MACHINE LEARNING ASSISTED CALIBRATION OF DUCTILE FRACTURE LOCUS MODEL

Computational Engineering Project Group 2

Members:

Jahanzaib Naveed

Erald Shahinas

Shreyas Giridar

Supervisor: Li Zinan

AALTO UNIVERSITY

1 Abstract

With the emerging demand for lightweight component design and high fuel efficiency of High strength steels, their fracture behaviour must be studied so that we can accurately are reliably predict their behaviour. While experimental-numerical method can predict the material parameters for damage model accurately. Their calibration process is complex and time-consuming. With the use of machine learning and artificial intelligence to predict the model parameter which describes the ductile fracture, the calibration process can efficiently speed up. This study adopts a three-dimensional damage and fracture loci based on the anisotropic evolving non-associated Hill48 (enHill48) model [33]. To replace the calibration process of damage evolution and fracture, machine learning assisted parameter identification has been proposed using artificial nural network which uses multi layer perceptron model. Four different model have been proposed and using visual and numerical validation methods one final model has been proposed.

2 Introduction

Ductile metals are used everywhere, from airplanes to electronic components in smartphones. Understanding and predicting the behaviors of these metals is crucial in keeping them safe from failure. It is even more important to capture and describe the fracture behaviour of high strength steels as they are increasingly being used to fulfil the requirements of lightweight component design with high fuel efficiency.

Ductile fracture models can be described through coupled and un-coupled models. Although the uncoupled models have a simple structure and have an easy parameter identification, they do not describe the damage to the plasticity of the material. However, within the region of plastic deformation, uncoupled models describe the damage [28]. Contrastingly, coupled models are more sophisticated, as they include the plastic deformation process with the evolution of damage resulting in failure [28]. Some commonly used uncoupled models are Mohr-Coulomb [3], Rice and Tracey [30], and Johnson-Cook criteria [31]. GNT or Gruson-Tvergaard-Needleman [32] is a coupled model that is commonly used. The uncoupled model has easy structure and parameter identification but has low prediction accuracy for high stress and cannot describe the damage evolution. However, coupled models are more reliable and can predict accurately while also describing the damage evolution. Coupled models come at a cost, as they are complex models and due to their complexity, the parameter identification, practical applications are limited [28].

According to [28], [33] proposed semi-coupled damage model, where the fracture strain is modified to consider the evolving damage variable with initial damage rather than just initial values. By only considering initial values, prediction of the fracture behaviour at large stress/strain becomes poor. This inclusion of the instantaneous values helps in describing the damage softening effect which usually occurs at high stress/strain. For Hill48 or enHill48 introduces that fact that not only hardening curves but also R-values are evolving and changing depending on plastic deformation. Hence, the anisotropic parameters change during the plastic deformation and should be taken as a function of equivalent plastic strain [33].

There has been two similar research done. In [28] the research is done on the parameter identification for 6061 aluminium alloy sheets. Due to the trade-off between parameter calibration complexity and the descriptive ability, fracture loci based on the uncoupled fracture criterion of Lou-Huh was embedded with the Bai-Wierzbicki model. [28] also uses machine learning for parameter identification. Experimental along with FEM data was used for the parameter calibration. Artificial neural network was used for parameter identification, where the ANN was trained using NTR10. Inputs were Force-displacement curve and output was the parameter set. Since only one specimen (NTR10) was used as the training set, the overall prediction quality for all the specimen was satisfactory. [34] explores the possibility of describing the fracture model using one specimen. However, this study uses the coupled ductile fracture model, where the three-dimensional fracture locus is a function of stress triaxiality and load angle parameter. [34] uses shear tension specimen for FEM simulation. Like [28], [34] uses ANN for parameter identification however of the parameter is already set to certain value. According to the results, ANN model A, which used a heat map of the fracture on the specimen to get the parameters was most accurate compared to the experimental values.

The aim of this research is to calibrate machine learning to identify the fracture parameter of high strength steels. The is achieved by, first taking using Swift Voce hardening curve to describe the material. Then, the calibration for locus fracture is done using Hill48 model which requires stress triaxiality and load angle parameter [33]. Stress triaxiality and load angle parameter can be obtained through experiments. However, it is complex, time and resource heavy. Hence, the calibration process is done through experiment-numerical hybrid. For each of the samples G0-G4, finite element simulation needs to be done. The training data is generated which is then used in machine learning to adjust the weights of the nodes.

3 Theoretical background

3.1 Brief (historical) overview of the topic

Uncoupled fracture indicators have been used to predict ductile fracture firstly in the 1950s. Studies such as by Freudenthal [4] have proposed a criterion based on the total plastic work. Later studies have proposed different criteria such as those based on equivalent plastic strain by Datsko [5], on the void growth mechanism driven by principal stresses by Cockcroft and Latham [6] and on the geometry of defects by Rice and Tracey [7]. Ground-breaking research was conducted by McClintock [8] and Rice and Tracey [7] on the question of how flaws in geometry affects ductile damage. It has been demonstrated through experimentation that the nucleation and expansion of voids associated with significant plastic flows results in the loss of elasticity and the introduction of a softening effect driven by stress triaxiality [8] [7] [9]. According to Bridgman [10], McClintock [8], Rice and Tracey [7], and others, the equivalent plastic strain to fracture and the stress triaxiality were two characteristics of the material ductility. [11]

The benefit of coupled modelling is the idea that they link mechanical attributes and damage progression. Micromechanical model formulations in studies by Rice and Tracey [7], Gurson [12], Tvergaard and Needleman [13], Nahshon and Hutchinson [14], Tutyshkin [15], Malcher and others [16], Brünig and others [17], Reddi and others [18], all include a softening process. They reflect the ductile fracture process physically and are based on the formation of micro voids. Another approach at a coupled method is based on the theory of damage mechanics, in which the material law has a damage parameter D as shown in studies by Lemaitre [19], Lemaitre and Chaboche [20].

3.2 Current publications

According to the study by Baltic and others [11], even though ductile fracture and damage evolution are mature fields of research, many details need to be tackled to

have reliable prediction of complex geometries and loading scenarios. An important challenge in the field is the assembly of a unified numerical method that can predict the behaviour of different materials under ductile fracture. The study [11] states that the method would benefit many real-world applications of metal alloys ranging from lightweight structures to small electronic parts. Previous studies such as those done by Torki and others [21], Scales and others [22] have shown that predicting material behaviour under real loading condition is extremely difficult as the loading path changes considerably from the experiment under laboratory conditions
A study by Yao and others [23] showed that using a semi-coupled fracture model is superior to only using a coupled or uncoupled model. The uncoupled model does not include damage on plastic behaviour, which is an essential part of the coupled model, resulting in a flawed model where the plastic deformation follows unchanged classical theory. However, the uncoupled model has simpler structure and easier parameter identification reducing the difficulty of finding parameters and computation needed. Multiple studies done by Pütz and others [24], Shen and others [25] have demonstrated how a semi-coupled model could work by combining the uncoupled model with damage framework. In research conducted by Liam and others [26] the damage doesn't affect the model until the point of damage initiation, before which the model behaves as simply uncoupled. Although better than the uncoupled method, the semi-coupled one still has difficulty calibrating parameters.

The study by Yao and others [23] displayed two different methods to calibrate the parameters of the semi-coupled model. A theoretical-experimental one which used FEM simulations and one based on machine learning. According to the study [23] it is possible to use machine learning to implement parameter identification, which replaces the calibration process of the damage evolution and fracture models by using a neural network model.

In the research by Yao and others [23] the predicted results of both methods were close to the experimental results which highlights the usefulness of using both. However, the machine learning model used less resources, as the CPU only used 189.2 hours to train the machine learning model. On the other hand, it the CPU 296.7 hours to finish parameter calibration of the parameters.

“The results strongly support the fact that the semi-coupled fracture model can successfully predict damage initiation, accumulation, and fracture of the 6061 alloy during forming. “ [23]

Another recent publication that talks about using machine learning in the calibration of a ductile fracture locus model is the paper written by Sandra B. [1] The paper focuses calibration methodology using a single geometry specimen. It combines Finite Element Modelling and Artificial Neural Network (ANN) to accomplish this task. Like our project, the combinations of the model parameters are used to generate the training database for our model. From the numerical experiment, the local displacement fields and global-force displacement histories are extracted and fed into the ANN model so that the influence of the local stress on the evolution of the local deformation is also considered.

There were many reasons why a machine learning model was used for this project. In the industrial level, imposing constant proportional loadings up to failure over the complete stress state domain is extremely difficult as knowledge of the ideal case of the constant strain-history path is not known in most cases [1]. Aside from that there are many computational challenges related in the industry as well related to spatial time and discretization error as highlighted by [2]. Machine Learning calibration is used to overcome these challenges.

The training data is passed from FE analysis of the specimen to the ANN model. The input consists of images of the complete local displacement of the field and the values of the global force-displacement curve. Both have been extracted from the complete numerical experiment. As the scale of the input and the output are very important, they have been normalized to the range of [0,1]
Thus, this study provides evidence to the support the notion that a single geometry is sufficient to gain knowledge about the hardening of the material provided that the sample experience a variety of different stress states in different positions under plastic deformation. It provides a foundation for automation of material parameter

determination using evolution of local and global features. The article provides evidence for two crucial conclusions:

- It is indeed possible to to determine the ductile locaus fracture of a material without prior knowlede of the loading path. This also means that the critical element of the dor the numerical model does not need to be ambigously chosen.
- It is indeed possible to estimate the realistic material paramters of an object regardless of whichever ANN is chosen provided that some preprocessing of the data is done based on prior knowledge of the material and the physics informed partinoniong of the ANN is done rather than just pure brute force of the application. [1]

Earlier articles have also talked about the use of Artificial Neural Networks for the use of parameter identification of the Gurson–Tvergaard–Needleman (GTN) model. A particular paper by Fethi A. et al [3] discusses the use of ANN for parameter identification in the sheet metal forming process. This paper highlights the need for an accurate estimation of the material parameters in order develop a virtual model for the variety of engineering problems involved in the forming process (like necking, macroscopic cracks etc) which, at the industrial level, can drastically affect the cost of the final product.

To set up the experiment the researchers make use of an application called DIC (Digital Image Correlation) which is based on the comparison of two images of the same model acquired during various stages of deformation. An algorithm is used to calculate the deformation based on the displacement of the pixels. These images are processed with the Aramis software developed by GOM. Aside from that several low-rate tensile tests are carried out by an INSTRON tensile test machine. The output from these results is used to train the neural network.

The experiment uses multilayer neural networks (MLNN) trained with back propagation supervised algorithm for its ANN model. The loss function in the back propagation is the TMSE (Training Mean square error) The back propagation works in nine steps summarized as follows:

1. Choosing the training rate and moment coefficient
2. Randomly initializing the weights
3. Choosing the input sample and propagation of the calculation through the network
4. Calculation of the outputs for all the neurons leaving the input layer to the output layer
5. Measure TMSE by difference between actual output and desired output
6. Stop the algorithm if TMSE is below threshold.
7. Calculate the contribution of one neuron to the error starting from the output and determination of the weight modification sign.
8. Correcting the neurons weights to decrease the error.
9. Repeat step 3.

To evaluate the results of the GTB identification, the researchers developed several experimental bulges tests and compared it with the values obtained by the GTN model. They found a good correlation between the numerical data and the experiment results. They also employed what is called an “Erichsen test” where they compared the deformation observed a rupture in a hemispherical punch of steel with the deformation observed in the experimental data. A rupture occurred in the same area in both the experimental and numerical data, further giving support for the model. Thus, the paper [3] was able to give sufficient evidence of the fact that machine learning models can be used to give accurate results for the identification of damage parameters and a neural network that is trained well can be a useful tool for predicting material deformation.

3.3 Swift Voce

Swift Voce is a hardening law which captures the relationship between stress and strain of undamaged material. This can then be used to fit a curve which closely follows the stress-strain of undamaged material. The fitted curve is extrapolated to larger strain value. The equation 1 is used for the fitting of swift voce

$$a \times (b \times (c + x)^{d}) + (1 - a) \times (e + (f - e) \times e^{-g \times x} \qquad (1)$$

In equation 1, a-g are parameters that need to be optimized to have a perfect fit and x is the strain value. The parameter a or alpha act as a weight for the whole function hence it must be bounded between [0,1]. Since the rest of the parameters have physical meaning it is also important for them to be bounded between [0,∞]. However, very high values even if resulting in a perfect fit cannot be considered optimized.

4 Experimental and numerical methods

4.1 The experimental data

The first aspect of the project was obtaining the relevant data for the specimen that we were going to be working on. In this experiment, we worked on five different specimens which were NDBR25(G0), NDBR10(G1), NDBR6(G2), CHD6(G3), and SH(G4). Through experiments, we were able to collect the force displacement values for each of these specimens for the QP1000 material, we were also able to collect the Uniaxial Tension Flow curve which would be helpful in modelling these specimens in Abaqus. Once the data was collected, we moved on to the actual modelling of these specimens.

4.2 Explaining the specimen

As explained, we took the experimental data for our specimens (G0-G4) and used them for our modelling. For our modelling, we took reference from the 2D sketches that were provided. The following are the sketches that were used to model G0-G4.

