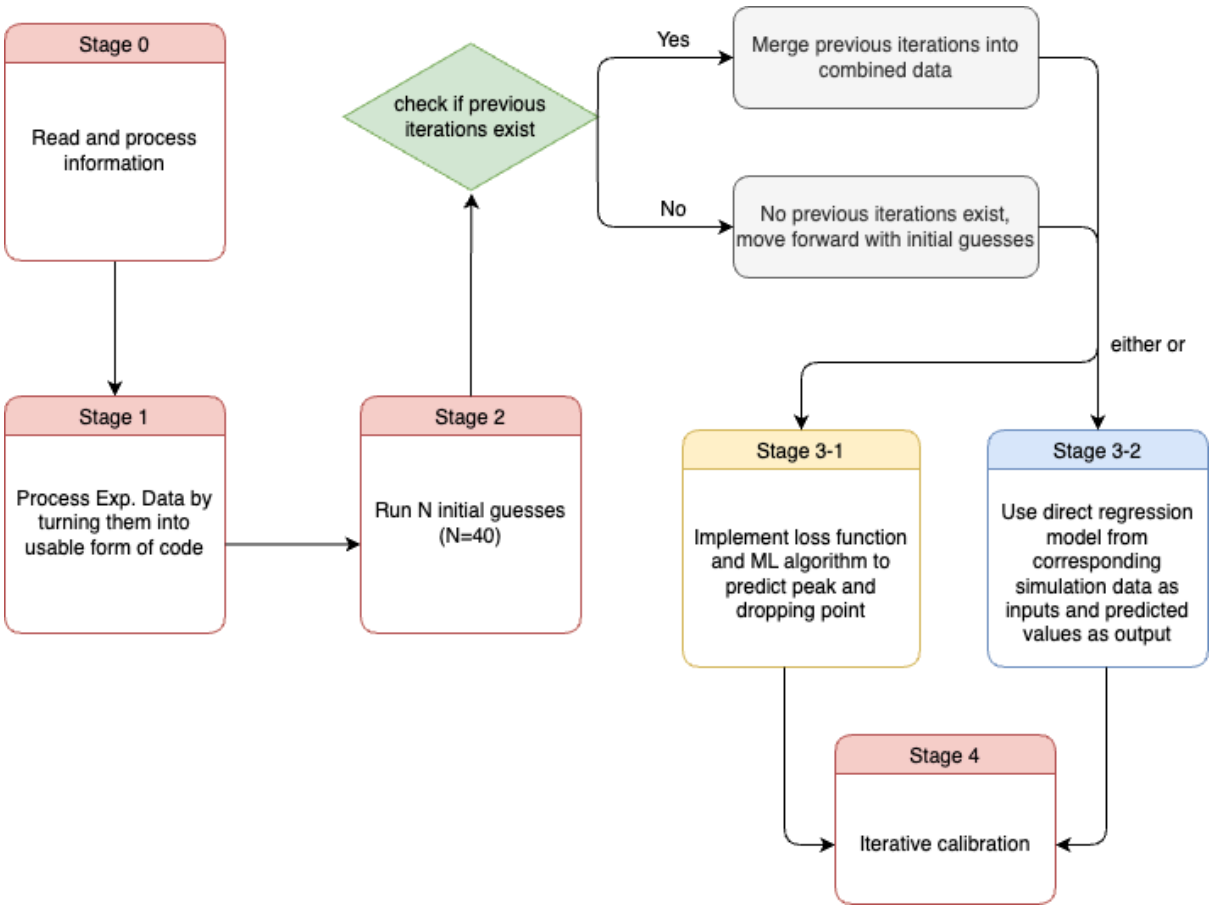


Figure 4.5.1 Optimization and ML workflow Part 1

In the second part, our proposal was to add 3 more predicting points: 25th, 50th, 75th quantile in the displacement direction of the data to train our machine learning

model shown in stage 4-1. After running with the added points the results have significantly improved which is shown in chapter 5. Overall, the newly suggest workflow provides an accurate interpretation of the parameters needed for this experiment.

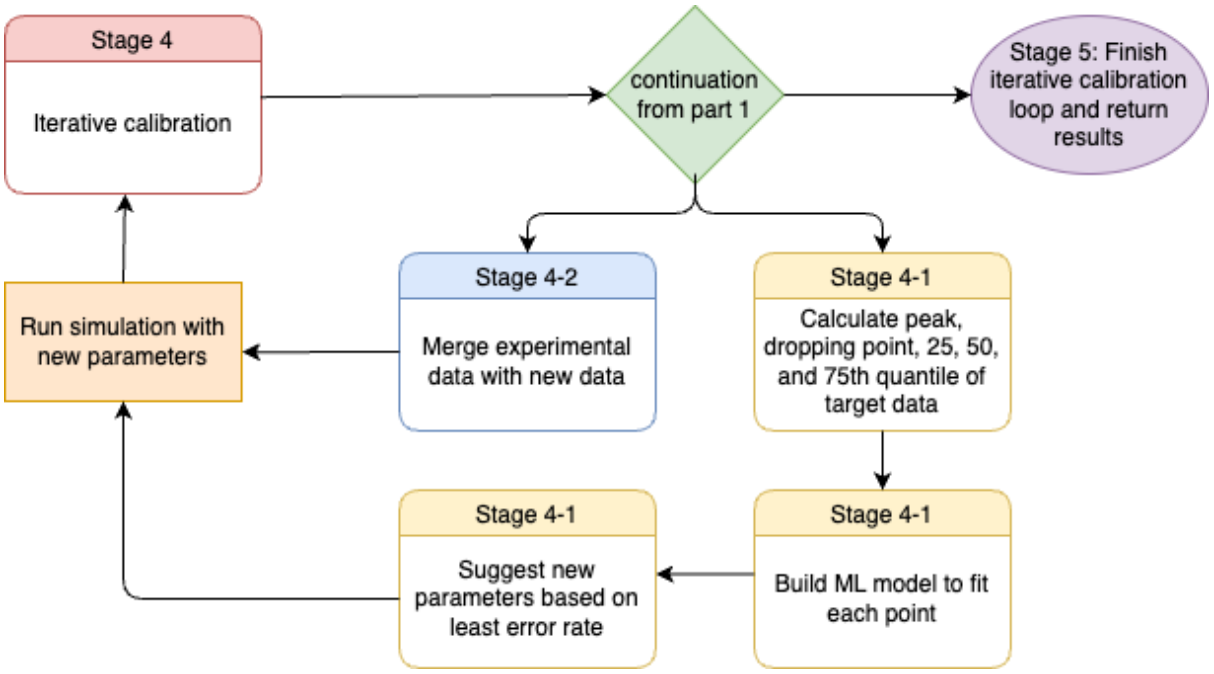


Figure 4.5.2 Optimization and ML workflow Part 2

5. Machine Learning

5.1 Introduction

Crystal Plasticity Finite Element Models (CPFEM) are needed to figure out how a material reacts under stress. However, in order to use these models, the material parameters need to be calibrated and this calibration process is based on traditional, manual curve-fitting techniques. Therefore, in general, it can be said that this method is ineffective, time-consuming, and prone to human error. One solution to make this calibration process more optimized is using machine learning (ML) models. ML methods can learn from existing data and help automate this process. By doing so, this calibration process can be more time-efficient, accurate, and effective, which means the whole CPFEM process would be more efficient.

In this project, we collected nanoindentation test data and used that data to calibrate CPFEM parameters. Our goal was to develop an ML algorithm to automate this process. Firstly, we ran the nanoindentation simulations on CSC, and as a result, we got several force-displacement (FD) curves. These curves gave us important information about the material properties, such as hardness and yield strength. After that, we gave these curves to an already trained ML algorithm, and as an output, we aimed to receive the material parameters. The parameters we wanted to receive were initial resolved shear stress (tau_0), hardening exponent (a), hardening rate (h0), and critical shear stress (tcs). After that process, our final goal was to get these parameters with high precision, while decreasing the time spent.

The flow chart given below in *Figure 5.1* shows the workflow we followed in this project from the start to the end. Firstly, the workflow begins with **parametric studies**. That represents the section where we run many tests with different parameters to collect data so that we can train the ML algorithm. In this section, it is important to note that the parameter ranges were taken based on experimental data and literature. After that, the collected data goes through **data preparation**. In this stage, the data gets cleaned and normalized. After that, there is the **feature engineering** stage. In this section, the data is simplified and the number of features is decided. The overall goal here is to reduce the complexity of the final ML algorithm. The next stage is **algorithm testing**. In this section, different ML models

are trained and we selected one that can predict the CPFEM parameters with the least error. After deciding on the algorithm, we moved on to **material-specific training**. In this section, the model is trained with individual materials. The materials used in this paper are DP1000, QP1200 (Ferrite), and QP1200 (Martensite). The last step is **iterative optimization.** In this section, the goal is to analyze the performance of our ML algorithm and develop it more to further optimize the process.

In the following subsections of this section, we will go through every stage of the chart in detail. Our main goal is to explain the methodology, results, and limitations thoroughly.

Figure 5.1: ML Flow Chart

5.2 Data Preparation and Feature Engineering

The training data quality and feature engineering play a critical role in the accuracy of ML algorithms. Therefore, after collecting our data from the parametric studies, our first step was creating a quality dataset. Our main goal in this step was to make sure that our training dataset was accurate and consistent.