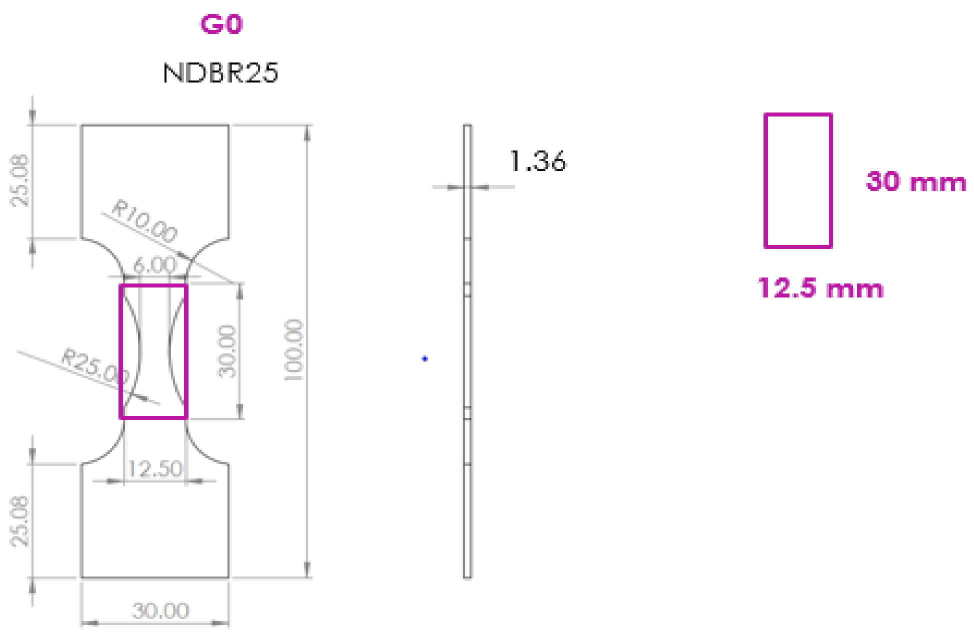


Figure 1, a: G0 Model Sketch

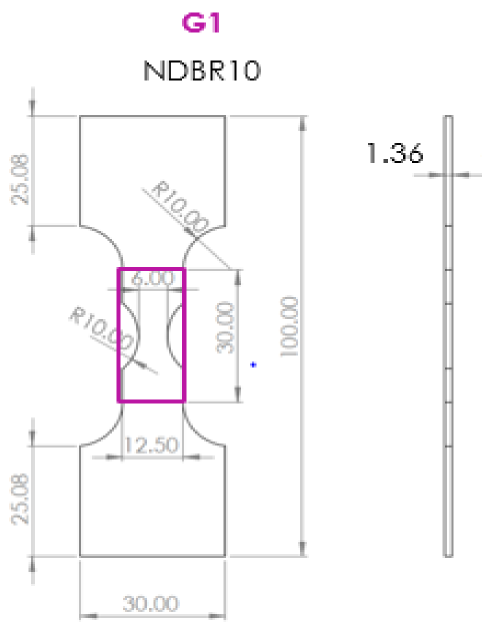


Figure 1, b: G1 Model Sketch

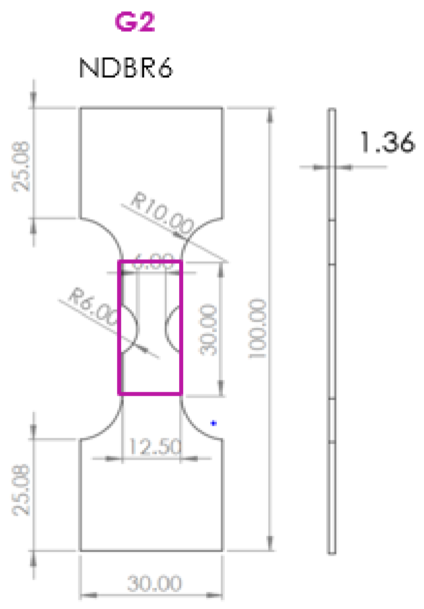


Figure 1, c: G2 Model Sketch

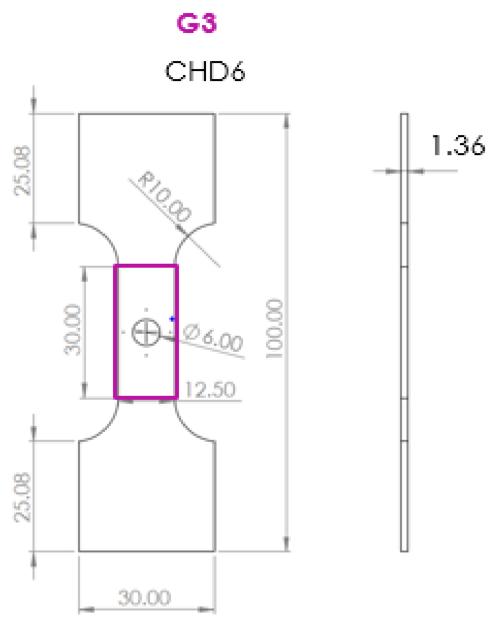


Figure 1, d: G3 Model Sketch

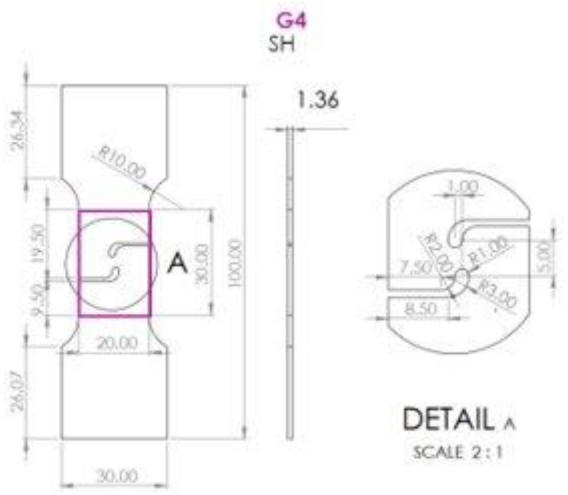


Figure 1, e: G4 Model Sketch

Instead of modelling the entire specimen, we will only be concerned with the areas highlighted in purple as that is the area where most of the deformation occurs and it is the area which experiences the most stress. This will cut down the amount of computation required in our model. We will also be taking half the thickness as specified in these sketches to reduce computational time. G0-G3 have similar designs when compared to G4 which could be considered a complex geometry. This is important as it will add more knowledge to us when we are comparing the results of the different geometries. Finally, using these sketches as our base, we can begin modelling these specimens.

4.3 FEM simulation

For all the specimens, a 3D model was made in Abaqus. First, a basic 2D sketch of the model was made and it protruded to the given height. After that, a material property was created based on the properties we had obtained from our experimental data. The material was then assigned to our model. An Assembly instance was created for our part with Instance type independent and was assigned to our part. We then created steps in our model followed by setting the amplitude. We then choose the field output for our jobs where we specified the values, we were interested in. Boundary conditions were made for each of the models where the Z-axis symmetry was chosen. We also choose the side of our model where we would be observing displacement. We also divided our specimen into different sections which would be important for mesh instances. Different datums based on those sections were made for our models.

When we were creating the mesh, we used a structured technique for the finest and coarsest sections (the centre ones) while sweep technique was used for the transfer regions. An appropriate element size was chosen, and the mesh instance was created for the object. Afterwards, a job was created in Abaqus for our chosen model. A data check was performed on the model followed by the job submission itself. If no errors were detected during this process, the job was created for our model and the file was saved. An example of the Abaqus model can be seen for G0 in figure 06 below.

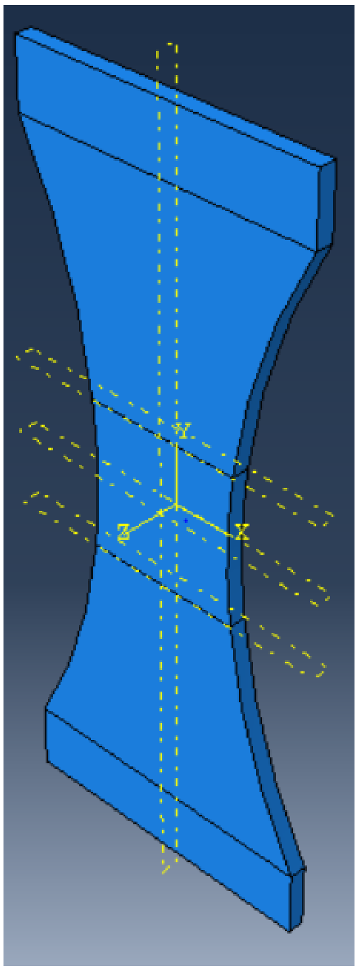


Figure 2: G0 Model

After the job was created, the data could be visualized using the ODB display options available from Abaqus. One could see where the stress was concentrated the most and how the model was deformed. Using those results from the ODB file, we can collect data about the force that was experienced by all the nodes and the average displacement experienced by those nodes as well. These could then use this information for us .rpt file which would give the step-time, displacement and force our specified model. The entire process is done for all 5 of our models.

4.4 Job submission automation

After finishing work on the models, the process of job submission was done locally in order to examine the quality of the created models. However, this proved inefficient early in the project as the Abaqus software requires a considerable amount of computation power. Thus, it was decided to move the process of job submission to a cloud platform with reasonable computational resources and that could run jobs in the background. The cloud platform used was CSC (IT Center for Science Ltd.) and the Puhti supercomputer.

To facilitate the process of submitting different jobs for each model and each parameter used, it was decided to use batch files to automate the process and eliminate cumbersome manual labour. First, a common csc project was created for all team members. Afterwards it was necessary to also be part of the 2001978 abaqus2 project in order to gain access to the Abaqus module inside of CSC. The CSC Puhti documentation provided enough guidance in order to remotely connect to the server and submit batch files.

The final Slurm batch file can simultaneously run 15 jobs (5 models and 3 parameters for each) which greatly facilitates the job submission. (See Appendix i)

4.5 Swift Voce

One of the key parameters needed for the FEM simulation is the yield stress and strain values (Plasticity values). To get these data Swift Voce hardening law was used. Experimental data of stress and strain was used as to fit the curve and get the parameters of the Swift Voce. To ignore the softening phenomena, the experimental data was only considered until peak stress.

Initially Matlab and its fitting toolbox were used. However, the resulting curve fitting was sub-optimal. Hence python with scipy library was used, as it easily allows to define a piecewise function which improved the fitting quality.

The input data was extracted from a csv file as X and Y. They were then trimmed so that the data only has datapoints until peak stress. Then a Swift Voce function was defined. This function returned a piecewise Swift Voce, where the piecewise was defined to be X<0.02. Curve fit function from scipy library was used to get the fitted curve's parameters. Then a strain from 0 until 3 was created with the precision 0.001, which was then used to get the true stress. Figure 7 3 shows the result. The Blue line corresponds to the experimental data and the black line describes the plasticity using the parameters of the fitted curve.

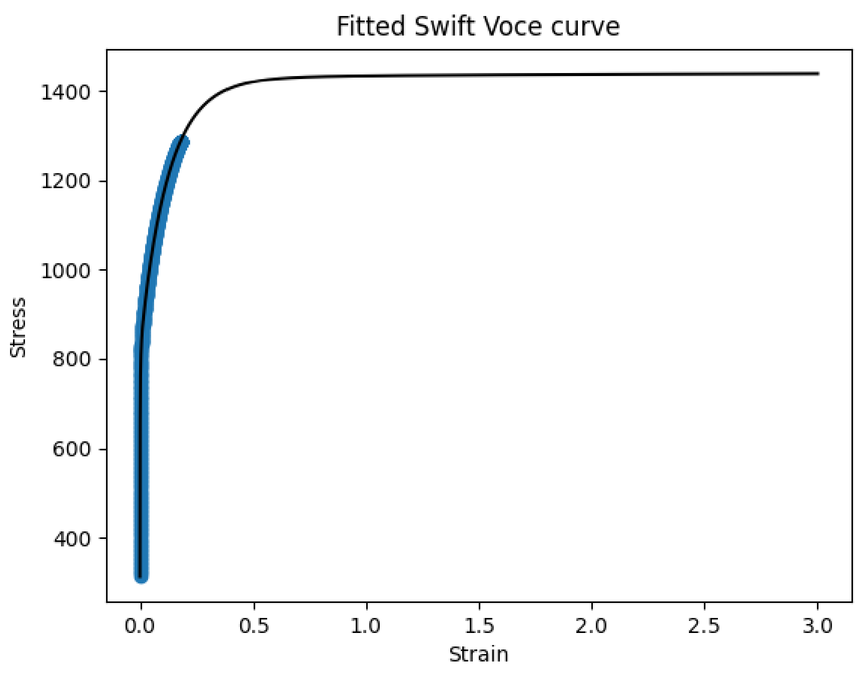


Figure 03: Fitted Swift Voce Curve

Initially only one swift voce curve in the rolling direction was used for damage initiation simulation. However due to the change in abaqus version, the previous subroutine did not work. Hence, a new subroutine was used which included seven swift voce curves in different directions. This used enHill 48 model which required stress-strain relationship curves and r-value curves in different directions. Four of them were for true stress and strain in rolling direction, diagonal direction, transverse direction and lastly biaxial direction. The other 3 were for r-value curves in three directions RD, DD, TD.

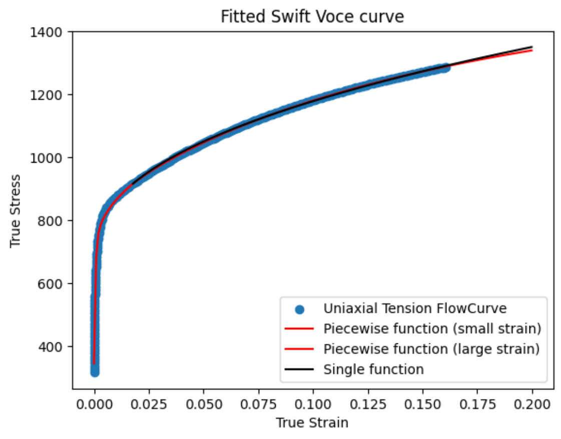


Figure 4, a: RD true stress-strain curve

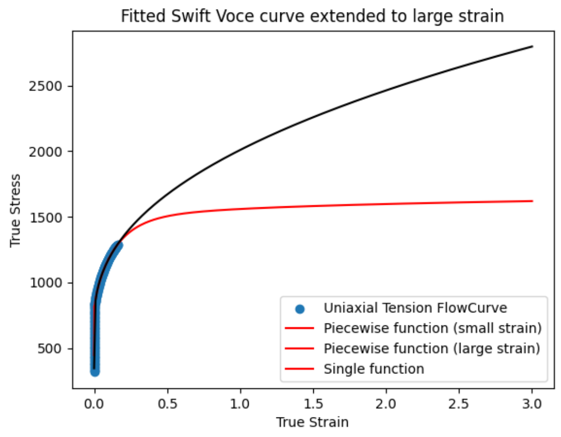


Figure 4, b: RD true stress-strain curve extended to large strain

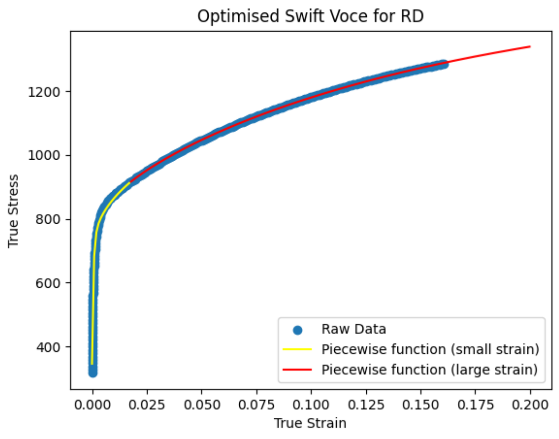


Figure 4, c: Optimised RD fitted curve

Figures 4 represents the fitted Swift Voce curve for true stress strain curve for RD. Two piece wise function was used to get an accurate approximation of the relationship between stress and strain for extrapolation. The first pricewise was from the zero strain up to 0.024. In the Figure 8 the first piecewise is the same for similar to the single swift voce curve. From 0.024 up to the end another piecewise was used. This helps in accurately capturing the relationship and is reflected in Figure 4 b, where there is a huge difference when extrapolating the using a single function and using piecewise to describe the true stress-strain. The final optimised swift voce curve for RD can be seen in Figure 4 c.

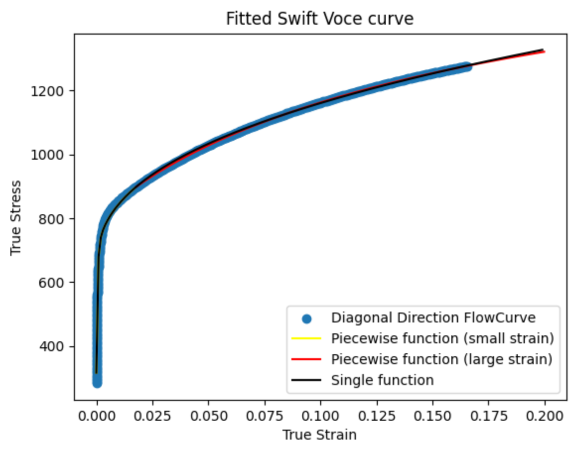


Figure 5, a: DD true stress-strain curve

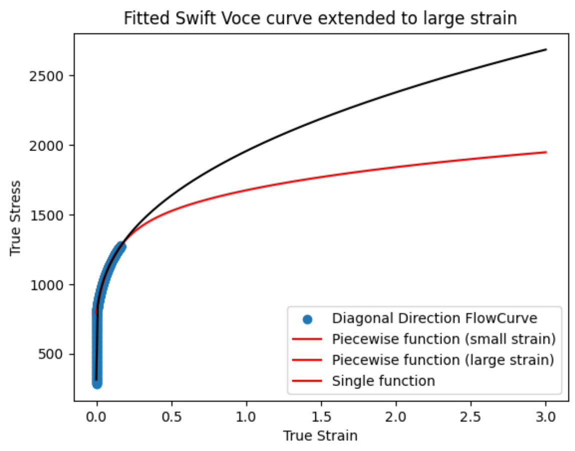


Figure 5, b: DD true stress-strain curve extended to large strain

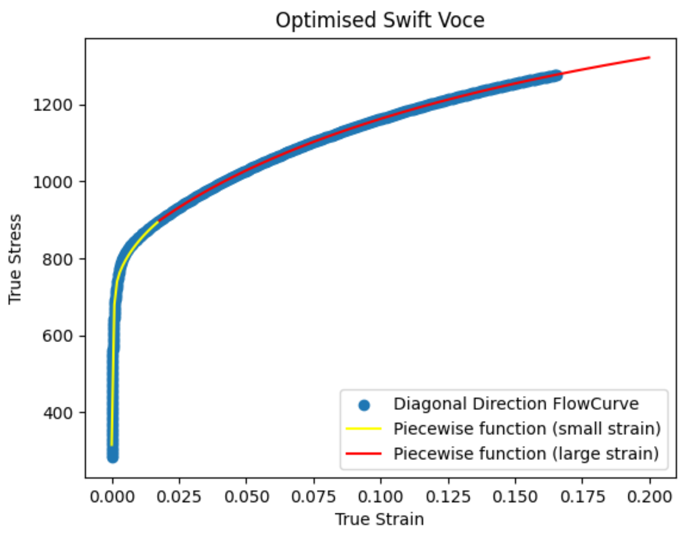


Figure 5, c: Optimised DD fitted curve

Similarly, to the Swift Voce for RD, DD was also fitted and optimized. In figure 5 a it can be observed that the black curve which represented by singular swift voce function looks to have a good fit on the stress strain curve of the material. However when extrapolated to incorporate high strain values shown in figure 5 b, the black curve has high overestimation compared to the pricewise function. Hence the optimized swift voce curve for DD was the piece wise function. Where the first piecewise is from 0 to 0.01 and the other from 0.01 to 3. This can be seen in figure 5 c.