5.2.1 Raw Data Collection and Preprocessing

We collected our raw experimental data from the parametric studies on DP1000, QP1200 (Ferrite), and QP1200 (Martensite). The parametric studies provided us with FD curves. These curves contain important information about the mechanical properties of the materials such as hardness, elastic modulus, and deformation. However, this raw data is not directly usable for ML algorithms since they contain inconsistencies, noise, and variations. In order to solve this problem and get a dataset suitable for an ML algorithm, we used the following data preprocessing methods.

- **Normalization:** The dataset should have the same scaling so that different sessions can be compared. To do that, we normalized all data to a [0,1] range using Min-Max Scaling.
- **Outlier Removal:** Outliers affect the ML algorithms heavily. Therefore, we cleaned the dataset using the interquartile range (IQR) method. This method removed outliers while keeping the important data. In hardening rate (h0) and critical shear stress (tcs), many outliers were observed.
- **Missing Data Handling:** Missing values can confuse the ML algorithms. Therefore, we replaced them with the mean of the respective feature. This way we maintained the integrity of the dataset while avoiding data loss.

After these data preprocessing steps, we ensured that the dataset could be used for ML algorithm training and produce accurate results.

5.2.2 Feature Engineering

Feature engineering is the process of selecting the important features and outputs while eliminating the non-necessary ones. By doing that, the dimensionality of the dataset can be decreased, increasing the overall efficiency of the ML algorithm.

We carefully selected the features to represent the materials' physical properties, and they are shown below.

- **Key Parameters:** Initial resolved shear stress (tau0), hardening exponent (a), hardening rate (h0), and critical shear stress (tcs) were chosen for their direct influence on FD curve characteristics.
- **Eliminated Parameters:** Initial shear modulus (g0), rate parameter (m), and shear rate were eliminated to decrease the dimensionality of the dataset. Our goal was to add them back after getting the first accurate ML algorithm.

In terms of outputs, we tried to capture critical points to get a better fit of the graph. The process is shown below.

- **Peak and Drop Points:** Peak and drop points are critical for our study since they are key indicators of material behavior. Therefore, we selected them as output variables.

- **Percentile-Based Metrics:** The 25th, 50th, and 75th percentiles of the FD curve give us information about each stage of the process. Therefore, we added them as output variables to increase the accuracy of the ML algorithm.

At the end of this process, we achieved a dataset that has only the important features and outputs. This increases the accuracy and efficiency of the ML algorithm.

5.2.3 Structured Input-Output Framework

After the data preprocessing, feature engineering, and output engineering, we got our final training dataset. This dataset is summarized in *Table 5.2.1*, which shows every feature and output variable.

Input Features	Output Variables
Initial Resolved Shear Stress (tau0)	Peaking Point
Hardening Exponent (a)	Dropping Point
Hardening Rate (h0)	25th Percentile Point
Critical Shear Stress (tcs)	50th Percentile Point
Initial Shear Modulus (g0)	75th Percentile Point
Rate Parameter (m)	
Shear Rate	

Table 5.2: Dataset Properties

5.3 Algorithm Selection and Testing

After preparing our final dataset, we needed to select the ML algorithm that suited our dataset the best. This was a very critical step in this study since every ML algorithm has different strengths and weaknesses. We tested several ML models, and in this section, we will discuss the specifics of the ML models that we tested, the error values we got, and the final ML algorithm we chose.

5.3.1 Models Evaluated

In total, we trained four ML models and we selected them according to their strengths and weaknesses. They are shown below.

1. **Artificial Neural Networks (ANNs):** ANNs are flexible models, and they can learn complex, nonlinear relationships. The reason we chose ANNs to test is because of their flexible nature.
2. **Long Short-Term Memory Networks (LSTMs):** LSTMs work well with sequential or time-series data. LSTMs can help with the sequential data in FD curves, and that's the reason we chose to test them.
3. **Convolutional Neural Networks (CNNs):** CNNs are mainly used for pattern recognition. However, CNNs with 1D convolutions can potentially detect localized features, and that's the reason we tested CNNs.
4. **Random Forest Regressors (RFs):** RFs are made out of multiple decision trees. They work well with structured data and that's the reason we tested it.

After choosing the ML models to test, we wrote the code for every one of them.

5.3.2 Model Evaluation Metrics

After that, we needed to find a common evaluation method to be able to compare the results of the ML models. The metric chosen is shown below.

- **Root Mean Square Error (RMSE):** RMSE is used to measure the error of the predicted outputs. We used RMSE and not MSE, because our values are relatively small and RMSE can provide us with error values that have the same unit as our parameters. This makes understanding and analyzing the error values easier.

5.3.3 Results of Model Comparison

After deciding on the models and the error metric, we ran the ML algorithm and observed the results. The performance of them is summarized in *Table 5.3.1*. The table shows the accuracy of each method for different points, and also, the computational efficiency of each model.

ML Model	Peak Point RMSE	Drop Point RMSE	Training Time
LSTM	0.483	0.959	High
CNN	0.246	0.842	Moderate
RF	0.454	0.915	Low
ANN	0.283	0.704	Moderate

Table 5.3.1: Model Comparison

According to the table given above, the ANN model has the lowest mean error values and the training time is moderate. From these results, we can conclude that our dataset has a complex structure and the relationships between the inputs and outputs are not linear. As a result, we can confidently choose ANN to continue our study with.

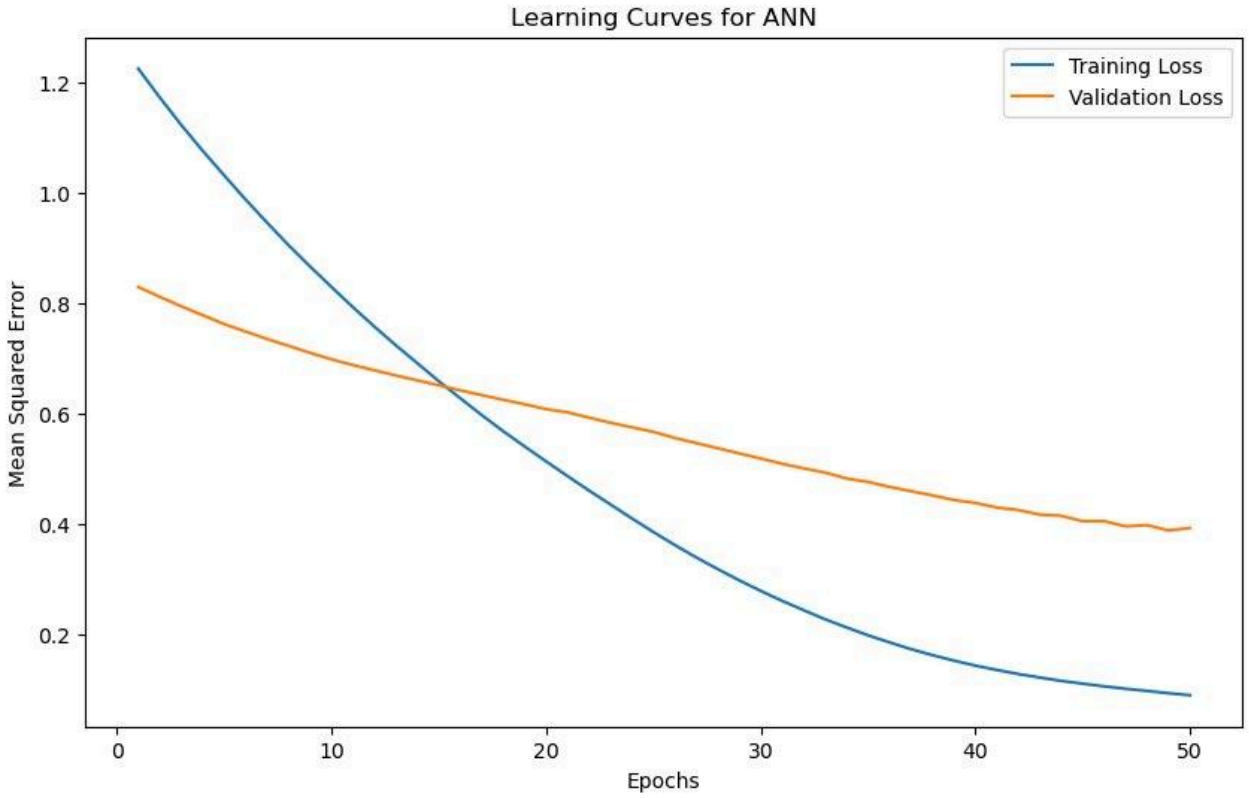


Figure 5.3.2: Learning Curves for ANN

Figure 5.3.2 given above shows the learning curves for ANN. The blue line represents the training loss and the orange line represents the validation loss. We can see that as the model trains more, both the training and validation loss goes down. The decreasing training loss means that the ANN model is able to capture the trends in our training data very well. In addition, the fact that the validation loss is stabilizing at a low value means that the ANN model also works well with unseen data. We can see that the gap between the two curves gets closer as we train the model more, this means there is minor overfitting in our results. This graph shows us that the ANN model is able to predict the trends in our dataset very accurately, and therefore, fits very well to our dataset. So as a result, we can conclude that ANN is the most effcetive model for oır studies.

5.4 Material-Specific Calibration

After choosing our machine learning model, we moved into training it for our spesific materials. We trained our ANN model seperatly with three different material datasets. These are DP1000, QP1200 (Ferrite), and QP1200 (Martensite). Each training dataset had on average 50 datapoints. After training our ANN algorithm for each one of the materials, we gave the dataset of an experimental curve that we got from previous papers to the ANN algorithm. As an output, we expected the parameters that would give that experimental curve. Finally, after we got the predicted parameters, we ran the simulation with those parameters and compared the predicted curve that we got with the experiemantal one to evaluate the performance of our ANN model.

5.4.1 Calibration for DP1000

The first material we tested was DP1000. It is a dual-phase steel consisting of Ferrite and Martensite. Due to this composition, it has unique mechanical properties. The ANN predicted parameters for DP1000 is given below.

- tau0 = 215.0
- a = 1.5
- h0 = 4000.0
- tcs = 395.0

These parameter values are promising since they are in the range that we expected them to be in from the experimental curve that we got from academic papers.

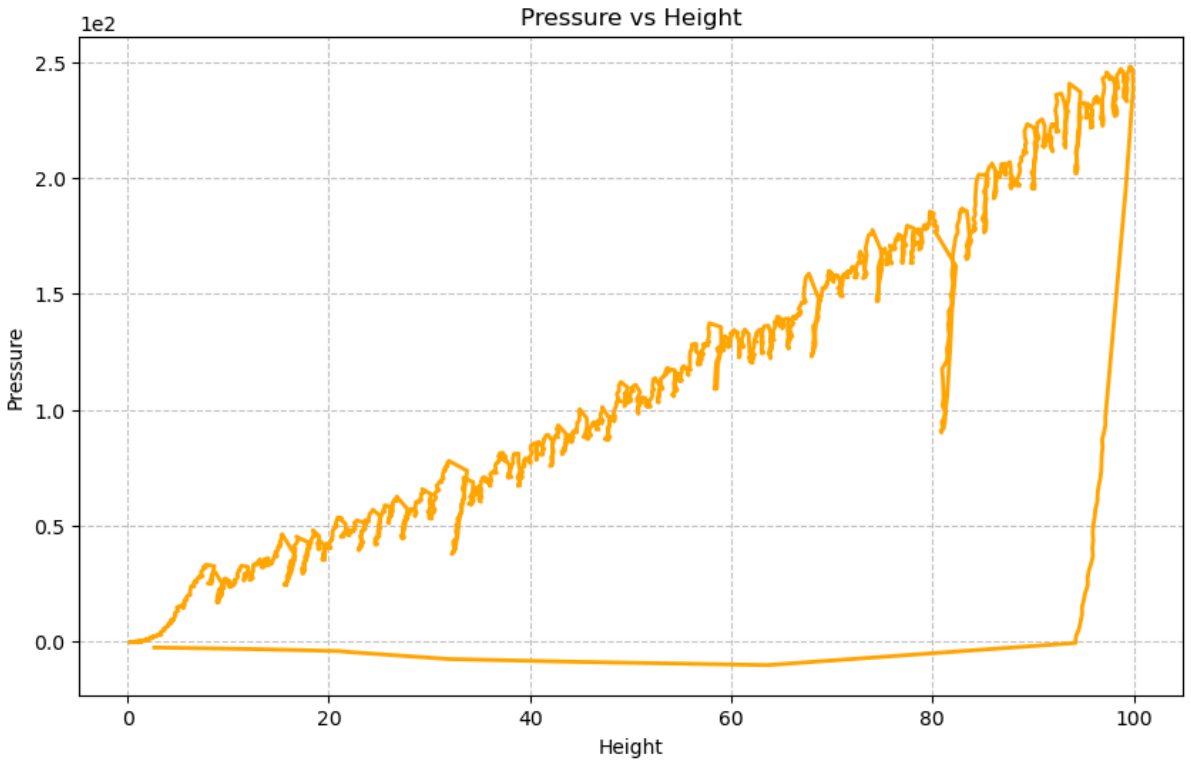


Figure 5.4.1: DP1000 Thereotical Curve

The theoretical FD curve for DP1000 is shown above in ***Figure 5.4.1***. This graph represents the expected response of the material when it is nanoindented. We will use this curve to compare it with our ML predicted curve. The important characteristics of the curve is given below.

- **Linear Loading Phase:** We can see that the curve follows a straight line at the start. In this section, the force increases linearly with displacement, showing the elastic behavior of the material.
- **Plastic Deformation Phase:** After the first straight line segment, the curve starts to get curvy as it gets closer to the peak point. This section represents the plastic deformation.
- **Unloading Phase:** After the peak point, the load is slowly removed. At that stage, we can see the elastic recovery of the material. This gives us information about the stiffness and resilience of the material.

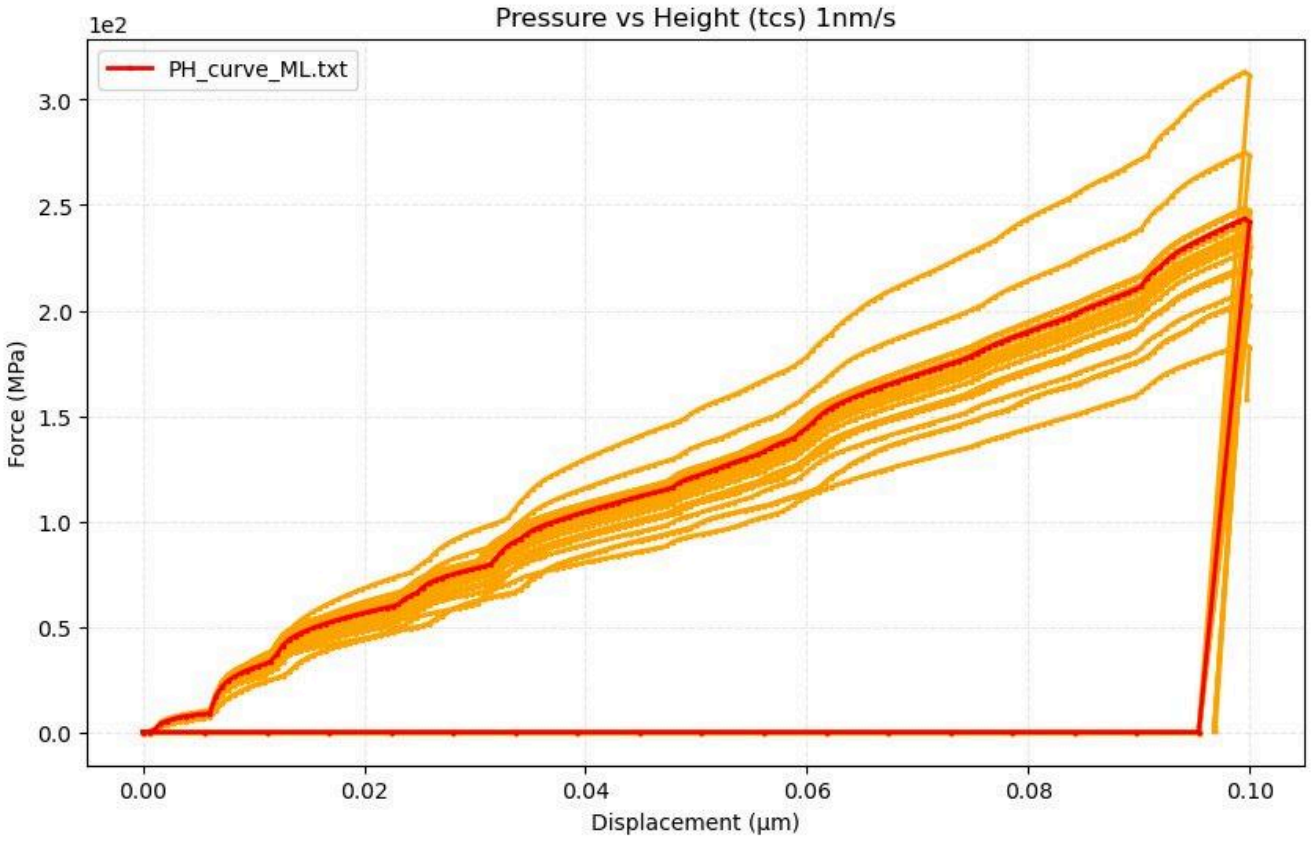


Figure 5.4.2: DP1000 Predicted Curve

The predicted FD curve for DP1000 is shown above in ***Figure 5.4.2***. The red line represents the curve that the ANN algorithm predicted and the other lines represents the other possible curves that the ANN algorithm considered. The optimal curve found is similar with the theoretical curve, which is promising. The important characteristics of the curve is given below.

- **Loading and Deformation Phase:** Just like the theoritacal curve, the predicted curve also has the linear loading phase where the force and the displacement is linearly related. Similarly, the predicted curve also has a plastic deformation phase following the linear loading phase, which is a very promising similarity.
- **Unloading Phase:** The unloading phase of the curves are really close. They have similar slope values after the peak point. This shows the accuracy of the ANN model.