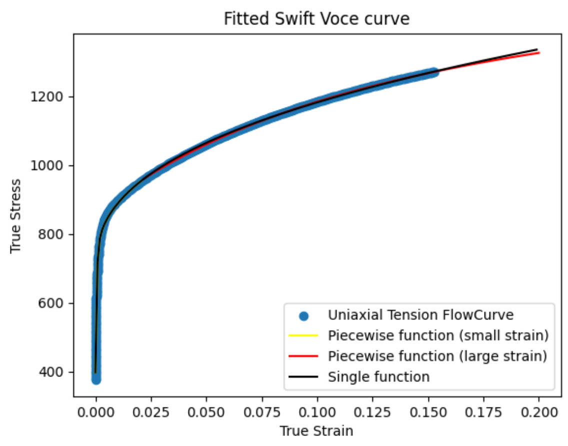


Figure 6, a: TD true stress-strain curve

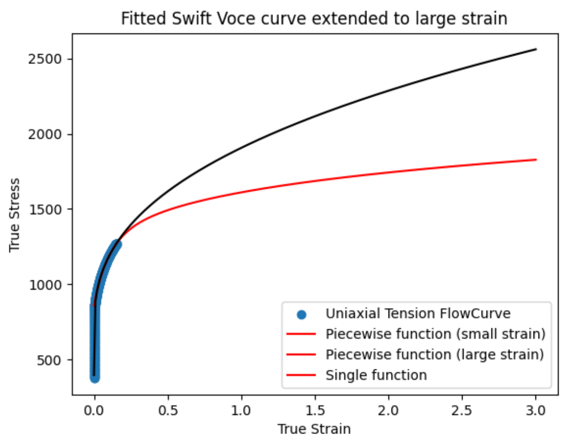


Figure 6, b : TD true stress-strain curve extended to large strain

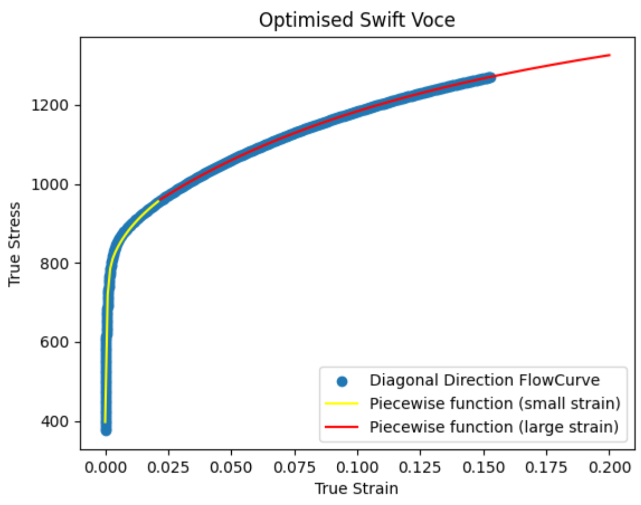


Figure 6, c: Optimised TD fitted curve

Figures 6 describe the true stress-strain relationship of TD. Similar to RD and DD, it can be observed that single swift voce curve has a drastic overestimation compared to a piecewise function. The first piecewise was from 0 to 0.08 and the other was from 0.08 to 3.

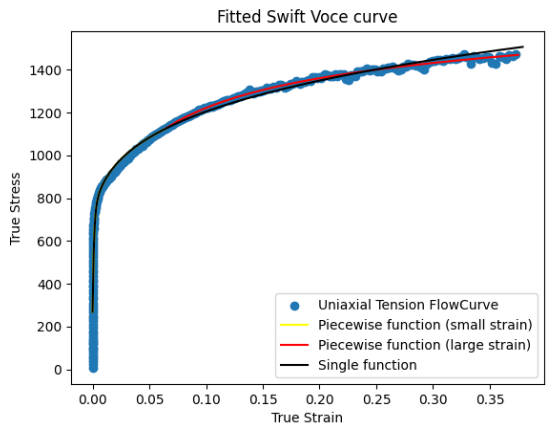


Figure 7, a: Biaxial true stress-strain curve

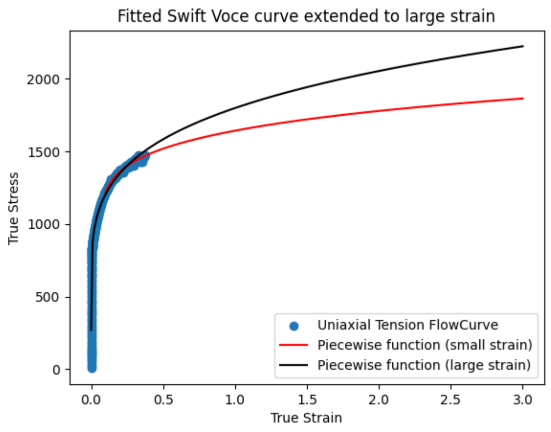


Figure 7, b : Biaxial true stress-strain curve extended to large strain

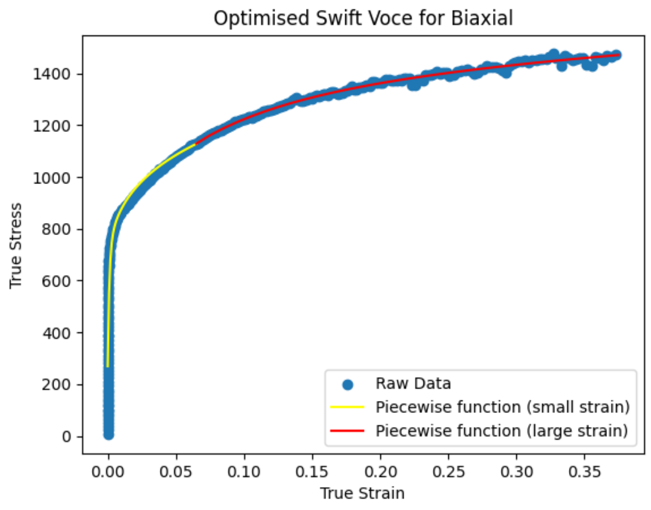


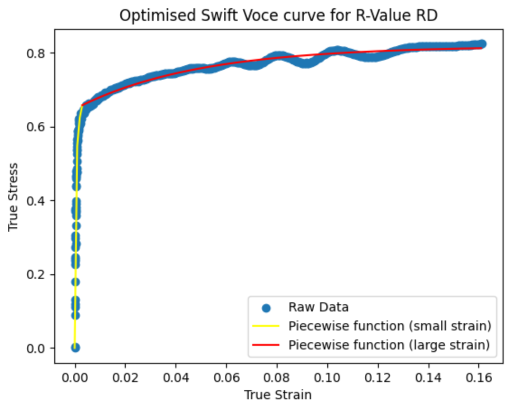
Figure 7, c: Optimized Swift Voce curve for Biaxial Direction

Similarly the Biaxial direction was also done with a piecewise, yielding a better estimation compared to a singular function. The optimised fit can be found in figure 7 c.

Unlike true stress-strain curves for RD, DD, TD and Biaxial, the Swift Voce model for the r-value curves is a little different. Instead of using seven parameters in the case of RD, DD, TD and Biaxial, the r-value model only uses three parameters to describe the curve. The Voce exponential law is defined as,

$$r(\varepsilon - P) = C_1 + C_2(1 - e^{(-C_3\varepsilon^{-P})}) \quad (2)$$

Where C_1, C_2, C_3 are the unknown parameters.



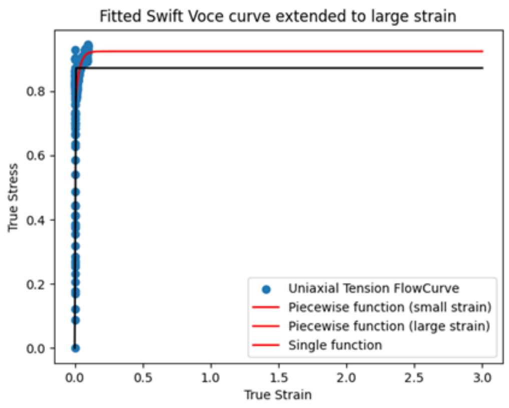
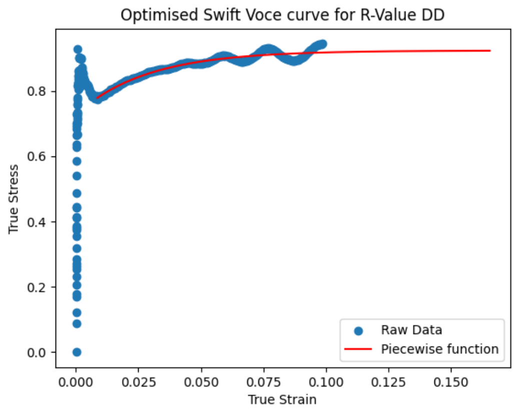


Figure 8, a: R-value RD true stress-strain curve

Figure 8, b: R-value RD true stress-strain curve

Graphs 8 shows Voce-exponential fitted curve for R-value RD. Taking a look at graph 8, it is clear that R-value RD has a more similar stress-strain scatter plot to RD, TD, DD and Biaxial. However, there is still some wave pattern at larger strain values. Due to this it was decided to have a piecewise split at 0.003 strain value. At this point the scatter plot shifts from being straight line up to resemble more of logarithmic growth. Hence, both parts of the Voce-hardening piecewise hardening was used, unlike in the case of R-value DD and R-value TD. Figure 8 a shows the optimised curve for R-value RD



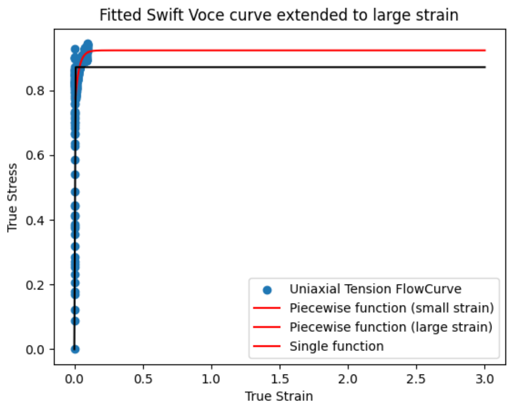


Figure 9, a: R-value DD true stress-curve

Figure 9, b: R-value DD true stress-strain strain curve

Unlike, RD, DD, TD and Biaxial stress strain curves, R-value DD data has a bit more difficult scatter plot. Due to this it was decided to skip the first part/ smaller strain values and only try to predict the latter part of the scatter plot. This was done as it was necessary to extrapolate the data to strain value of 3. Hence, by only capturing the latter part of the curve it is still possible to get better fit. In figure 9 a it can be observed that the single function is plotted from the origin, but the second part of the piecewise function is plotted from about 0.01 strain value. It can be observed from figure 9b that but having a piecewise function and only taking the latter part can greatly help in accurate fitting. Similarly it can be observed in figure 9 b that the piecewise function has a better fit compared to the single function which tends to under estimated unlike in the case of RD, DD, TD and Biaxial, where it over estimated.

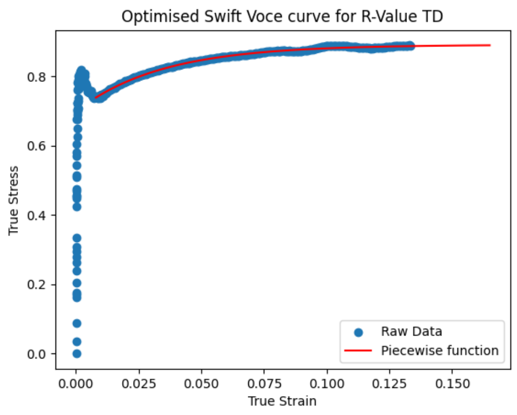


Figure 10, a: R-value TD true stress-strain curve

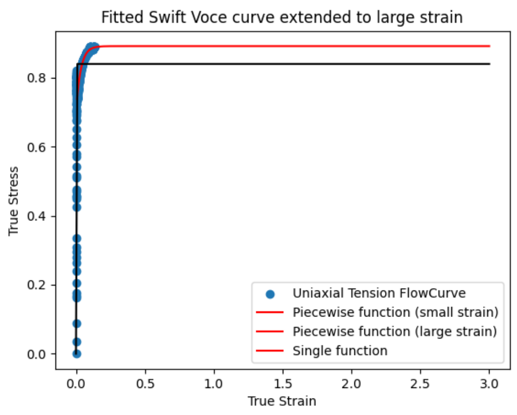


Figure 10, b: R-value TD true stress-strain curve

Similar to R-value DD, R-value TD scatter plot also had difficult to model small strain value. Hence the same strategy was used, which yielded in pretty good result. It can be observed from figure 14 b that only the second part of the piecewise function was used to model this. Which would be the same as using single function from the starting point of the second piecewise. From figure 14 b it is evident that the piecewise function has a better fit and extrapolation compared to single function.

Table 1: List of parameters for Swift-Voce for RD, DD, TD and Biaxial

	A	B	C	D	E	F	G
RD 1	0.529043	2526.578	8.51E-29	0.422	1427.705	734.125	1658.924
RD 2	0.221966	4747.313	2.86E-35	0.049973	651.0779	0.00222	6.444745
DD 1	0.565588	2335.995	3.16E-28	0.399872	1460.681	727.5845	2131.651
DD 2	0.824877	1708.088	0.024342	0.162778	1488.583	136.4697	8.437884
TD 1	0.243672	5004.793	6.51E-27	0.391044	907.3608	525.3794	1780.99
TD 2	0.543445	2439.827	0.04517	0.140625	604.9706	0.000387	9.825161
Biaxial 1	0.495692	2800.106	0.001198	0.243598	814.9343	8.18E-10	868.4386
Biaxial 2	0.894388	3.039128	2.844345	5.644658	2.704797	4.616346	10.27877

Table 1 tabulates the parameters used for the fitted Swift-Voce curve for RD, TD, DD And Biaxial. Where 1 next to RD/DD/TD/Biaxial represents the first pricewise and 2 represent the second piecewise. As expected the alpha of A parameter acts as a weight hence it's value should be between [0,1], which is reflective in table 1. Since all the other parameter have some physical meaning regrading the stress state and strain, they cannot be have negative values. The fit optimization has been done such that these physical meanings have been preserved. A bound from [0,1] for the A parameter and [0,∞] was used during the curve fit.

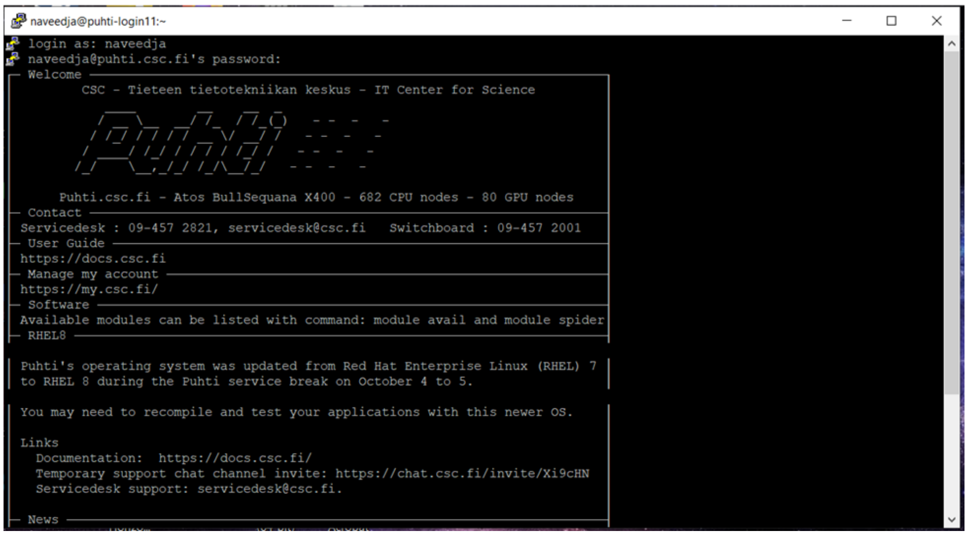
Table 2: List of parameters used in Voce-exponential for R-values RD, DD and TD

R-Value	C_1	C_2	C_3
RD 1	9.38E-16	0.636391	1968.875
RD 2	0.647333	0.17082	20.8649
DD	0.727647	0.196118	35.17942
TD	0.698039	0.192619	29.35943

Voce parameters for fitted R-values have been tabulated in table 2. Similar to Swift Voce, all the unknow parameters have physical meaning hence they cannot take negative values. Similar condition was used when fitting the curve on the scatter plot of R-value TD, DD and RD. Form the Voce equation for r-values it can be observed that the parameter C2 acts as a weight similar to A parameter in Swift-Voce. This can also be seen in table 2. Where C2 parameters only have values between [0,1], which is indicative of weight values.

4.6 Utilization of CSC Server

Once the input files for all the specimens were created, it was time to run FEM simulations for these input files from which valuable data could be collected. These FEM simulations would be incredibly detailed and complex and so we used the assistance of the supercomputers owned by CSC [35]. For our purposes, we used the Puhti supercomputer by CSC for Abaqus simulation. We used WinSCP for our file management system within the CSC server and we used Putty [36] as the control terminal through which we would communicate with the Puhti server and send our commands. (See Figure 11)

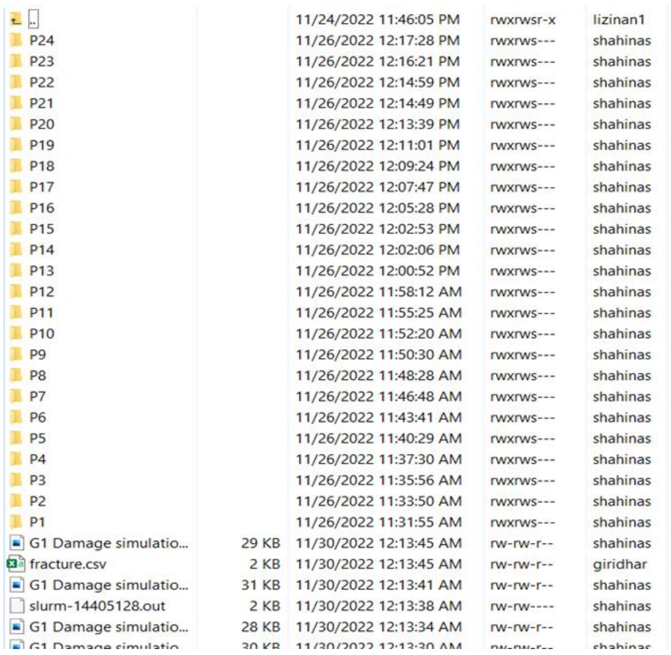


```
naveedja@puhti-login11:~
login as: naveedja
naveedja@puhti.csc.fi's password:
┌─ Welcome ──────────────────────────────────────────────────────────┐
│        CSC - Tieteen tietotekniikan keskus - IT Center for Science  │
│                                                                     │
│     Puhti.csc.fi - Atos BullSequana X400 - 682 CPU nodes - 80 GPU nodes
├─ Contact ──────────────────────────────────────────────────────────┤
│ Servicedesk : 09-457 2821, servicedesk@csc.fi   Switchboard : 09-457 2001
├─ User Guide ───────────────────────────────────────────────────────┤
│ https://docs.csc.fi
├─ Manage my account ────────────────────────────────────────────────┤
│ https://my.csc.fi/
├─ Software ─────────────────────────────────────────────────────────┤
│ Available modules can be listed with command: module avail and module spider
├─ RHEL8 ────────────────────────────────────────────────────────────┤

│ Puhti's operating system was updated from Red Hat Enterprise Linux (RHEL) 7 │
│ to RHEL 8 during the Puhti service break on October 4 to 5.                 │

│ You may need to recompile and test your applications with this newer OS.   │
│
│ Links
│   Documentation:  https://docs.csc.fi/
│   Temporary support chat channel invite: https://chat.csc.fi/invite/Xi9cHN
│   Servicedesk support: servicedesk@csc.fi.
│
├─ News ─────────────────────────────────────────────────────────────┤
```

Figure 11: Putty Terminal

Using the batch files, we had created, we would send the input file for each specimen to the CSC server along with the selected damage model (enHill48 in our case) and the selected damage parameters as well. We also specified the maximum amount of time a single simulation(a job) should take in order to not time out simulations which cannot be completed due to any reason. The command would be sent from the Putty terminal and the status of the jobs could be monitored from there as well. For our purposes, we selected jobs based on the same geometry. Different Jobs with different damage parameters but the same geometry was sent in an array.

First, Jobs without damage parameters were sent in order to get the plasticity models, after which models with damage parameters were sent. All these Jobs were saved in different folders based on their parameter set. For each job, we received the .ODB file, the .prt file, the .rpy file along with other files as well.(See figure)



```
..                             11/24/2022 11:46:05 PM   rwxrwsr-x   lizinan1
P24                            11/26/2022 12:17:28 PM   rwxrws---   shahinas
P23                            11/26/2022 12:16:21 PM   rwxrws---   shahinas
P22                            11/26/2022 12:14:59 PM   rwxrws---   shahinas
P21                            11/26/2022 12:14:49 PM   rwxrws---   shahinas
P20                            11/26/2022 12:13:39 PM   rwxrws---   shahinas
P19                            11/26/2022 12:11:01 PM   rwxrws---   shahinas
P18                            11/26/2022 12:09:24 PM   rwxrws---   shahinas
P17                            11/26/2022 12:07:47 PM   rwxrws---   shahinas
P16                            11/26/2022 12:05:28 PM   rwxrws---   shahinas
P15                            11/26/2022 12:02:53 PM   rwxrws---   shahinas
P14                            11/26/2022 12:02:06 PM   rwxrws---   shahinas
P13                            11/26/2022 12:00:52 PM   rwxrws---   shahinas
P12                            11/26/2022 11:58:12 AM   rwxrws---   shahinas
P11                            11/26/2022 11:55:25 AM   rwxrws---   shahinas
P10                            11/26/2022 11:52:20 AM   rwxrws---   shahinas
P9                             11/26/2022 11:50:30 AM   rwxrws---   shahinas
P8                             11/26/2022 11:48:28 AM   rwxrws---   shahinas
P7                             11/26/2022 11:46:48 AM   rwxrws---   shahinas
P6                             11/26/2022 11:43:41 AM   rwxrws---   shahinas
P5                             11/26/2022 11:40:29 AM   rwxrws---   shahinas
P4                             11/26/2022 11:37:30 AM   rwxrws---   shahinas
P3                             11/26/2022 11:35:56 AM   rwxrws---   shahinas
P2                             11/26/2022 11:33:50 AM   rwxrws---   shahinas
P1                             11/26/2022 11:31:55 AM   rwxrws---   shahinas
G1 Damage simulatio...   29 KB  11/30/2022 12:13:45 AM   rw-rw-r--   shahinas
fracture.csv              2 KB  11/30/2022 12:13:45 AM   rw-rw-r--   giridhar
G1 Damage simulatio...   31 KB  11/30/2022 12:13:41 AM   rw-rw-r--   shahinas
slurm-14405128.out        2 KB  11/30/2022 12:13:38 AM   rw-rw----   shahinas
G1 Damage simulatio...   28 KB  11/30/2022 12:13:34 AM   rw-rw-r--   shahinas
```

Figure 12: File Structure for one of our geometries, G3

From the .ODB files, we were able to extract the information we needed based on the data extraction scripts we had prepared for each model.

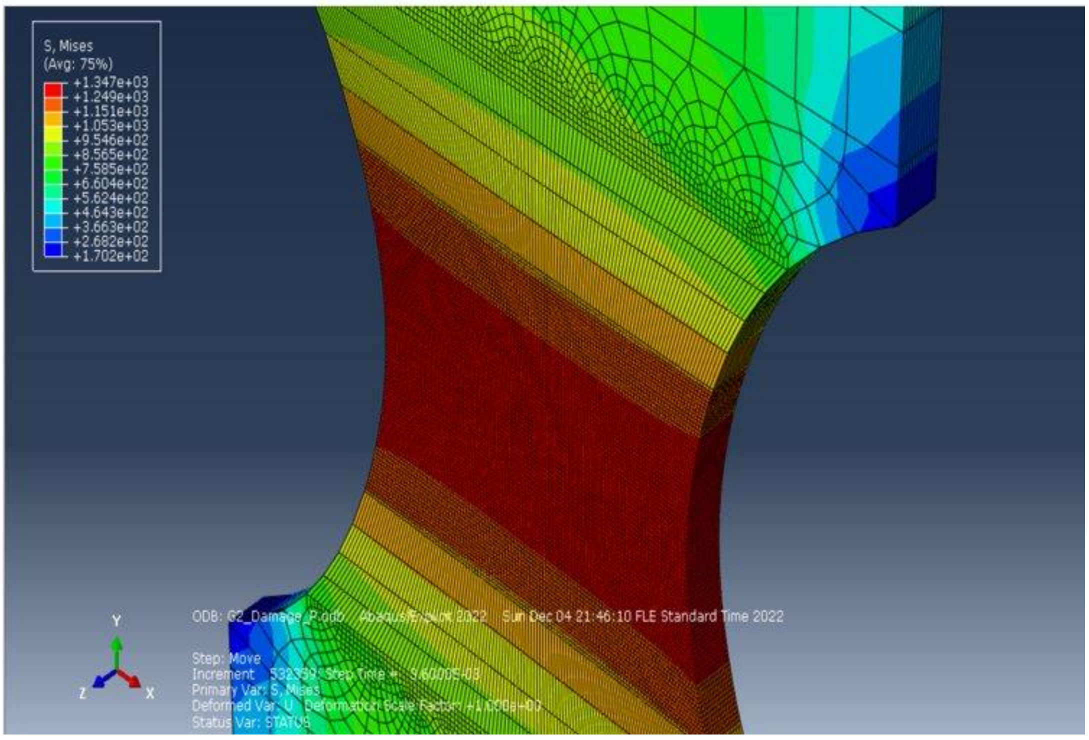


Figure 13: Close up on the tear for Specimen G2

4.7 Scripts for Data Extraction

In order to get the Local and Global data from each of the parameter set, we would have to open each of the .ODB file and extract the needed data from each of them. As there are many parameters sets for each of the multiple specimens, doing this would have taken a long time and would have been a cumbersome process. Hence, we developed python scripts which could do that work for us for all of the parameter sets.

From the plasticity model which was generated first from the CSC server we took the .ODB file and opened it on the Abaqus software. We selected the Visualization Module and selected the option of Create XY Data. We selected the option to operate on the field output of the ODB file. Using Unique Nodal as our position we selected the reaction force and the spatial displacement in the y direction, which is our direction of concern. After selecting the appropriate nodes for the script(See Figure), the XY data for the specimen were created.

Once the XY data was created, we could then select create XY Data again and choose the option of operate on XY Data. We then select all the force elements in the data, take the sum multiply it by 2 as they are in both directions. It is then divided by a 1000 so that the values are in Kilo Newton(KN). These values are saved as the Force. After that, all the displacement from the elements is taken and the average values are saved as the displacement. Both the force and displacement values are then saved to a .rpt file.

We go back to the Create XY Data option and select the option to operate on the ODB field data. We use the Integration Point as our position and this time we select the Triaxiality and PEEQ values as our values of interest. We also select The Max, Min, and Mid Stress Principles. We then select predetermined values for our critical points and create more XY Data. We then select the option to operate on XY Data. The principal values are then added to the .rpt file as well.

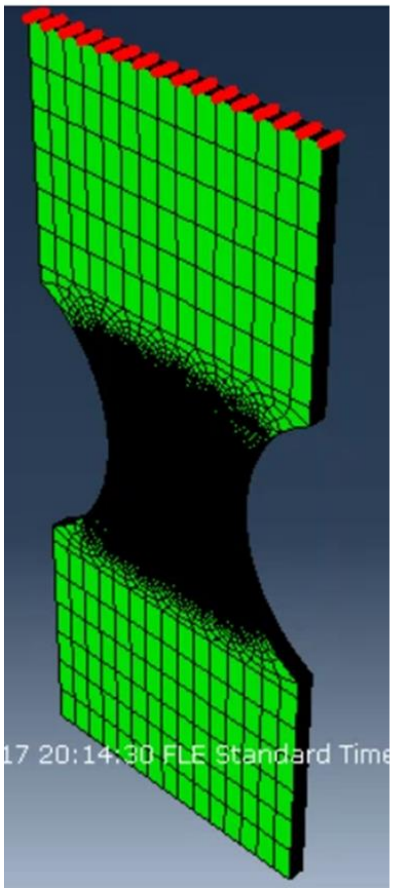


Figure 14: Selected Node points for Displacement

After that is done, we took the working directory of the Abaqus Software and took the generated .rpy file. We converted it into a .py file and made slight changes so that it could be run on the CSC server. We did this same process for all the specimen and used the script developed for all of the damage parameters sets for the each of the geometries. The Script was run on the CSC server and the .rpt file produced was saved on the respective folder as well

4.8 Selecting Fracture Data for Machine Learning:

In order to test out which parameter values are suitable for our end result; we needed a way to measure one force displacement graph with another. We decided that rather than comparing a whole graph with the experimental graph, it would be more efficient to compare some significant points in each of the curves with one another in order to find out parameter values give better result than the other.

We decided to work with the fracture point in each of these curves. The central idea being that if the simulated curves fracture at the same point(or at similar values) than the fracture point, it would mean that the parameter set is the correct value for our enHill model. Therefore, for each of the jobs run, we needed to extract the fracture point of those curves and save it for the purposes of our training data for our Machine Learning model.

Some of the parameter sets(especially with those with extreme values), we found that fracture was not found. In order to find which ones were fractured and which were not, we looked at the damage parameter SDV16 (Ductile damage initiation indicator) and checked if that value was 1 (or anywhere near 1) as the simulation was running. If it was, that meant that fracture was achieved. The fracture points of all these specimens for all of the parameter sets formed the data set for our Machine Learning Model.

4.9 Picking the damage parameters

As CSC resources are limited, we are limited in the amount of job submission and data points that we can later use to train our machine learning. Thus, it is crucial for us to make the most out of the resources by using a combination of damage parameters which yields the most meaningful results.

There are 4 damage parameters for each job, each ranging from 0-3. We want to have parameters from different ranges for example [0-1], [1-2], [2-3] in order to know how different values of a parameters affect the final fracture curve. Also it is important know have parameters from the same range be able to ‘interact’, that is to share the place in the same parameter set, with values from different ranges in order to understand how different value combinations affect the final function.

Our idea is to firstly divide our possible range for each parameters into three ranges, [0-1], [1-2], [2-3]. Then we want to have values from each range for each parameter. For example considering the first parameter, two parameter sets will have it having a random value from 0 to 1, another two will have it having a random value from 1 to 2 and so on. For the remaining three parameters we get random values. This way we fix three parameters and we select one from different ranges, in this case the first parameter. In the end we will have 6 parameters sets by using this method on parameter 1. We do the same for the remaining three, obtaining 24 parameter sets in total for each geometry and 120 different parameter sets in total.

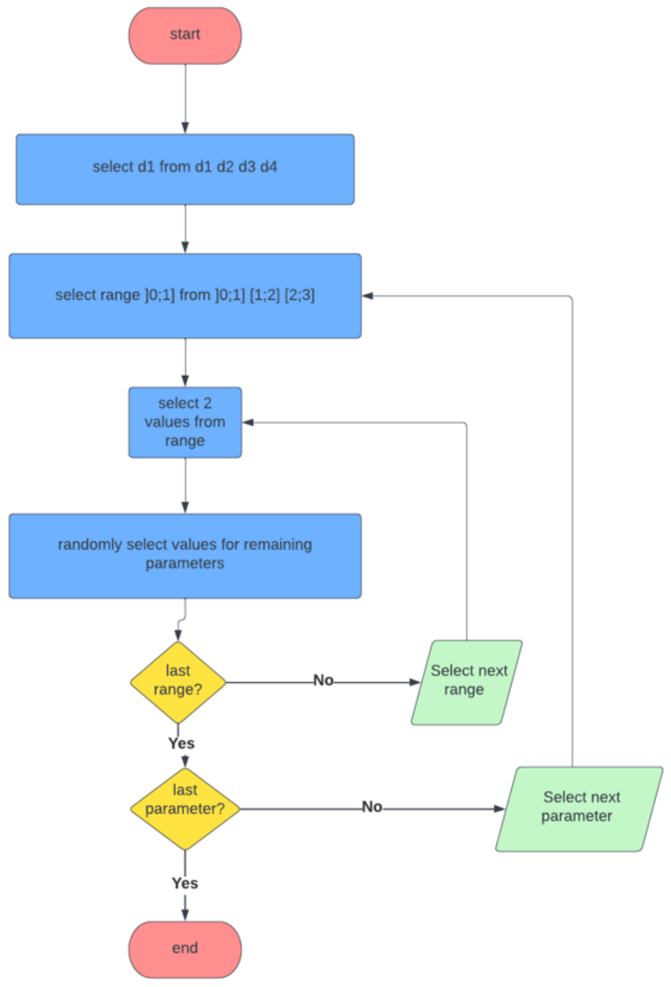

start
select d1 from d1 d2 d3 d4
select range]0;1] from]0;1] [1;2] [2;3]
select 2 values from range
randomly select values for remaining parameters
last range?
No
Select next range
Yes
last parameter?
No
Select next parameter
Yes
end

Figure 15: Flow-chart for picking damage parameters

4.10 Extraction of Plots and Data from data files:

For the produced .rpt, we ran scripts to get and save the relevant data. For all the jobs, their force displacement data was saved and plotted with the experimental values so that we could have visual information about how well some of the damage parameters did compared to other. We used python scripts to plot these values using MATLAB. For all the jobs that had achieved fracture, we took the values (Force, Displacement, Triaxiality etc) at which they achieved fracture(fracture point) and saved that value into a csv file. This was done for each geometry so that every specimen had a csv file that contained all the fracture points from the parameter sets. The Damage parameter values for all of these sets were saved in a separate .csv file. Each of these CSV files combined to from the data for the Machine Learning Model. For all the points that did not reach fracture, zeroes were placed in the CSV files for them.