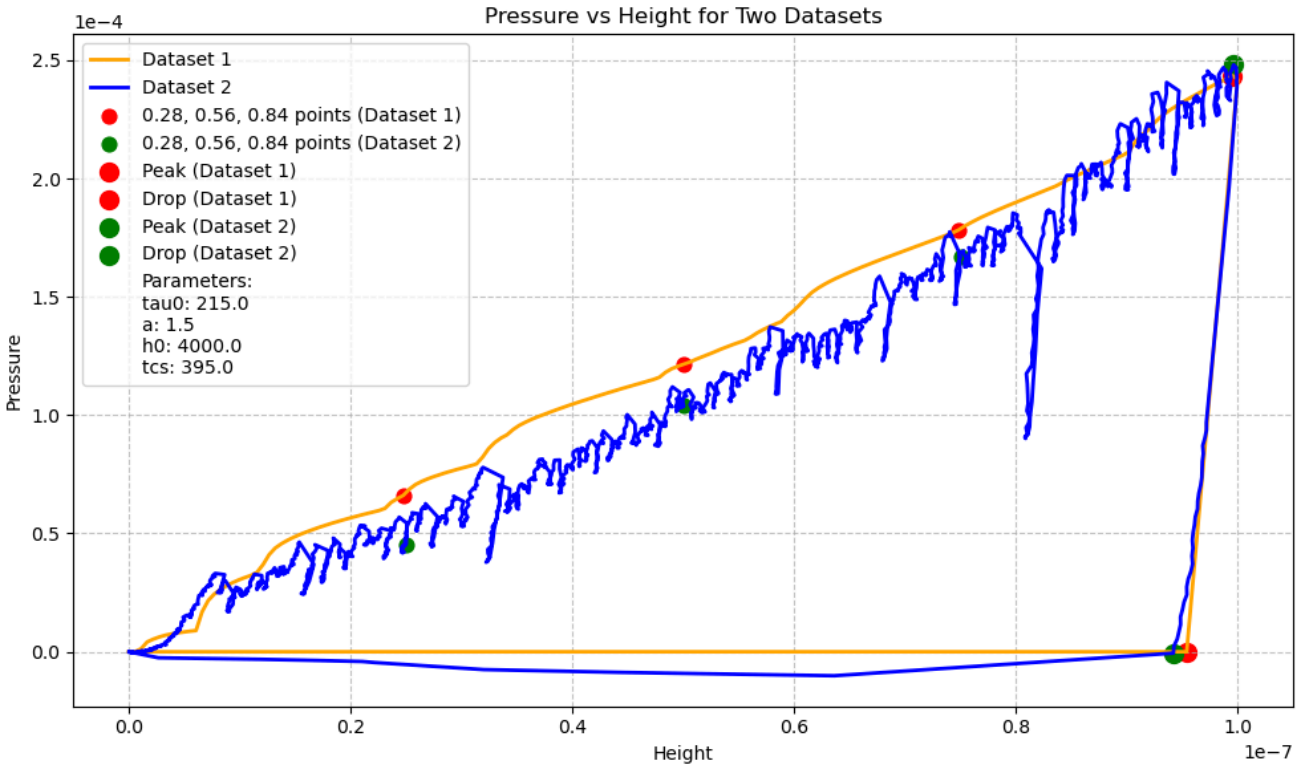


Figure 5.4.3: DP1000 Comparison

The comparison of the theoretical and predicted curves is given above in ***Figure 5.4.3.*** Dataset 1 represents the predicted curve and dataset 2 represents the theoretical curve. the point along the line corresppons to height values of 0.28, 0.56, and 0.84. The peak, dropping and percentile points are shown with markers. The details can be seen from the table shown in the leftside of the graph. By looking at the figure, we can say that the curves align very well and the ML algorithm is very promising.

After plotting the graph, we calculated several error metrics to evaluate the performance. The results are shown below.

- **Root Mean Square Error (RMSE):** 1.331e-05
- **R-Squared (R^2) Value:** 0.975

We can see that the RMSE value is extremely small, and the R^2 value is 97.5%. A low RMSE value means that the points that we chose along the curve aligns really well and a high R^2 percentage means that the curves overlap. From these values, we can conclude that the ANN suits our dataset well and the prediction is accurate.

5.4.2 Calibration for QP1200 (Ferrite)

The second material we tested was QP1200. This material is a complex dual-phase microstructure. It consists of Ferrite and Martensite, but in this section, we will focus on the Ferrite phase, which is known its ductility and low strength. The ANN predicted parameters for QP1200 Ferrite is given below.

- tau0 = 345.0
- a = 0.79
- h0 = 5906.25
- tcs = 542.81

These parameter values are promising since they are in the range that we expected them to be in from the experimental curve that we got from academic papers.

The theoretical FD curve for QP1200 Ferrite is shown below in ***Figure 5.4.4***. This graph represents the expected response of the material when it is nanoindented. We will use this curve to compare it with our ML predicted curve. The important characteristics of the curve is given below.

- **Linear Loading Phase:** We can see that the curve follows a straight line at the start. In this section, the force increases linearly with displacement, showing the elastic behavior of the material.
- **Plastic Deformation Phase:** After the first straight line segment, the curve starts to get curvy as it gets closer to the peak point. This section represents the typical hardening behaivour of Ferrite.
- **Unloading Phase:** After the peak point, the load is slowly removed. At that stage, we can see the elastic recovery of the material. This gives us information about the stiffness and resilience of the material.

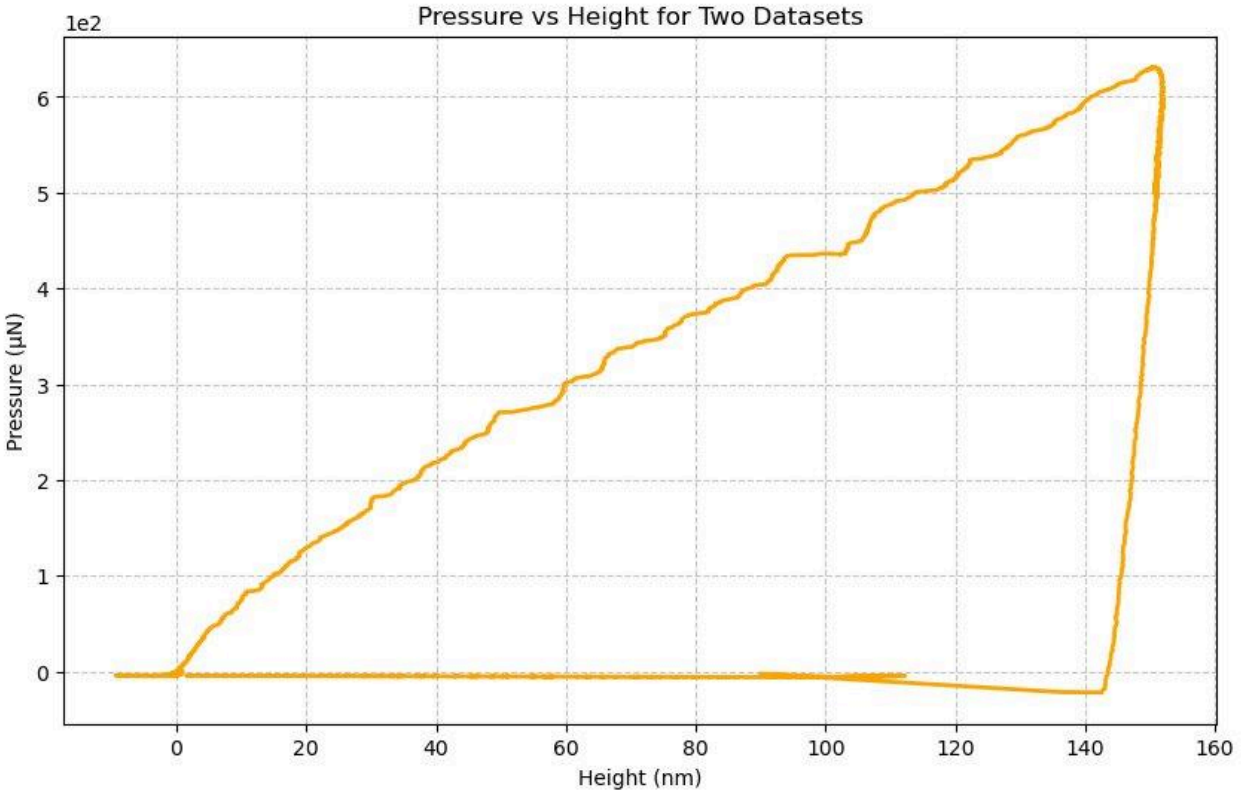


Figure 5.4.4: QP1200 Ferrite Thereotical Curve

The predicted FD curve for QP1200 Ferrite is shown below in ***Figure 5.4.5***. The red line represents the curve that the ANN algorithm predicted and the other lines represents the other possible curves that the ANN algorithm considered. The optimal curve found is similar with the theoretical curve, which is promising. The important characteristics of the curve is given below.

- **Loading and Deformation Phase:** Just like the theoritacal curve, the predicted curve also has the linear loading phase where the force and the displacement is linearly related. Similarly, the predicted curve also has a plastic deformation phase following the linear loading phase, which is a very promising similarity.
- **Unloading Phase:** The unloading phase of the curves show some slight deviations. However, they still have similar slope values after the peak point. This shows the accuracy of the ANN model.