Some of the Graphs obtained can be seen below:

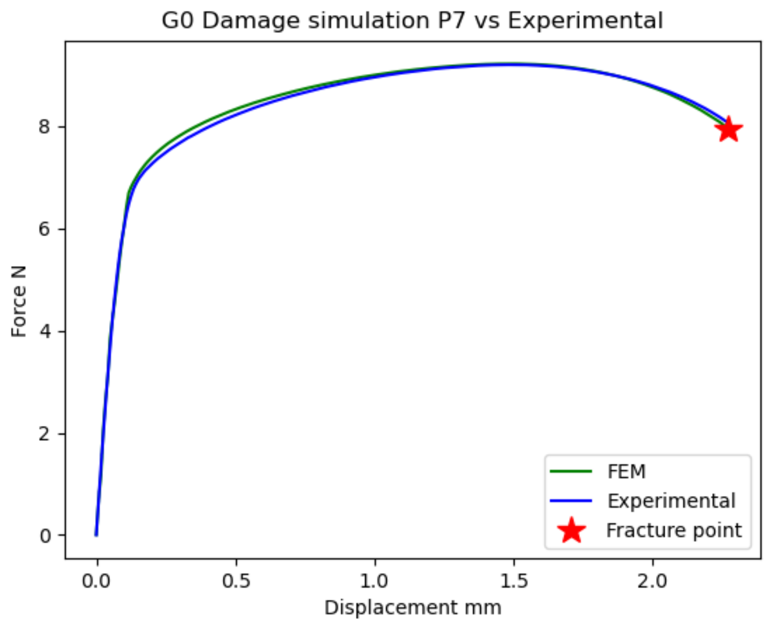


Figure 16, a: Example of a good parameter set in G0

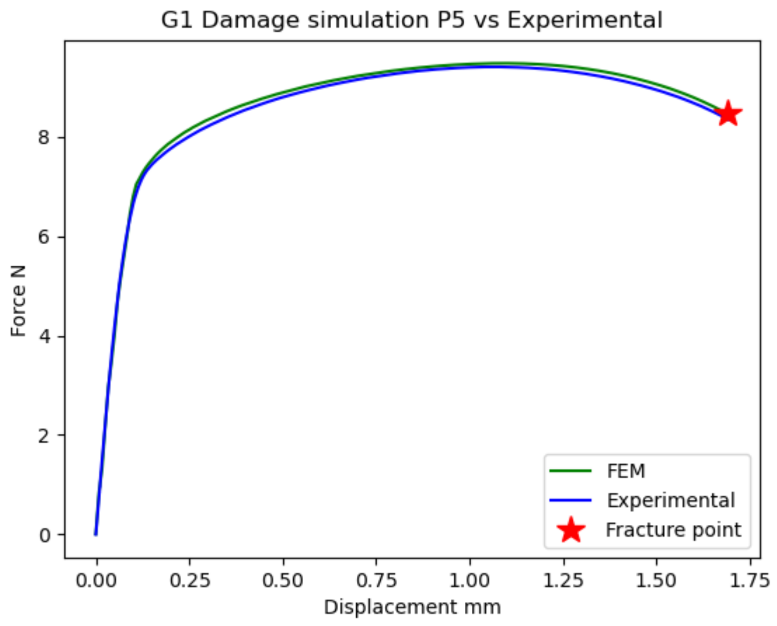


Figure 16, b: Example of a good parameter set in G1

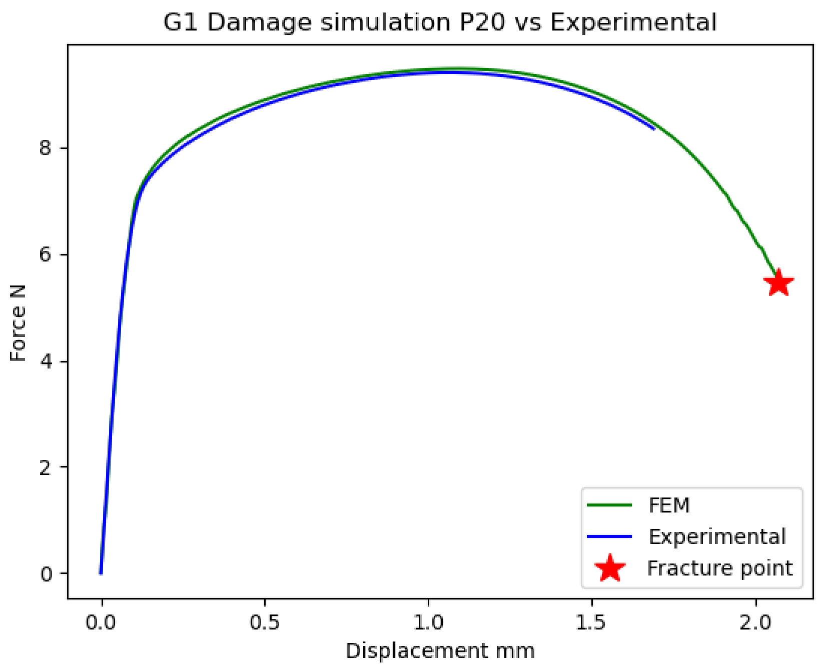


Figure 16, c: Example of a bad parameter set in G1

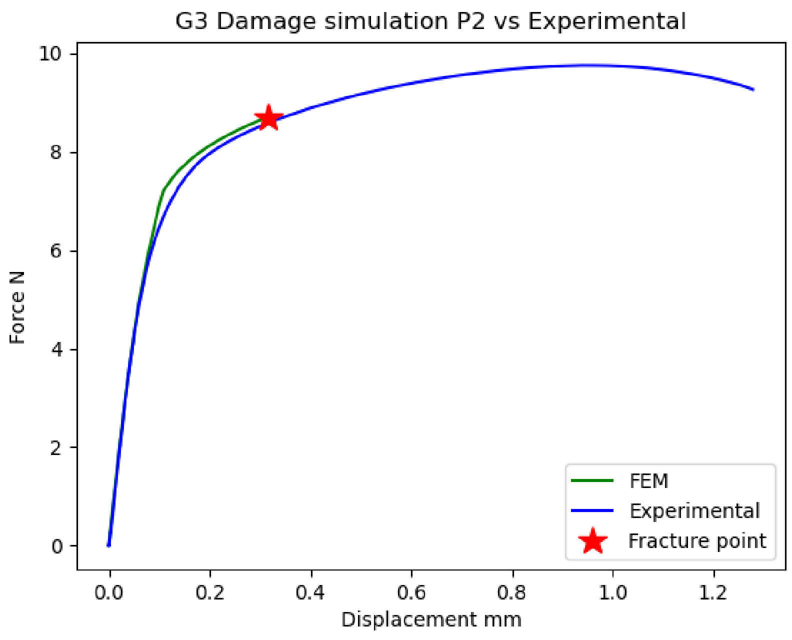


Figure 16, d: Example of a Bad parameter set in G3

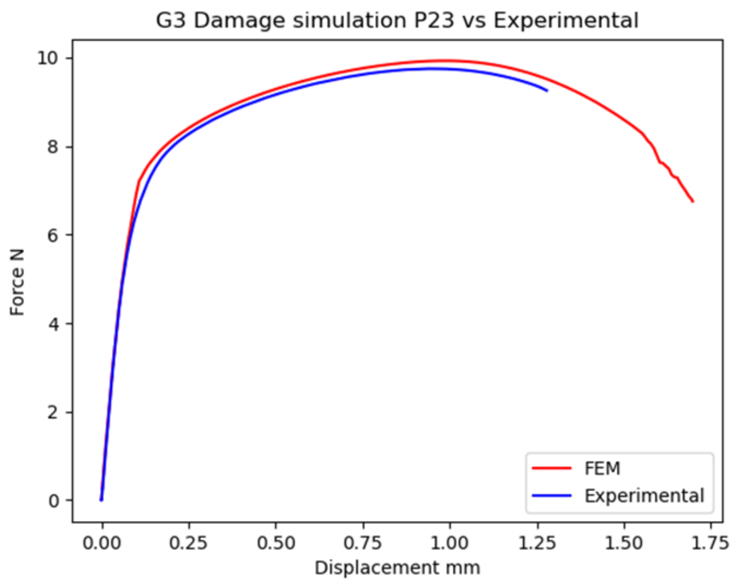


Figure 16, e: Example of a non fracturing parameter set in G3

4.11 Results of Simulation:

As seen from the above model, we were able to get variety of results from our FEM simulation. The plots above compare the experimental data with the FEM data for a particular parameter set. It also shows the point at which the FEM simulation achieves fracture(if it reaches fracture at all). Some were very good parameter sets and managed to fracture around the same point as the experimental results. Others were relatively bad parameter sets as the achieved fracture way before or after the experimental data. Some did not achieve fracture at all and thus were bad parameter sets. An early observation is that when parameter values were set to the extreme, the observed graph would be relatively bad as compared to parameter values in the middle.

Another observation is the results of the simulation we received for G4 and how they compared to the results of the rest od the specimen. The range of paramters set chosen for each of the specimen produced a variety of results. Some were good, some were bad while others did not fracture etc. However, The results produced for G4 were not of the same quality as the rest of the graphs. Many did not fracture and were very far off from the experimental values (see figure 17). This may be attributed to the complex geometry of the G4 model. As figure 18 explains, the model of G4 experinces both compression and tension stress states which makes it difficult for our FEM simulation to get reliable data. One method to get reliable results could have been to go for double precision in our FEM simulation instead of single precision but that would have inreased the time it takes for a job to be completed significantly and would have drained much more resources as well.

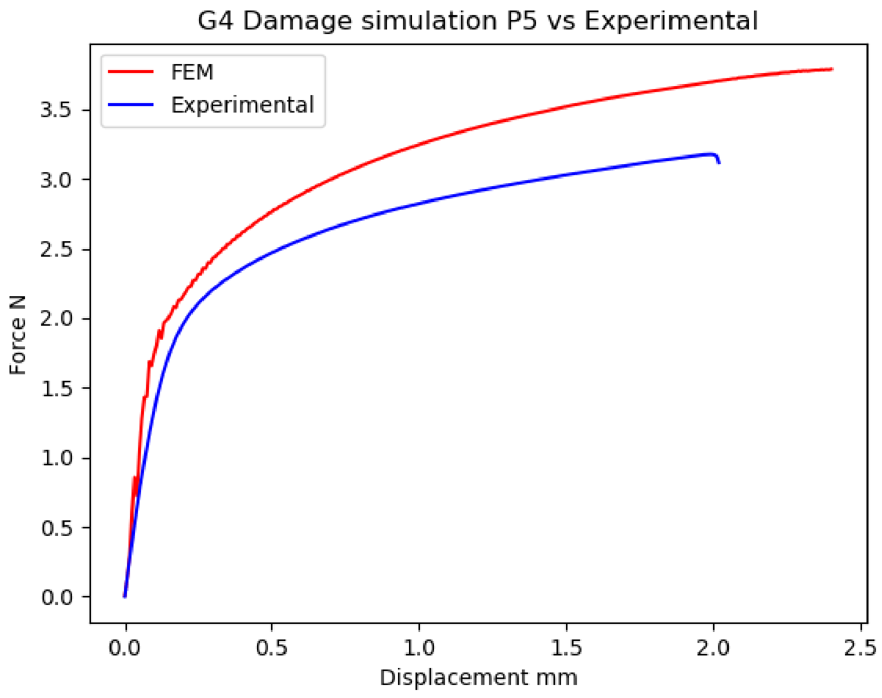


Figure 17: A parameter set of G4

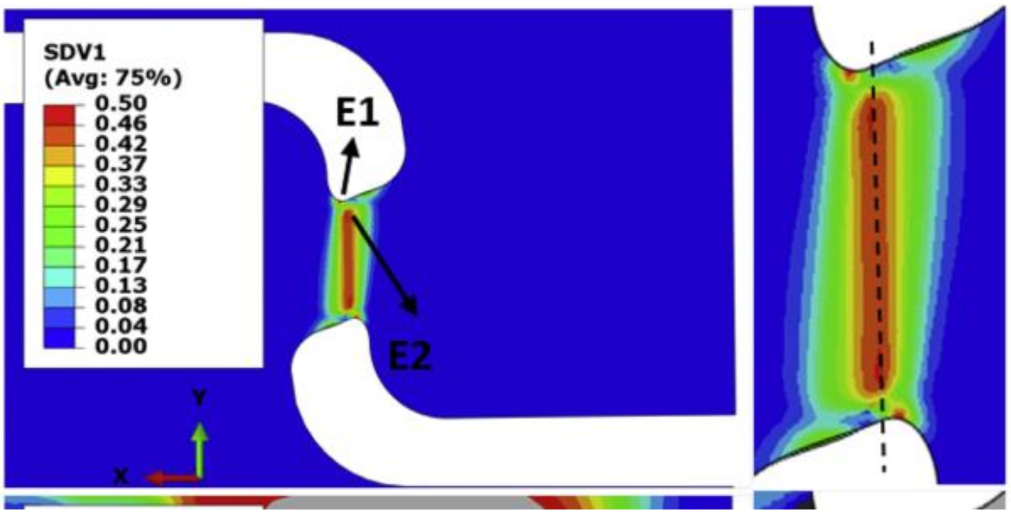


Figure 18: Visualisation of the stress state in G4 [38]

All these parameter sets would be crucial for our machine learning model. The good parameter set would allow our model to learn what could be considered good data for predicting parameter values. Meanwhile, sets that have bad results or do not fracture at all also allow the machine learning model to learn what are the data values to avoid. A accurate loss function that takes both these things into account would be sufficient for our end result.

5 Machine learning

5.1 Understanding the problem

The optimization of damage parameters, according to the literature that we are based upon, is possible by using machine learning models. Our model of choice should be able to understand the complex nonlinear relationship of the outputted force displacement curve by Abaqus with the damage parameters. Our first inputs or features of choice for the machine learning model include the force displacement curves and the local data. Meanwhile the model should output the damage parameters as labels which is what we are interested to find. One of the nuances of the problem that we are trying to solve compared to traditional machine learning problems is that we don’t want to predict the value of a function at a specific point, but rather we want to find the values of constants that are being used in a function of which we have the input and corresponding output value.

In the most simple form of a machine learning model we have a set of input data x and a set of output data y and we want to find y = k(x) at some unknown point x with an unknown function k(). A simple machine learning model such as Linear Regression would be able to solve this problem if k() is a linear function.

In our case we have x and y given and we want to find the underlying relationship between them which is function k(). Function k() is an unknown relationship which uses our four damage parameters and only at the correct parameters will the output of function k() be the same curve as the one obtained by experiment, for each geometry. This relationship is a core part of the material and thus is unchanged between different geometries. Thus, we should be able to find an optimal parameter set which works well with all geometries.

5.2 Selecting the model

Instead of using the whole force displacement curve we decided to use only the fracture points as features. The reason is that is if we used the whole curve there would be too many features leading to overfitting and also since we already got good results with the plasticity models we already have a good fit for before the fracture occurs.

We also decided that we need a model which is able to predict the complex relationship of the force and displacement in the material such as a neural network. As the amount of training data that we have disposable is relatively low we can use a simple library such as Sklean and mlpregressor the Multi Layer Perceptron model inside of Sklean which is used for regression.

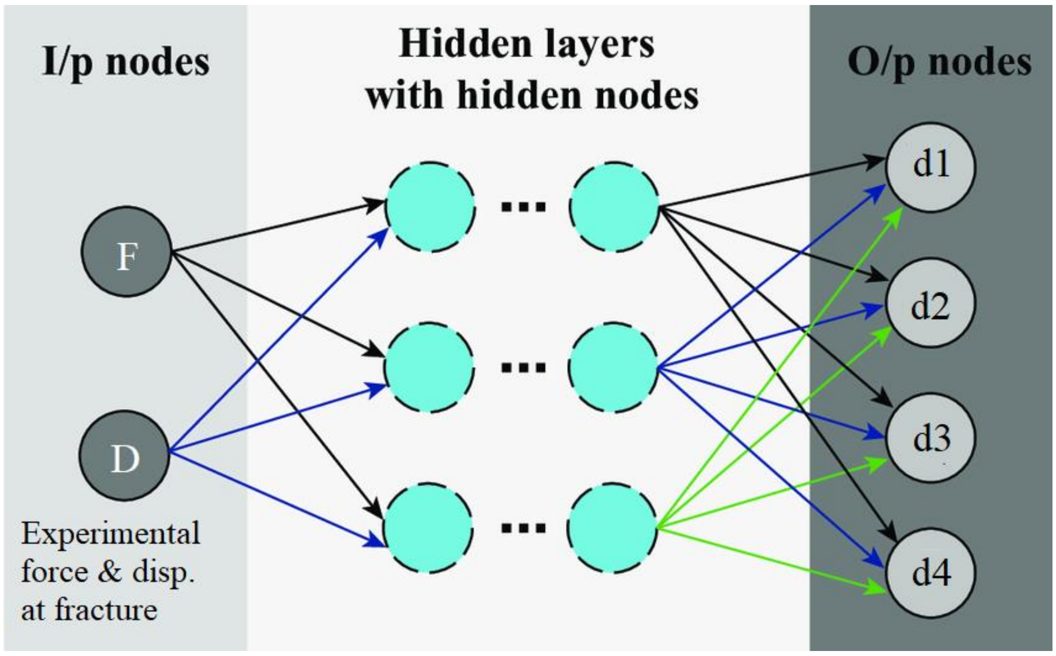

I/p nodes
Hidden layers with hidden nodes
O/p nodes
F
D
Experimental force & disp. at fracture
d1
d2
d3
d4

Figure 19: Visual representation of our machine learning model

5.3 Tuning the Machine Learning models

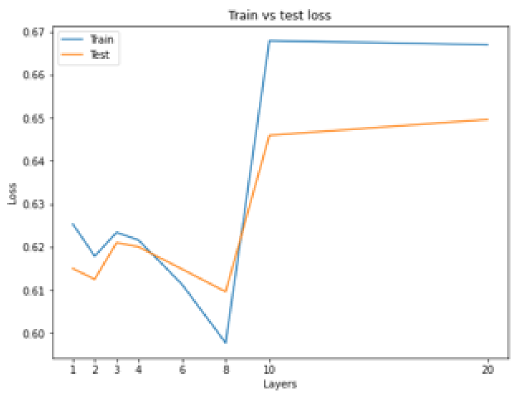
5.3.1 Machine Learning model 1

The first machine learning model was choose to only include global data (force and displacement) as features and the four damage parameters (d_1, d_2, d_3, d_4) labels. Out of 120 total simulation data collected, only 90 data points were used for training the model. This is due to the fact only 90 simulations fractured and the features contained the force/displacement at the fracture points. As a result of not having large number of data points, it would be futile to create a complex model as the model would tend to overfit the training data. Hence smaller model with 3 layers including input and output layer. For the number of neurons we used three, two for the number of features plus one for bias [37]. The 90 data points was further split into traning and validation set, with a split of 33%. Hence in total there is about 60 data points for training and about 30 data points for validation. The predicted damage paramters was (1.46, 1.48, 1.31, 1.36). Using the validation set, mean squared error was calculated from predicted lables and true lables. This was done using the built in function from Sklearn.metrics. The mean squared error for this model is 0.788.

5.3.2 **Machine Learning model 2**

The second machine learning model is similar to the first one. Again it was chosen to only include global data (force and displacement) at fracture as features and the four damage parameters (d1 d2 d3 d4) as labels. In this model it was decided to only include data points which not too far from optimal fracture in order to not negatively effect the model. Only data points where fracture occurred within square root of 2, Euclidean distance from the fracture point were used. Out of 90 fractured simulated specimens only 80 were used for training. In order to select the number of layers and neurons was chosen by trying models with different number of layer and neurons and choosing the ones with the least error.

First models with different amount of layers were tested and the one with the least testing error was the one with 8 hidden layers. After 20 layers the error seemed to converge. After that models with 8 hidden layer and with different amount of neurons were tested. The final number of neurons chosen was 30 neurons. The feature data points were also normalized into a [-1;1] range in order for none of the features to have a bigger influence into the final model. The 80 data points was further split into training and validation set, with a split of 33%. Hence in total there is about 52 data points for training and about 28 data points for validation. The predicted damage parameters were (1.45, 1.47, 1.26, 1.52). Using the validation set, mean squared error was calculated from predicted lables and true lables. The mean squared error for this model is 0.586.



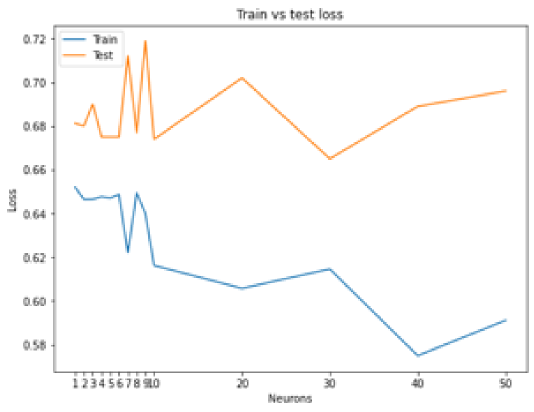


Figure 20, a: Loss - layers plot **Figure 20, b**: Loss - neurons plot

Machine Learning Model 2+

The fourth machine learning model is similar to model 2. For this model we decided to only use displacement at fracture as a feature. The reasoning behind this decision comes from plotting the liinear correlation matrix. From Fig. we can see that Force has relatively low linear correlation with all the damage parameters especially compared to displacement and the local data. The predicted damage parameters were (1.63, 1.42, 1.32, 1.45). The mean squared error for this model is 0.629.

5.3.3 Machine Learning Model 3

Similar to models one and two, model 3 uses MLP regressor to train and predict the optimal damage parameters. The simulation contains global data (force and displacement) and local data gathered from a singular element in the FEM element mesh. It is possible to include all the data as features including global and local data for better prediction.

As an output from the simulation 300 steps or frames are created from the frame 0 until 0.01. Hence there is a total of 301 including 0, possible data points for global and local data. However if the fracture occures eailer to 301, then the frames after the fracture frame has nan or 0 values. Therefore we can assume that beyond the point of fracture the values are irrelelavent. Since some of the dramage parameters cause some of the models to fracture at random or unpridicitable frames, the frames after fracture will be padded with zeros so that the end number would be 301. The local data consists of PEEQ, stress triraxilaty and load. The whole loading path from the starting frame was used as it could help ML model to predict better, rather than only incude data from fracture frame. Hence the final feature array had froce and displacement from fracture frame, then 301 x PEEQ, 301 x stress triaxlity and 301 x load angle, with padding of zeros for the local data. In total there was 90 data point, where a single data point conisted of 905.

Similar to the research done by Yao [23], the number of neurons selected was 14 with 3 layers including input and output layer. The predicted damage parameter given by this model is [1.40, 1.47, 1.24, 1.44]. The mean squared error is 0.682.

5.4 Loading path and fracture locus

5.4.1 Loading paths

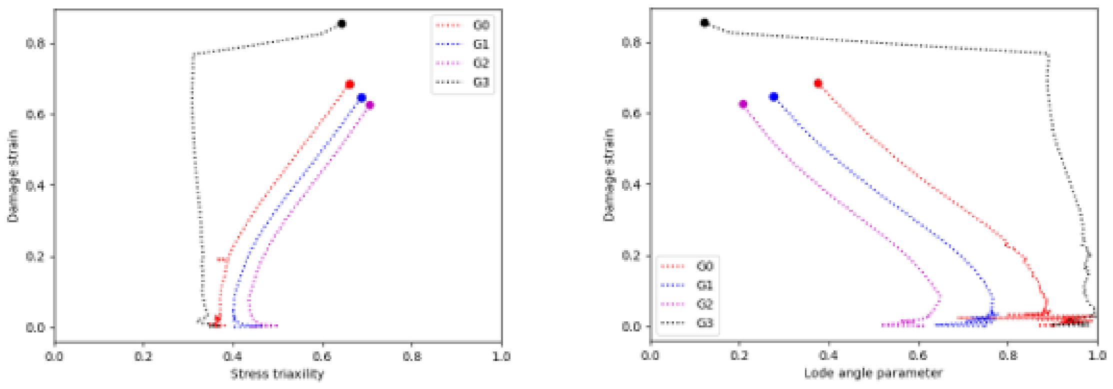


Figure 21, a: Loading path using model 1 results

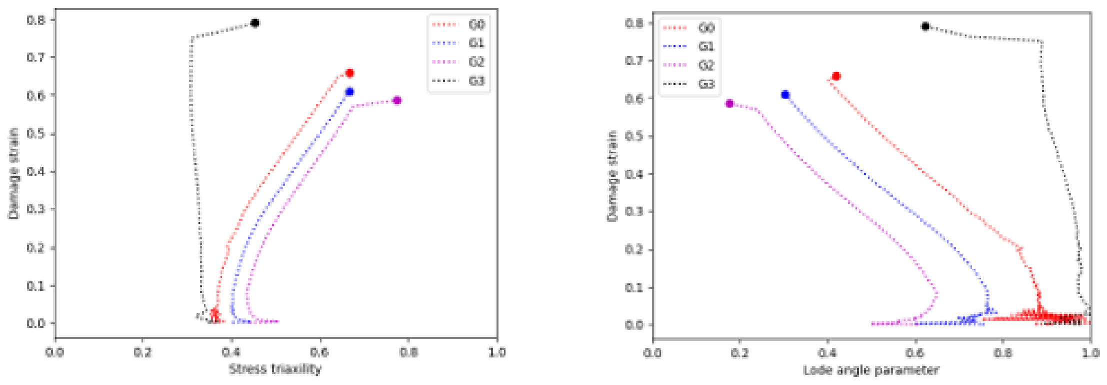


Figure 21, b: Loading path using model 2 results

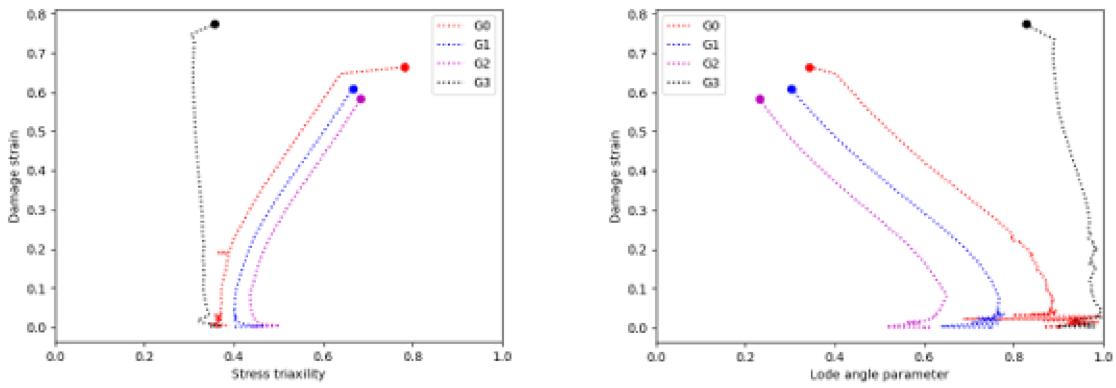


Figure 21, c: Loading path using model 3 results

After getting the parameter sets output from each model, we run a new set of jobs for each geometry for each model. Using the local data outputed from Abaqus we constructed the loading paths for each model. The loading paths follow the same path for majority of frames except for the fracture frame where they all differ by the lode angle or triaxility.

5.4.2 Fracture locus

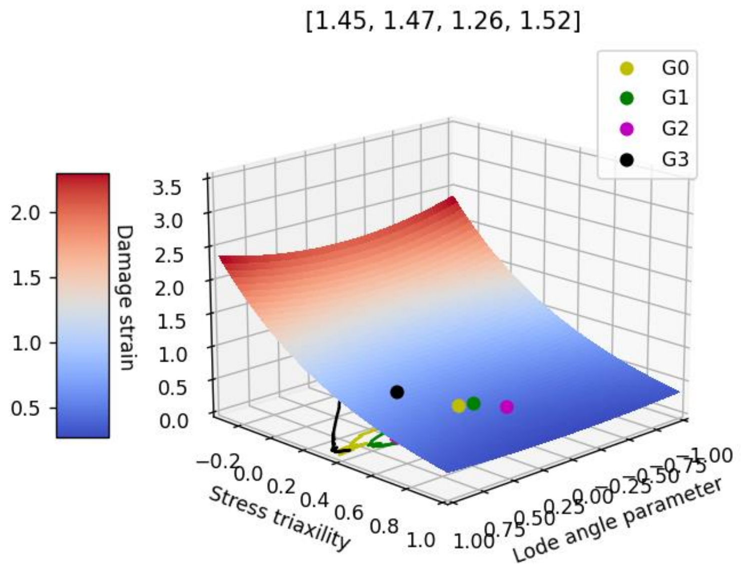


Figure 22: Fracture locus using model 2 results

After plotting the loading paths into 2 dimensional axes, we also plotted the loading paths into 3 dimensions together with the fracture locus figure 22. In order to plot the fracture locus, we used EQ2[40] and the damage parameters from model 2 [1.45, 1.47, 1.26, 1.52].

$$\bar{\varepsilon}_{ddi}(\eta,\bar{\theta}) = (d_1 * e^{-d_2*\eta} - d_3 * e^{-d_4*\eta}) * \bar{\theta}^2 + d_3 * e^{-d_4*\eta} \tag{3}$$

Since it is difficult to observe the closeness of fit for a 3-dimensional fracture locus with the fracture points, we plotted the fracture points against the projection of the locus function on figure 23 (left). Each of the colored points represents a fracture point whereas the curve with similar color to the point represents the values of the fracture locus at the same projection as the point.

5.4.3 Surface fitting

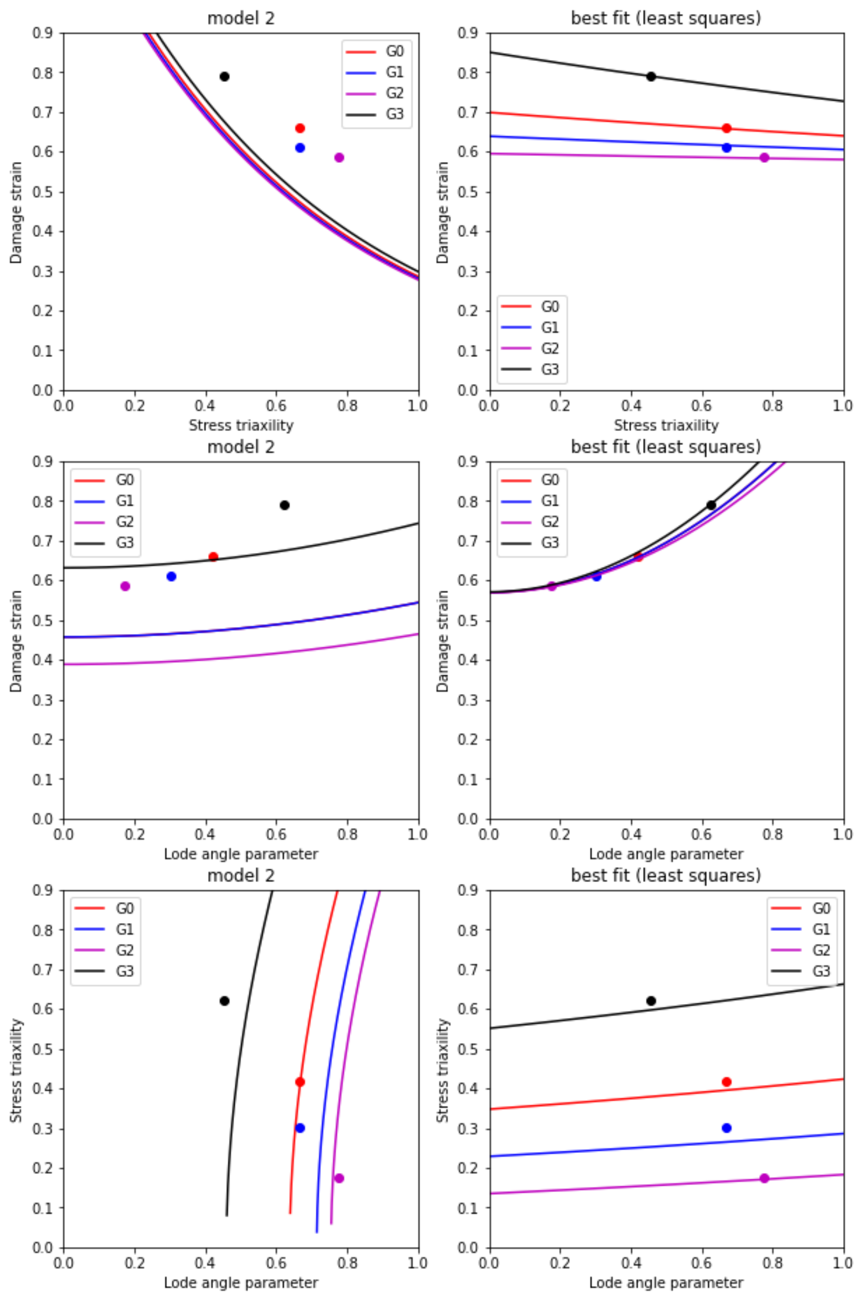


Figure 23: 2d projections of our fracture locus (left) and projection of optimal fracture locus from surface fitting (right)

The fracture locus from the damage parameters obtained from our model were not optimal when compared to the fracture points. In order to obtain a fracture locus that passed through all the fracture points, we used least squares in order to find the best fit for the locus surface and the fracture points. The results can be seen as projections on figure 23 (right). Least squares works by changing the damage parameters (d1,d2,d3,d4) in order to find the surface with the least squared error from the fracture points. By using least squares, we obtained the damage parameters [1.27 0.29, 0.61, 0.01]

6 Results and discussion

6.1 Machine Learning model 1 results

With the predicted damage parameter, another set of simulation was ran.