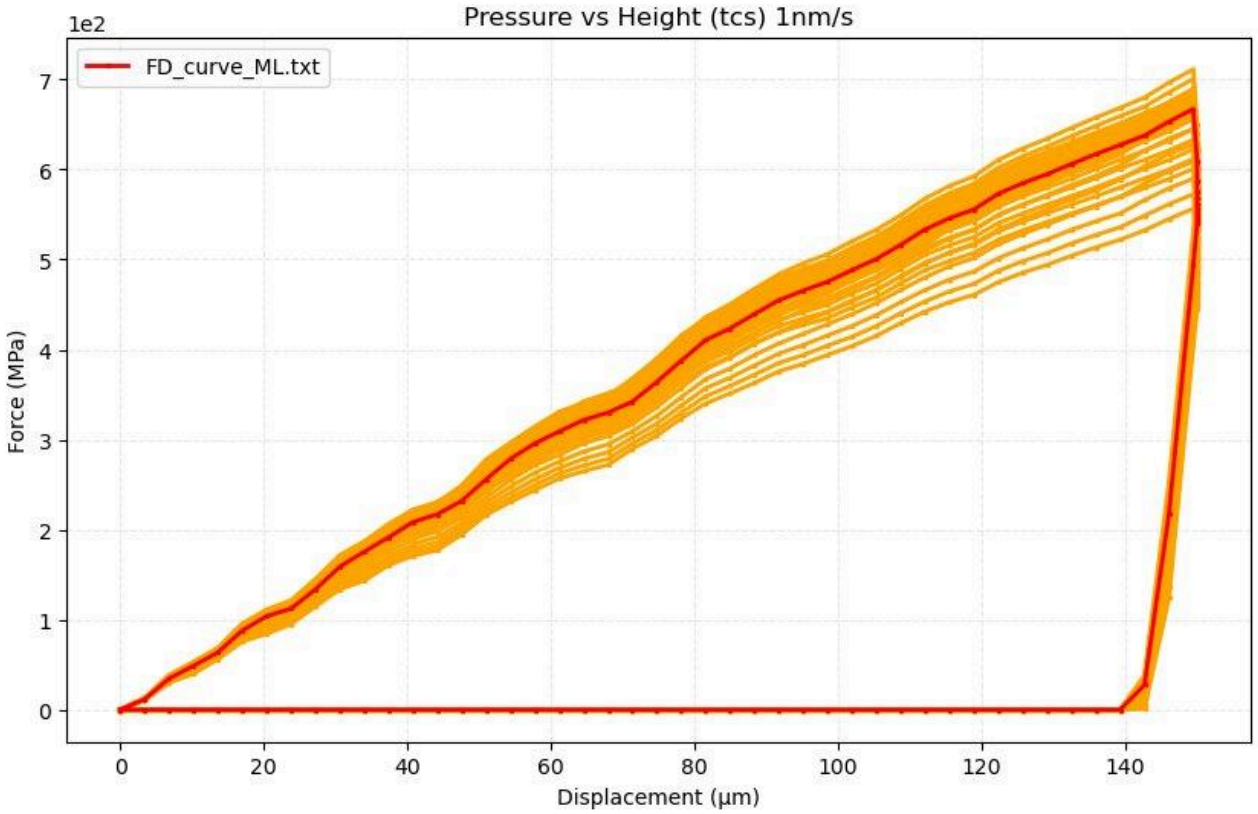


Figure 5.4.5: QP1200 Ferrite Predicted Curve

The comparison of the theoretical and predicted curves is given below in ***Figure 5.4.6.*** Dataset 1 represents the predicted curve and dataset 2 represents the theoretical curve. the point along the line corresppons to height values of 0.4, 0.8, and 1.2. The peak, dropping and percentile points are shown with markers. The details can be seen from the table shown in the leftside of the graph. By looking at the figure, we can say that the slope of the curves are slightly different but they are still not too far apart.

After plotting the graph, we calculated several error metrics to evaluate the performance. The results are shown below.

- **Root Mean Square Error (RMSE):** 3.128e-05
- **R-Squared (R^2) Value:** 0.983

We can see that the RMSE value is extremely small, and the R^2 value is 98.3%. A low RMSE value means that the points that we chose along the curve aligns really

well and a high R^2 percentage means that the curves overlap. From these values, we can conclude that the ANN suits our dataset well and the prediction is accurate.

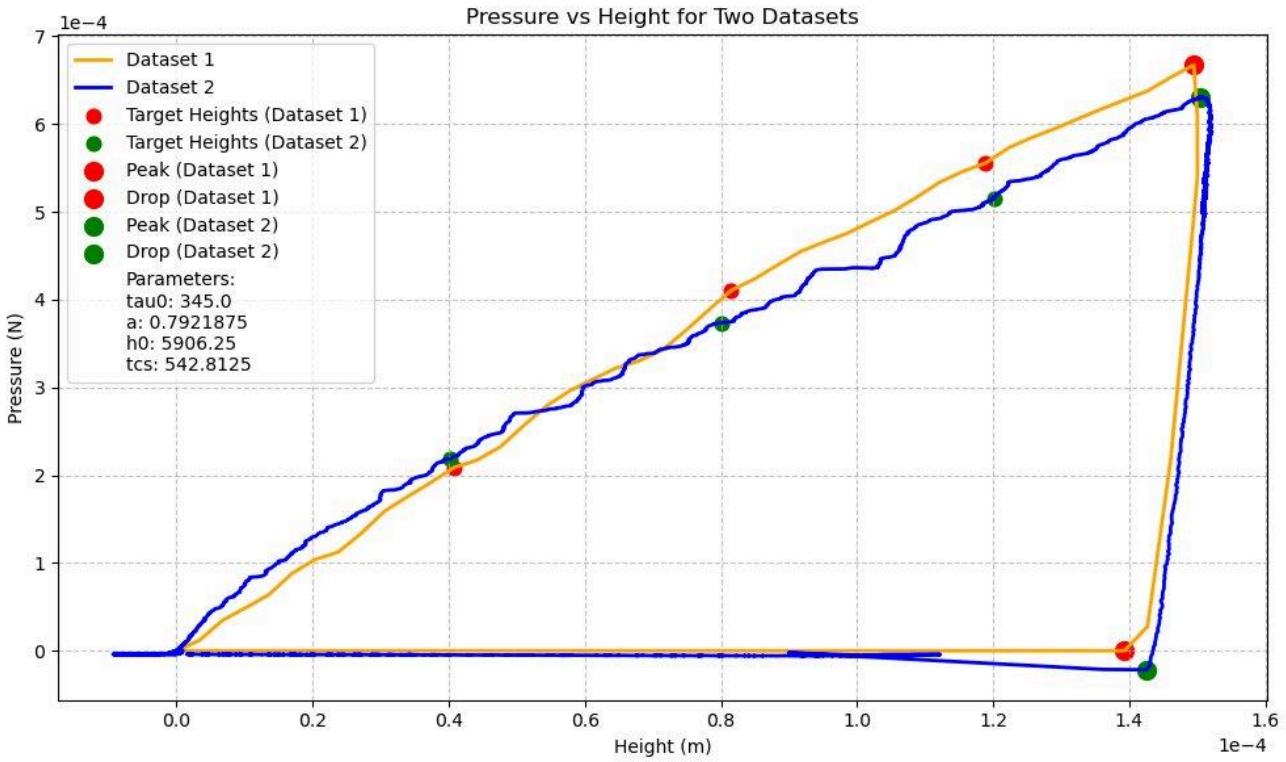


Figure 5.4.6: QP1200 Ferrite Comparison

5.4.2 Calibration for QP1200 (Martensite)

After the QP1200 Ferrite, we moved onto QP1200 Martensite. Unlike QP1200 Ferrite, this form of QP1200 is known for its high strength and brittleness. The ANN predicted parameters for QP1200 Martensite is given below.

- tau0 = 366.071429
- a = 0.825
- h0 = 10625
- tcs = 723.214286

These parameter values are promising since they are in the range that we expected them to be in from the experimental curve that we got from academic papers.

The theoretical FD curve for QP1200 Martensite is shown below in ***Figure 5.4.7***. This graph represents the expected response of the material when it is nanoindented. We will use this curve to compare it with our ML predicted curve. The important characteristics of the curve is given below.

- **Linear Loading Phase:** Compared to other materials, this curve follows a steeper straight line at the start. This shows how high the initial stiffness of Martensite is.
- **Plastic Deformation Phase:** After the first straight line segment, the linearity starts to change. This section represents the limited ductility of Martensite.
- **Unloading Phase:** At the unloading stage, even though it is still elastic, we see a lower recovery rate compared to Ferrite. This shows the Martensite's brittle characteristics.

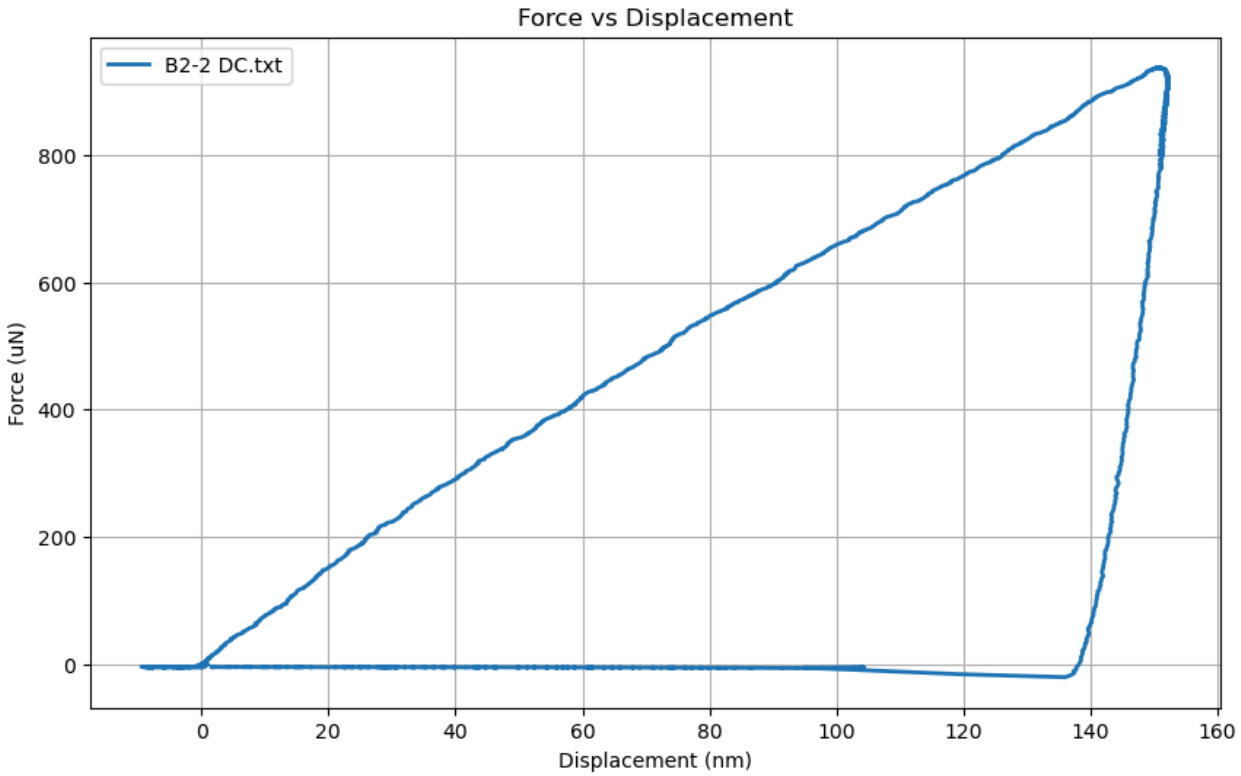


Figure 5.4.7: QP1200 Martensite Thereotical Curve

The predicted FD curve for QP1200 Martensite is shown below in ***Figure 5.4.8***. The red line represents the curve that the ANN algorithm predicted and the other lines represents the other possible curves that the ANN algorithm considered. The optimal

curve found is similar with the theoretical curve, which is promising. The important characteristics of the curve is given below.

- **Loading and Deformation Phase:** Just like the theoritacal curve, the predicted curve also has the linear loading phase where the force and the displacement is linearly related. Similarly, the ANN algorithm predicted the linear to exponential change segment really well.
- **Unloading Phase:** The unloading phase of the curves show some slight deviations. However, they still have similar slope values after the peak point. This shows that further improvements can be done to capture the brittle behavior better.

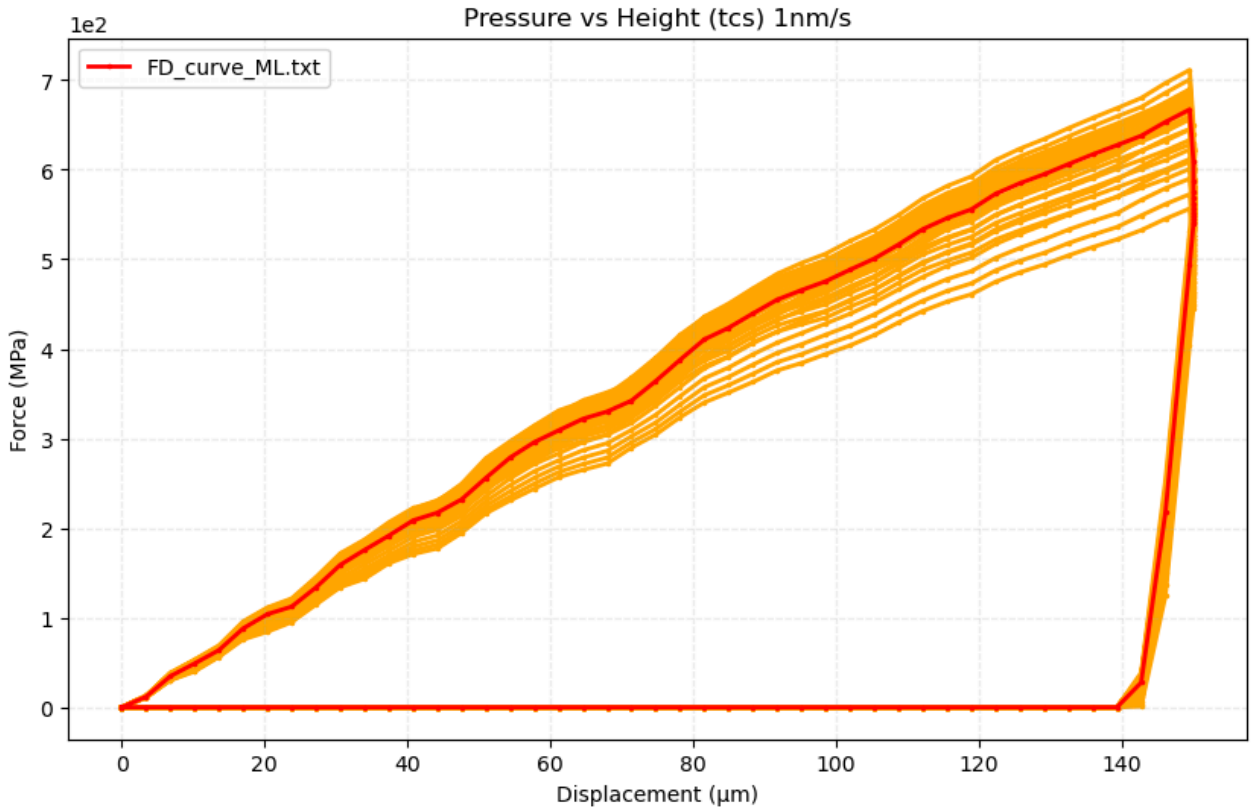


Figure 5.4.8: QP1200 Martensite Predicted Curve

The comparison of the theoretical and predicted curves is given below in ***Figure 5.4.9.*** Just like the other graphs, dataset 1 represents the predicted curve and dataset 2 represents the theoretical curve. the point along the line corresppons to height values of 0.4, 0.8, and 1.2. The peak, dropping and percentile points are shown with markers. The details can be seen from the table shown in the leftside of

the graph. By looking at the figure, we can say that our ANN algorithm worked the best with QP1200 Martensite.

After plotting the graph, we calculated several error metrics to evaluate the performance. The results are shown below.

- **Root Mean Square Error (RMSE):** 2.228e-05
- **R-Squared (R^2) Value:** 0.995

We can see that the RMSE value is extremely small, even smaller than the Ferrite graph. In addition, QP1200 Martensite has the lowest R^2 value we got so far, which is 99.5%. A low RMSE value means that the points that we chose along the curve aligns really well and a high R^2 percentage means that the curves overlap. The results suggest that while the ANN performs well overall, additional data and refinements could further enhance its accuracy.