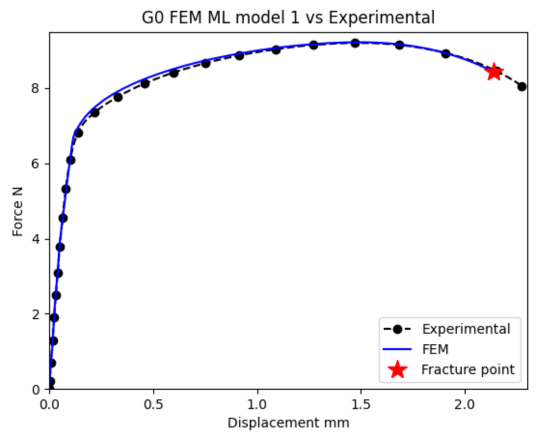


Figure 24, a: G0 FEM vs Experimental Model 1

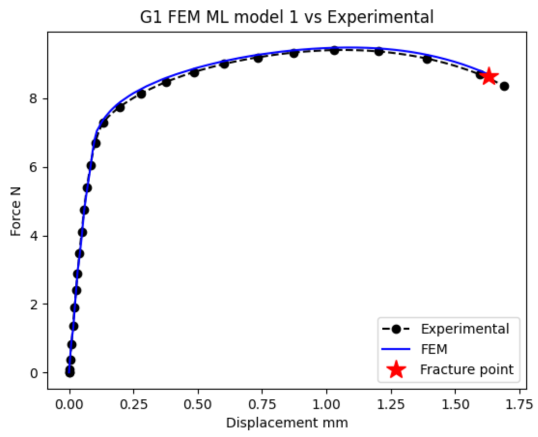


Figure 24, b: G1 FEM vs Experimental Model 1

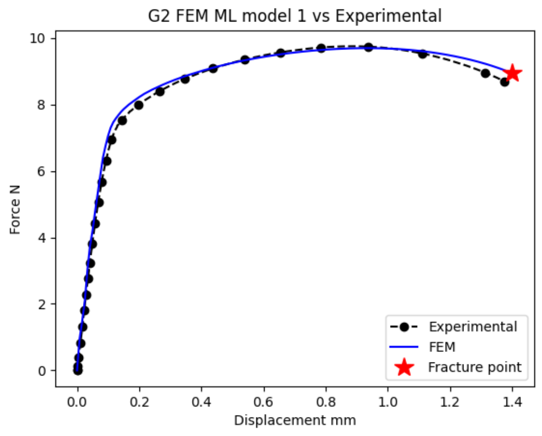


Figure 24, c: G2 FEM vs Experimenal Mode 1

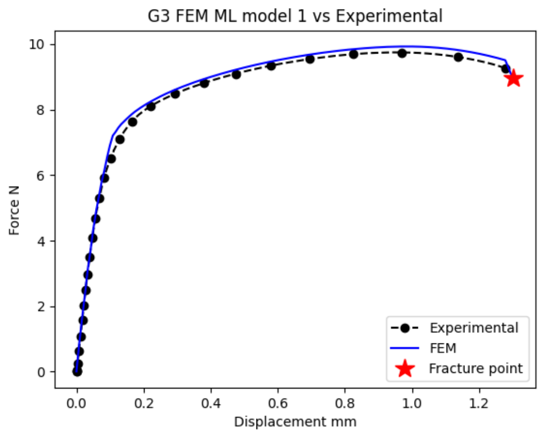


Figure 24, d: G3 FEM vs Experimental Model 1

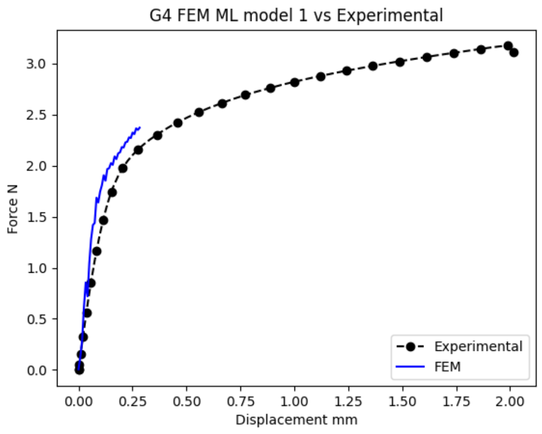
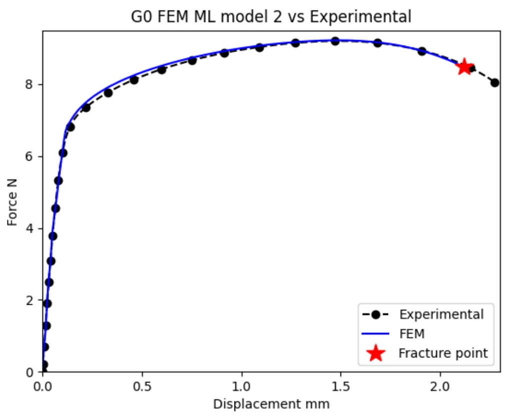


Figure 24, e: G4 FEM vs Experimental Model 1

Figure 24 shows the FEM simulation using the predicted damage parameters by the ML model 1. While the black dotted line decribes the experimental curve, the blue curve describes the simulated curve using the model 1 damage parameters. It can be observed that G0 and G1 fractures much before that their respective experimental fracture points. However figure 24 c and d with respresents G2 and G3 geometries, have better fracture point. However G4 has some error finishing the simulation hence only a part of the curve is visible.

6.2 Machine learning model 2 results



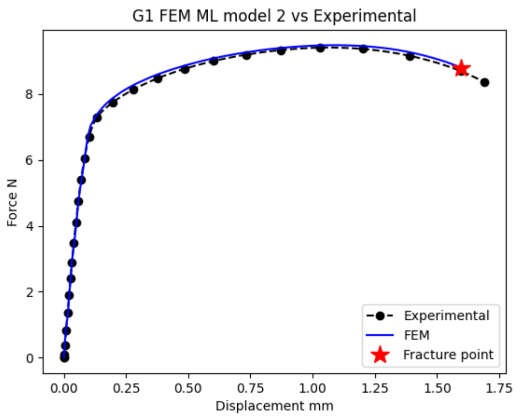


Figure 25, a: G0 FEM vs Experimenal Model

Figure 25, b: G1 FEM vs Experimental Model 2

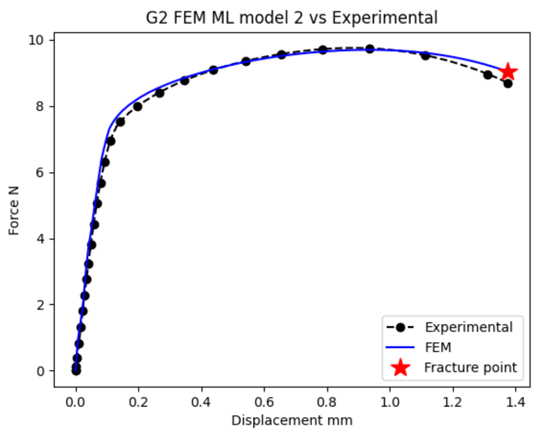


Figure 25, c: G2 FEM vs Experimental Model 2

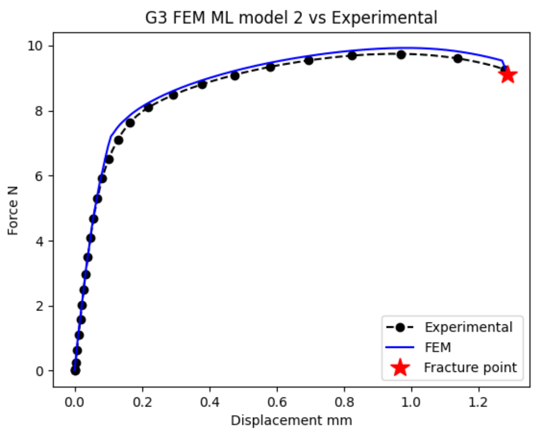


Figure 25, d: G3 FEM vs Experimenal Model 2

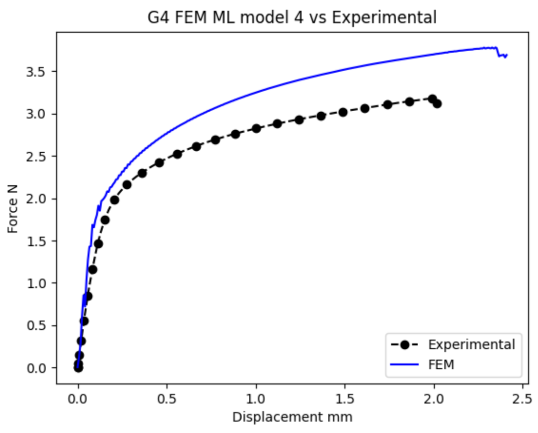


Figure 25, e: G4 FEM vs Experimenal Model 2

Figure 25 shows the FEM simulation using the predicted damage parameters by the ML model 3. Similar to model 1 results, graphs for G0 and G1 seem to quite similar. However, graphs for model G2 and G3, ML model 2 has better prediction closer fracture force and displacement prediction compared to model 2. The damage prameter predicted by this ML model allows G4 to fully complete the simulation. However the simulation does not fracture.

6.3 Machine Learning model 3 results

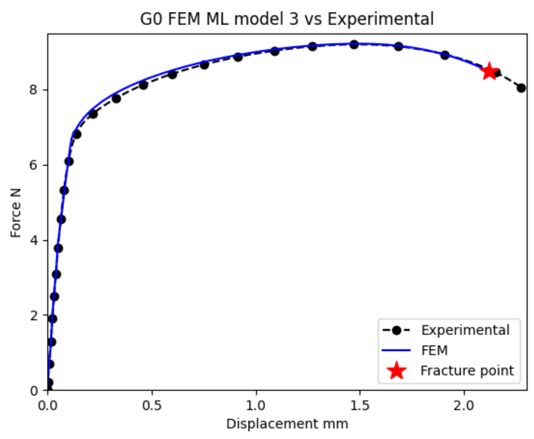


Figure 26, a: G0 FEM vs Experimenal Model 3

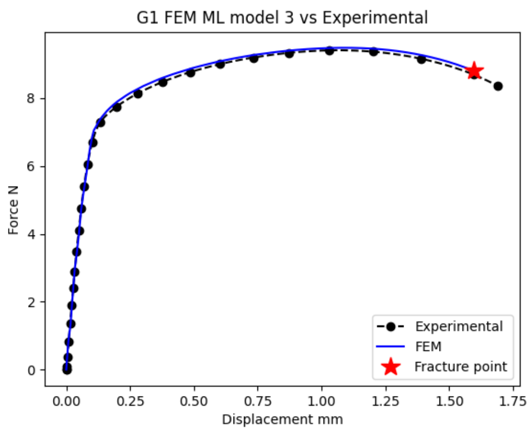


Figure 26, b: G1 FEM vs Experimental Model 3

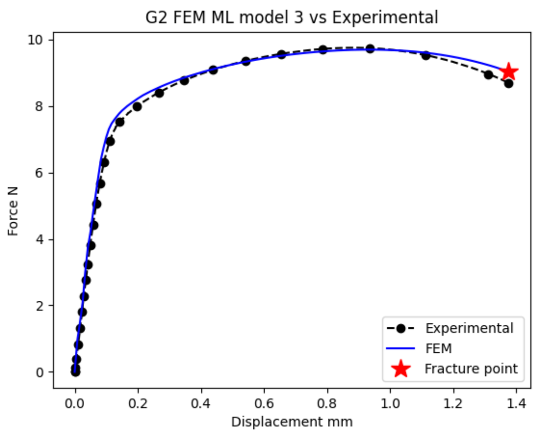


Figure 26, c: G0 FEM vs Experimenal Model 3

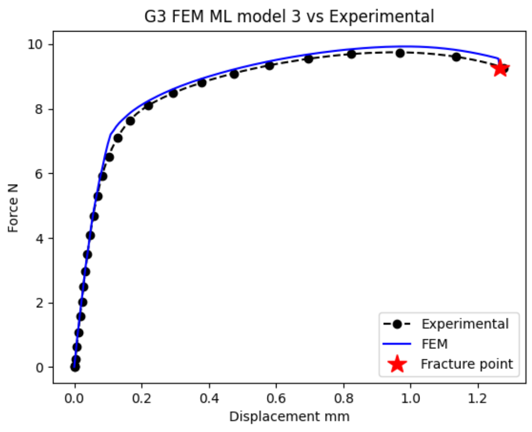


Figure 26, d: G1 FEM vs Experimental Model 3

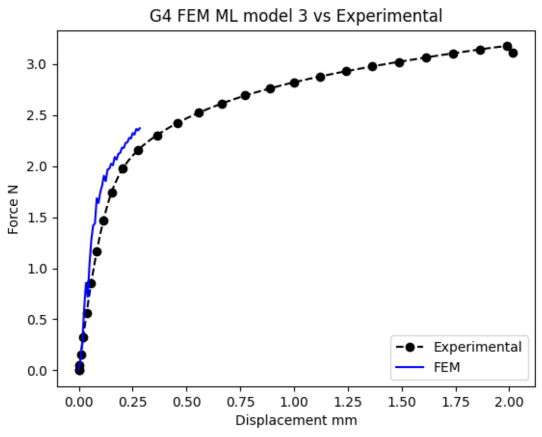


Figure 26, e: G4 FEM vs Experimental Model 3

Figure 26 shows the FEM simulation using the predicted damage parameters by the ML model 3. Similar to model 1, the graphs for G0 and G1 look quite similar. However, graphs G2 and G3 for ML model 3 seems to have a closer prediction to the actual experimental fracture compared to model 1. Similar to model 1, figure 26 e which represents the G4 geometry also could not comple the simulation due to the error from Abaqus.