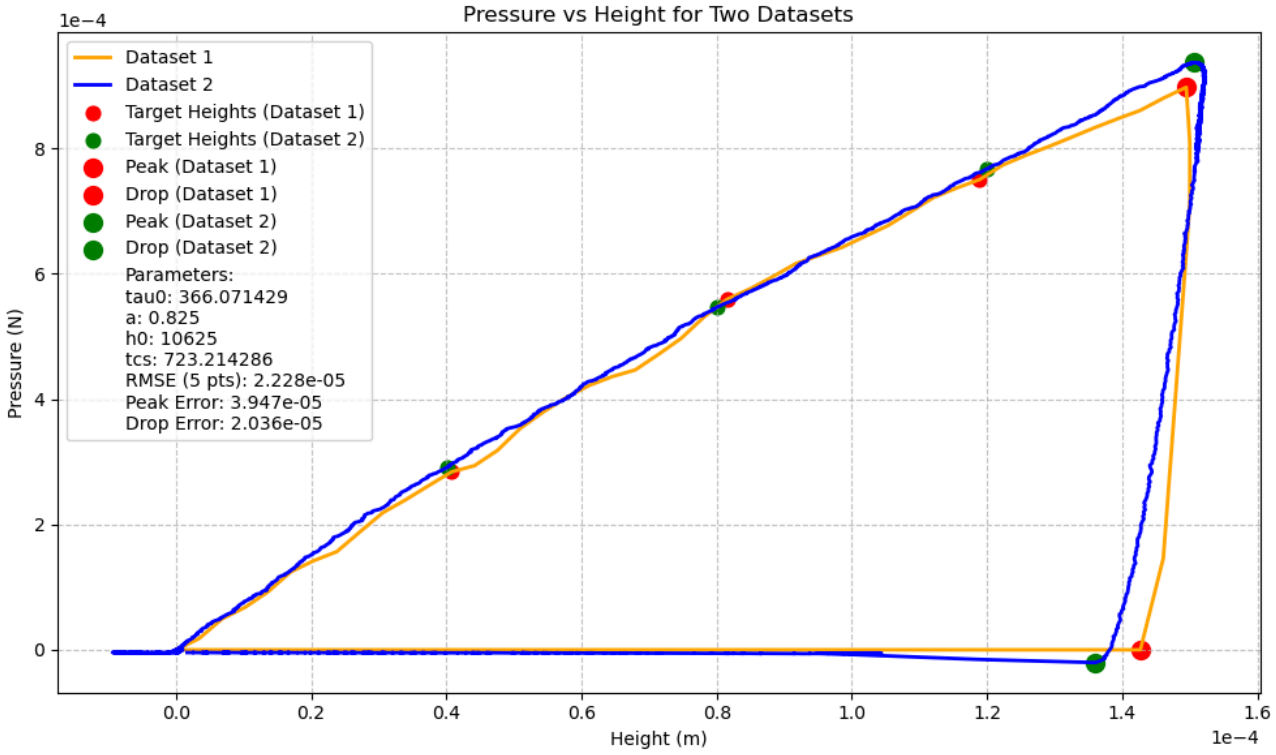


Figure 5.4.9: QP1200 Martensite Comparison

Analyzing the results from all of the materials so far, we get a very comprehensive understanding of their behavior. Also, the fact that our ANN model was able to adapt to all of the materials means that the model is very versatile and accurate.

5.5 Limitations and Future Improvements

5.5.1 Limitations

We got really promising results in terms of getting accurate outputs from our ML algorithm. Therefore, it can be said that this is a good starting point to fully automate the CPFEM parameter calibration process. However, there are several limitations that should be discussed.

1. **Limited Dataset Size**: On average, the datasets we used for each material had 50 datapoints. This is a very small number considering that the materials we are working with has complex structures. This can cause less accurate results since the algorithm becomes more sensitive to any noise or outliers.
2. **Strain Rate Sensitivity**: Our model struggled to get good results in the unloading phase. This may be caused by the fact that we eliminated several features at the start of our study to make our model less complex and without these parameters our ANN model is not able to get a better understandin
3. **Computational Resource Demands**: Training our model was computationally really demanding. This limitation can cause a problem in the future when this model wants to be be improved and more steps are added to the framework.

5.5.2 Future Improvements

In order to eliminate the limitations we talked about in the previous section, we recommend several improvements. With these improvements, we aim to increase the accuracy, scalability, and adaptability of our project.

1. **Expanding Dataset Size**: We can increase the dataset size to improve the accuracy of our ANN model. This would help with the error values of the current materials, and also any additional materials.
2. **Improved Feature Engineering**: We can add additional features that our ML model can learn from. For example, these can be unloading slopes, strain rate dependencies, and more points such as 10th and 90th percentiles.
3. **Optimizing Computational Efficiency**: We can try to optimize our ANN model to lower the computational demand to optimize the process.

7. Conclusion and Outlooks

In our study, we developed an efficient and automated workflow for optimizing nanoindentation related crystal plasticity parameters using machine learning models. We used nanoindentation experiments, CSC simulations, and Artificial Neural Networks (ANNs) to optimize the process and eliminate the challenges that manual parameter calibration process poses. Using the workflow we developed, there is no need to use the traditional, time consuming calibration methods.

Our ANN model showed reat accurac, meaning that it was able to capture complex, nonlinear dependencies in our dataset. We can also see that from the low Root Mean Square Error (RMSE) and high R-squared (R^2) values that we received.

We learned several key points during our project. Firstly, we were able to align theoretical and experimental force-displacement curves closely, significantly improving the predictive reliability of CPFEM simulations. Additionally, the use of ML changed the parameter calibration process significantly.

Despite these successes, limitations such as small dataset sizes and computational demands were noted. Addressing these issues by expanding datasets, incorporating additional features, and optimizing computational efficiency will further enhance the applicability and scalability of the proposed workflow.

The findings have broad implications for industries like automotive, aerospace, and energy, offering pathways to design materials with tailored properties while promoting sustainability and innovation. Future research could explore extending this workflow to other complex material systems and refining the machine learning model for greater generalizability and precision.

8. Personal Evaluation

Youngbin

This project has allowed me to further extend my knowledge in machine learning used in the engineering field. The use of different models has aided my understanding on various applications. It was very interesting to build Abaqus simulation models for nanoindentation and running a supercomputer to run massive amount of simulation using high performance computing. Also, as the group lead, I was able to improve my teamwork and leadership skills. The past 13 weeks of working on the project has given me new knowledge in computer science, data science, materials science and engineering, and teamwork.

Arturo

Overall, the project allowed me to extend much of the knowledge acquired in the material science and the finite elements courses. It was really interesting to combine that prior knowledge with experience from programming and machine learning in an interdisciplinary project. Moreover, I have never had the chance to experience working with such big files and running them in a supercomputer. I believe our group did an excellent job in communicating, scheduling, and organizing tasks every week to obtain the desired results. The project also helped me to develop teamwork and collaborative skills needed for practical and professional tasks.

Irem

I was mainly focused on the machine learning section of our project. I developed an automated parameter calibration process for the Crystal Plasticity Finite Element Method (CPFEM). It was really interesting and exciting to develop an ML algorithm for a real-world applicable project. This experience taught me a lot about ML in material science, and improved my problem-solving and technical skills. In addition, by participating in team discussions every week and working across multiple areas, I developed strong teamwork and communication skills. Overall, this project improved my technical skills, critical thinking, and ability to work in a multidisciplinary team.

Mac

I was mainly focused on the parametric studies and preparing the datasets derived from the study data. This project provided me with an opportunity to improve my understanding of material behavior under nanoindentation and how parametric changes influence the simulation results. Working on data processing and integration with machine learning workflows allowed me to deepen my technical expertise in computational materials science. Additionally, collaborating with my team improved my communication, organization, and problem-solving skills. Overall, this project expanded my knowledge of engineering applications, data-driven workflows, and teamwork in an interdisciplinary environment.

9. References

Liu, M., Lu, C., & Tieu, A. K. (2015). Crystal plasticity finite element method modeling of indentation size effect. International Journal of Solids and Structures, 54, 42-49. https://doi.org/10.1016/j.ijsolstr.2015.03.003

Liu, W., Lian, J., & Münstermann, S. (2020). A strategy for synthetic microstructure generation and crystal plasticity parameter calibration of fine-grain-structured dual-phase steel. International Journal of Plasticity, 126, 102614. https://doi.org/10.1016/j.ijplas.2019.10.002

Moreno, A. R., & Hähner, P. (2018). Indentation size effects of ferritic/martensitic steels: A comparative experimental and modeling study. Materials and Design, 145, 168-180. https://doi.org/10.1016/j.matdes.2018.02.064

Nguyen, X. B., Nguyen, T. K., & Vo, K. A. (2022). Large-scale automated parameter calibration in the crystal plasticity constitutive models during linear and nonlinear loading conditions. Aalto University Computational Engineering Project Report.

Wang, Z., Zhang, J., Hassan, H. U., & Yan, Y. (2018). Coupled effect of crystallographic orientation and indenter geometry on nanoindentation of single crystalline copper. International Journal of Mechanical Sciences, 144, 267-275. https://doi.org/10.1016/j.ijmecsci.2018.09.007

Zhang, M., Li, J., & Tang, B. (2018). Mechanical characterization and strain-rate sensitivity measurement of Ti-7333 alloy based on nanoindentation and crystal plasticity modeling. Progress in Natural Science: Materials International, 28(6), 763-772. https://doi.org/10.1016/j.pnsc.2018.10.003

Rajagopal, M., & Johnson, D. R. (2020). Application of machine learning in calibrating material parameters for finite element models. Materials Science and Engineering A, 789, 139550. https://doi.org/10.1016/j.msea.2020.139550.

Fang, L., Wu, J., & Zhang, W. (2019). A machine learning framework for optimizing crystal plasticity models. Computational Materials Science, 170, 109208. https://doi.org/10.1016/j.commatsci.2019.109208.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. Retrieved from https://www.deeplearningbook.org/

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), 2481-2495. https://doi.org/10.1109/TPAMI.2016.2644615

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003

10. Appendix

Machine Learning Training

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_absolute_error

# Reload the data files from specified sheets
parameters_df = pd.read_excel('parameters_finaleee.xlsx', sheet_name='QP1200_2')
target_df = pd.read_excel('parameters_final.xlsx', sheet_name='theory_QP1200_2')

# Extract feature columns and target columns from the training data
features = parameters_df[['tau0', 'a', 'h0', 'tcs']]
targets = parameters_df[['peak', 'drop point', '25% peak', '25% drop', '50% peak',
'50% drop', '75% peak', '75% drop']]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, targets,
test_size=0.2, random_state=42)

# Scale the features for optimal performance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the multi-output regressor with Gradient Boosting as the base model
model = MultiOutputRegressor(GradientBoostingRegressor())

# Train the model on the scaled training data
model.fit(X_train_scaled, y_train)

# Predict on the test set and calculate the Mean Absolute Error (MAE) for
performance
y_pred = model.predict(X_test_scaled)
test_mae = mean_absolute_error(y_test, y_pred)

# Define the target outcomes based on the "theory" sheet data
target_outcomes = target_df.mean()  # Adjust as needed to align with actual
theoretical target

tolerance = 0.05  # 5% tolerance
```

```
# Function to calculate if the predictions are within the tolerance range
def is_within_tolerance(predictions, targets, tolerance):
    # Check if all predictions are within (1 ± tolerance) range of the target outcomes
    lower_bound = targets * (1 - tolerance)
    upper_bound = targets * (1 + tolerance)
    return ((predictions >= lower_bound) & (predictions <= upper_bound)).all()

# Start with an initial set of parameters (mean of training data as a reasonable starting point)
current_params = X_train.mean().values.reshape(1, -1)
scaled_current_params = scaler.transform(current_params)  # Scale initial parameters

# Iterative optimization loop
max_iterations = 1000
for iteration in range(max_iterations):
    # Predict the outcomes for the current parameters
    predictions = model.predict(scaled_current_params).flatten()

    # Check if predictions are within tolerance
    if is_within_tolerance(predictions, target_outcomes.values, tolerance):
        optimal_params = scaler.inverse_transform(scaled_current_params)[0]  # Rescale to original parameters
        break

    # Adjust each parameter slightly and check if it improves prediction toward target
    improved = False
    for param_index in range(current_params.shape[1]):
        # Create a slight positive and negative adjustment to the current parameter
        for adjustment in [-0.01, 0.01]:  # Small adjustments
            test_params = scaled_current_params.copy()
            test_params[0, param_index] += adjustment
            test_predictions = model.predict(test_params).flatten()

            # Check if this adjustment brings predictions closer to the target
            if is_within_tolerance(test_predictions, target_outcomes.values, tolerance):
                optimal_params = scaler.inverse_transform(test_params)[0]
                improved = True
                break
        if improved:
            break
```

```
    else:
        # No improvement found with current adjustments; stop iterating
        optimal_params = scaler.inverse_transform(scaled_current_params)[0]
        break

# Display the final optimal parameters found
optimal_params
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# --------------------------
# Data Loading and Preparation
# --------------------------
parameters_df = pd.read_excel('parameters_finaleee.xlsx', sheet_name='QP1200_2')
target_df = pd.read_excel('parameters_final.xlsx', sheet_name='theory_QP1200_2')

# Extract feature columns and target columns
features = parameters_df[['tau0', 'a', 'h0', 'tcs']]
targets = parameters_df[['peak', 'drop point', '25% peak', '25% drop', '50% peak',
'50% drop', '75% peak', '75% drop']]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, targets,
test_size=0.2, random_state=42)

# Scale the features for optimal ANN performance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# --------------------------
# Build and Train the ANN Model
# --------------------------
# Define the neural network architecture
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    layers.Dense(64, activation='relu'),
```

```
    layers.Dense(y_train.shape[1])  # Output layer with one neuron per target
])

# Compile the model
model.compile(optimizer='adam', loss='mse')  # Using MSE as a loss function

# Train the model
model.fit(X_train_scaled, y_train.values, epochs=100, batch_size=16, verbose=1)

# Evaluate model performance on the test set
y_pred = model.predict(X_test_scaled)
test_mae = mean_absolute_error(y_test, y_pred)
print(f"Test MAE: {test_mae:.3f}")

# --------------------------
# Parameter Targeting Logic
# --------------------------
# Define the target outcomes based on the "theory" sheet data
# You may need to adjust how you define the target_outcomes.
target_outcomes = target_df.mean()  # Using the mean of theory data as the target

tolerance = 0.05  # 5% tolerance

def is_within_tolerance(predictions, targets, tolerance):
    # Check if all predictions are within (1 ± tolerance) of the target outcomes
    lower_bound = targets * (1 - tolerance)
    upper_bound = targets * (1 + tolerance)
    return ((predictions >= lower_bound) & (predictions <= upper_bound)).all()

# Start with an initial set of parameters (mean of training data as starting point)
current_params = X_train.mean().values.reshape(1, -1)
scaled_current_params = scaler.transform(current_params)  # Scale initial
parameters

# Iterative optimization loop
max_iterations = 1000
optimal_params = None

for iteration in range(max_iterations):
    # Predict outcomes for current parameters
    predictions = model.predict(scaled_current_params).flatten()

    # Check tolerance
    if is_within_tolerance(predictions, target_outcomes.values, tolerance):
        optimal_params = scaler.inverse_transform(scaled_current_params)[0]
```

```
            print(f"Found parameters within tolerance at iteration {iteration}:
{optimal_params}")
        break

    # Attempt to improve parameters
    improved = False
    for param_index in range(current_params.shape[1]):
        # Create slight positive and negative adjustments
        for adjustment in [-0.01, 0.01]:
            test_params = scaled_current_params.copy()
            test_params[0, param_index] += adjustment

            test_predictions = model.predict(test_params).flatten()
                    if is_within_tolerance(test_predictions, target_outcomes.values,
tolerance):
                optimal_params = scaler.inverse_transform(test_params)[0]
                      print(f"Found parameters within tolerance after adjustment at
iteration {iteration}: {optimal_params}")
                improved = True
                break
        if improved:
            break
    else:
        # If no improvement found, break out
        optimal_params = scaler.inverse_transform(scaled_current_params)[0]
        print(f"No improvement found after iteration {iteration}.")
        break

if optimal_params is not None:
    print("Optimal Parameters Found:", optimal_params)
else:
    print("No parameters found within tolerance.")
```

Plotting Curves

PH curve plotting

```
import matplotlib.pyplot as plt
import numpy as np



# Read data from file
```

```
data = np.loadtxt('your/directory', skiprows=4)   # Substitute the directory with
yours

# Extract columns
height = data[:, 1]  # Second column
pressure = data[:, 2]  # Third column

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(height, pressure, color='orange', linewidth=2, markersize=3)

# Set labels and title
plt.xlabel('Height')
plt.ylabel('Pressure')
plt.title('Pressure vs Height')

# Use scientific notation for y-axis
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))

# Add grid
plt.grid(True, linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```

Exp. Plot Comparisons

```
import pandas as pd
import matplotlib.pyplot as plt
import os

directory = 'dir' # QP1000

file_names = ['QPG1LC.txt', 'QPG1-2 LC LC.txt', 'QPG1-3 LC LC.txt', 'QPG1-4 LC
LC.txt', 'QPG1-5 LC LC.txt']

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

# Loop through each file, read the data, and plot it
for file_name in file_names:
   file_path = os.path.join(directory, file_name)
         data = pd.read_csv(file_path, skiprows=6, delim_whitespace=True,
engine='python', encoding='ISO-8859-1')

   # Rename columns to avoid extra spaces
```

```
    data.columns = ["Depth_nm", "Load_uN", "Time_s", "Depth_V", "Load_V"]

    # Extract columns (adjust column index as per your file structure)
    time = data["Time_s"]  # Second column
    load = data["Load_uN"]  # Third column

    # Plot the data
    plt.plot(time, load, linewidth=2, markersize=3, label=file_name)
    plt.xlabel("Time (s)")
    plt.ylabel("Load (uN)")
    plt.title("Load (uN) vs. Time (s)")
    plt.legend()
    plt.grid(True)
```

Training vs ML predicted curve comparison

```
import numpy as np
import matplotlib.pyplot as plt
import os

# Directory containing the .txt files
directory = 'your/dir'

# Get all .txt files in the directory
file_names = [file for file in os.listdir(directory) if file.endswith('.txt')]

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

# First, plot all regular files
for file_name in file_names:
    if file_name != 'FD_curve_ML.txt':
        file_path = os.path.join(directory, file_name)
        data = np.loadtxt(file_path, skiprows=2)  # Adjust skiprows as needed

        # Extract columns
        height = data[:, 1] * 1e6  # Second column
        pressure = data[:, 2] * 1e6  # Third column

        # Plot with lower zorder
            plt.plot(height, pressure, color='orange', linewidth=2, marker=".",
markersize=3, label=file_name, zorder=1)
```

```
# Then, plot the special file with a higher zorder
special_file_path = os.path.join(directory, 'FD_curve_ML.txt')
special_data = np.loadtxt(special_file_path, skiprows=2)
special_height = special_data[:, 1] * 1e6
special_pressure = special_data[:, 2] * 1e6

# Plot special file with higher zorder to ensure it's on top
plt.plot(special_height, special_pressure, color='blue', linewidth=2, marker=".",
markersize=3, label='FD_curve_ML.txt', zorder=10)

# Set labels and title
plt.xlabel('Displacement (µm)')
plt.ylabel('Force (MPa)')
plt.title('Pressure vs Height (tcs) 1nm/s')

# Use scientific notation for y-axis
plt.ticklabel_format(style='sci', axis='y', scilimits=(0, 0))

# Add grid
plt.grid(True, linestyle='--', alpha=0.25)


# Show the plot
plt.show()
```