6.4 Machine Learning model 2+ results

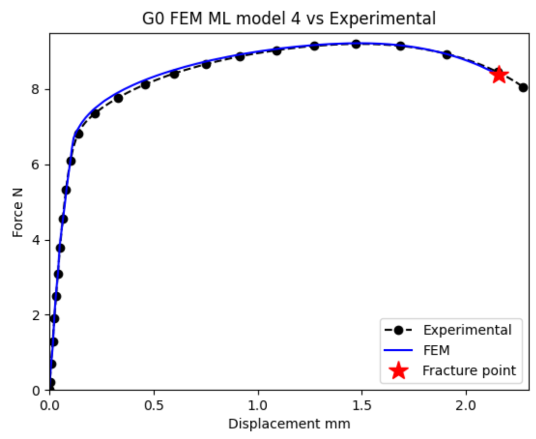


Figure 27, a: G0 FEM vs Experimenal Model 2+

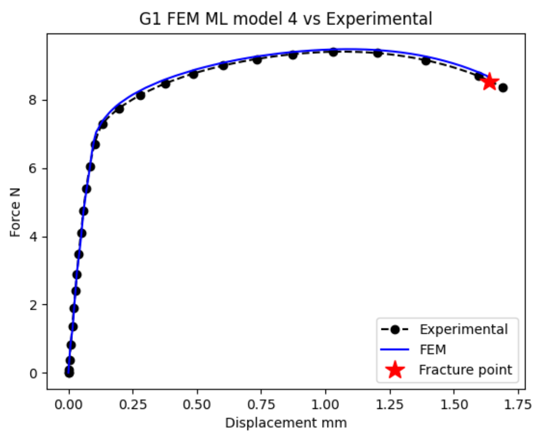


Figure 27, b: G1 FEM vs Experimental Model 2+

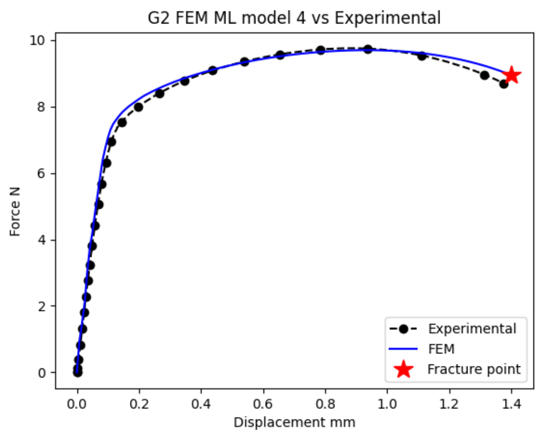


Figure 27, c: G2 FEM vs Experimenal Model 2+

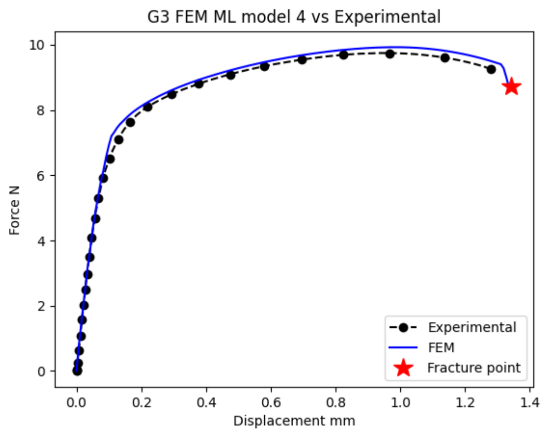


Figure 28, d: G3 FEM vs Experimental Model 2+

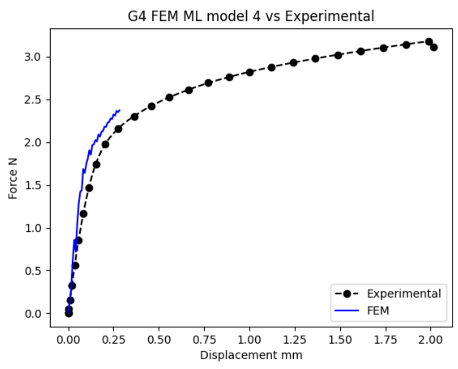


Figure 28, e: G4 FEM vs Experimental Model 2+

Figure 28 shows the FEM simulation using the predicted damage parameters by the ML model 2+. It can be observerd that although the change is not much in model 4 compared to model 1, 2, 3. The fracture point in graphs for G0 and G1 in model 2+ are little more closer to the experimental fracture point. The graph for G2 has a similar fracture force to model 3, but can the fracture displacement is much better. Similar problem to model 1 and 3 can be observed, where G2+ does not finish the simuation but rather stops with an error.

6.5 Surface fit damage parameters results

The fourth damage parameter using curve fit was 0.01. Since this is the lowest possible bound for d4, it was decided to run simulation for d4 as 0.01 and 0.1 to lessen the effect of lowest possible bound.

6.5.1 Surface fit with d4 as 0.01

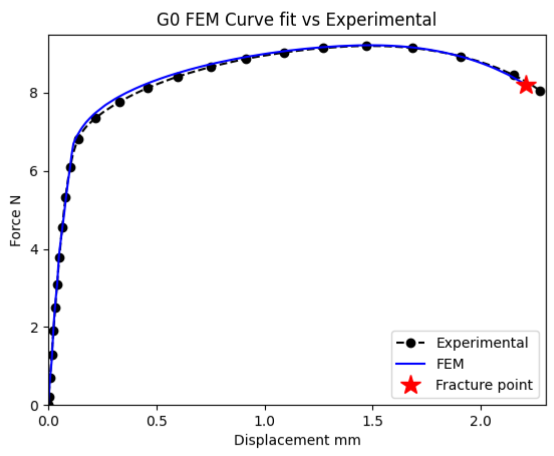


Figure 29, a: G0 FEM vs Experimenal Curve Fit, d4 = 0.01

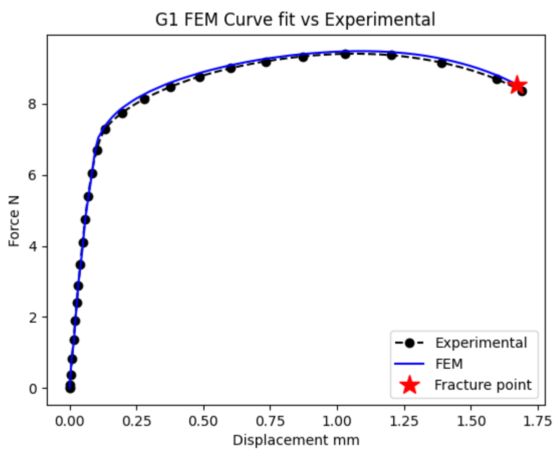


Figure 29, b: G1 FEM vs Experimental Curve Fit, d4 = 0.01

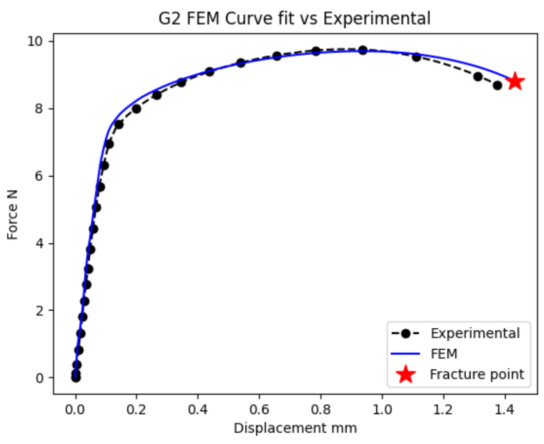


Figure 29, c: G2 FEM vs Experimenal Curve Fit, d4 = 0.01

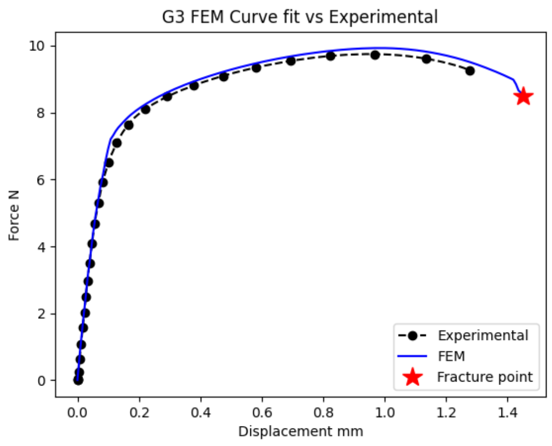


Figure 29, d: G3 FEM vs Experimental Curve Fit, d4 = 0.01

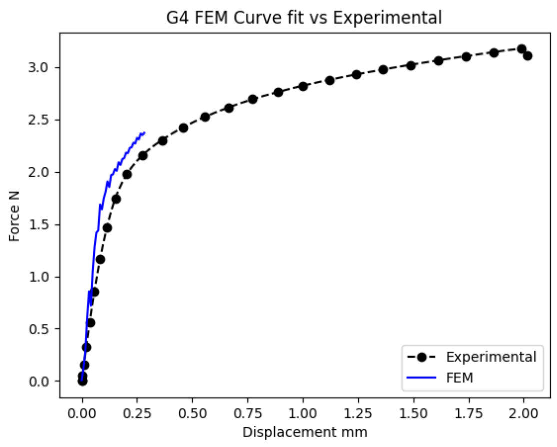


Figure 29, e: G4 FEM vs Experimental Curve Fit, d4 = 0.01

Figure 29 shows the FEM simulation using the predicted damage parameters by the surface fit. Similar to model 1-3, graphs for G0 and G1 has little to no difference and graph for G4 fails with an error at the same frame as previous models. However G2 and G3 have oversetimation in fracure force. While G2 has overstimation in fracture displacemnt G3 has understimation.

6.5.2 Surface fit with d4 as 0.1

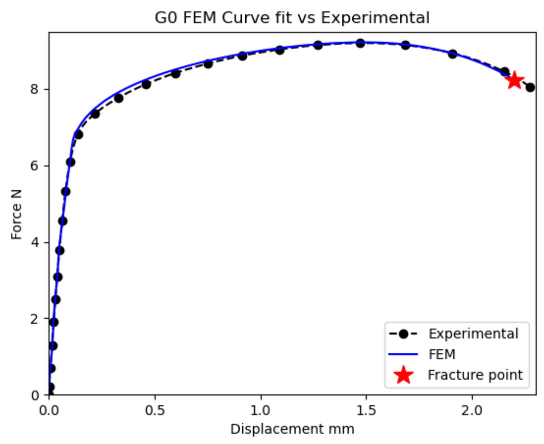


Figure 30, a: G0 FEM vs Experimenal Curve Fit, d4 = 0.1

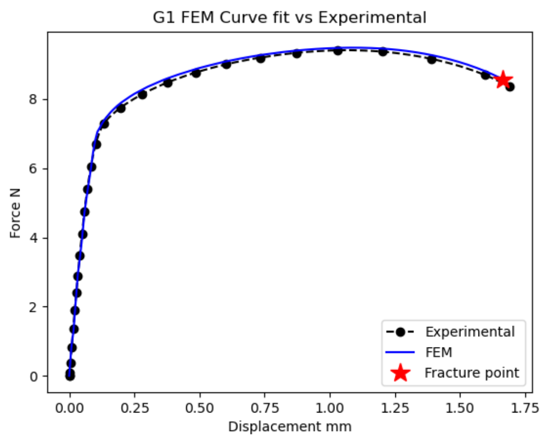


Figure 30, b: G1 FEM vs Experimental Curve Fit, d4 = 0.1

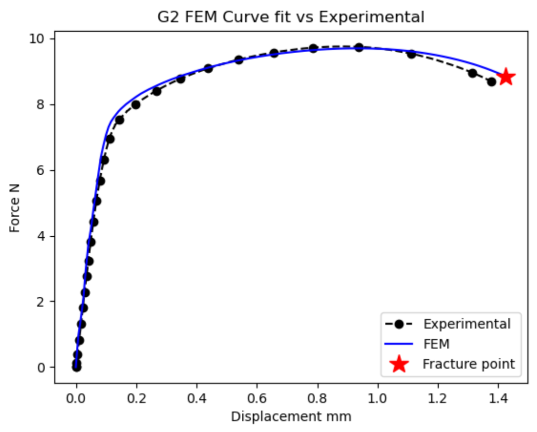


Figure 30, c: G2 FEM vs Experimenal Curve Fit, d4 = 0.1

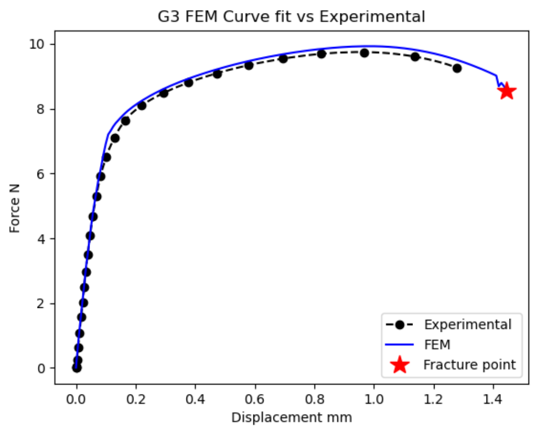


Figure 31, d: G3 FEM vs Experimental Curve Fit, d4 = 0.1

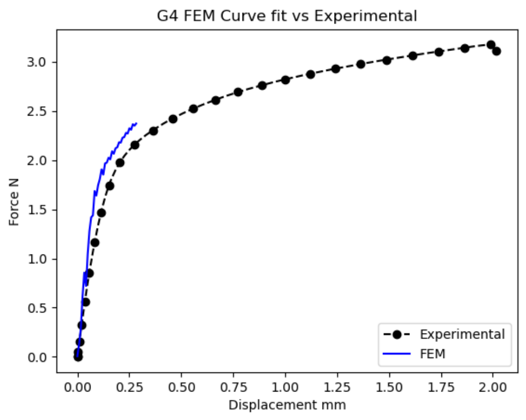


Figure 30, e: G4 FEM vs Experimental Curve Fit, d4 = 0.1

Figure 30 shows the FEM simulation using the predicted damage parameters by the surface fit. Similar to curve fit with d4 = 0.01, figures 30 a-d are very similar to their counterpart figures 29 a-d. Graph for G4 alos fails at the same frame with an error. However by changing d4 for 0.01 to 0.1, graph 42 which represents G3 seemes to be the most affected.

6.6 **Overall Results**

The data extracted from the FEM simulation was used in various ways by our machine learning models, each of which gave interesting and comparable results.

Table 3: Comparison of Machine Learning models

Machine Learning Model	**Mean Squared Error**	**Features Used**	**Damage Parameters Predicted(D1-D4)**
Model 1	0.788	Force, Displacement	[1.46, 1.48, 1.31, 1.36]
Model 2	0.586	Force, Displacement	[1.45, 1.47, 1.26, 1.52]
Model 3	0.682	Force, Displacement, PEEQ, TRIAX, Load Angel	[1.40, 1.47, 1.24, 1.44]
Model 2+	0.682	Displacement	[1.40, 1.47, 1.24, 1.44]

Throughout the entirety of our project, visual data obtained from the obtained plots were also used as a measure to see how well the FEM Models were compared to the actual data. The results for each model for each specimen can be seen on the figures shown above for each of the Machine learning Models.

As the model performance cannot analyzed accurately using mean squared error. Euclidean distance was calculated for each of the ML model, and within each ML model for each geometry. Eucaledian distance was calculated using the simulated fracture point and acutal or experimental fracture point.

Table 4: Tabulated result of Eucaledian distance ML model 1

	Eucaledian distance
G0	0.396
G1	0.314
G2	0.245
G3	0.278
Average	0.308

Table 5: Tabulated result of Eucaledian distance ML model 2

	Eucaledian distance
G0	0.451
G1	0.458
G2	0.339
G3	0.144
Average	0.348

Table 6: Tabulated result of Eucaledian distance ML model 3

	Eucaledian distance
G0	0.450
G1	0.461
G2	0.337
G3	0.011
Average	0.315

Table 7: Tabulated result of Eucaledian distance ML model 4

	Eucaledian distance
G0	0.341
G1	0.203
G2	0.245
G3	0.521
Average	0.328

Table 8: Tabulated result of Eucaledian distance Curve fit with d4 = 0.01

	Eucaledian distance
G0	0.169
G1	0.193
G2	0.125
G3	0.790
Average	0.319

Table 9: Tabulated result of Eucaledian distance Curve fit with d4 = 0.1

	Eucaledian distance
G0	0.199
G1	0.226
G2	0.154
G3	0.721
Average	0.325

6.7 Comparision of the results

All the models gave comparable results for each of the specimen when tested out with one another. The final damage paramter predicted by each model converge at similar points. The value of D2 for example, in all the paramters is nearly the same and so the optimum value could be estimated to be 1.46. All the other results as well are around one decimal place within each other. This gives can give furher credibility to our results as various models trained differently seem to predict similar values. Model 2 seems to give the lowest Mean-squred error out of all the ML models at 0.586. The mean squared error itself is measured against the validation set in the data. Model 1 and 2 give different mean squred errors even though they use the same features.

The Euclidean distance is also a measure that we used to calaculate the accuracy of our results. Essentially, the we calaculate the euclidean distance between the point of fracture of the FEM Data and the experimental data. Model 1 seems to have the lowest average euclidean distance between all the specimen but all the models seems to have a similar value across the board. For the damage paramters calculated by curve fitting, the euclidean distance between for G0-G2 is actually significantly lower than all the other ML models however the distance for G3 is significantly higher so the average distance seems to be the same for all of the models. The distance for the G4 specimen could not be calculated as it does not achieve fracture for any of the damage paramters predicted in the model.

Overall, no one model provides results that are better than the other. Some models work better for certain specimen and vice versa. The mean squared error and the average euclidean distance remain in close proximity to each other for all the models. More investigation and experiment would be needed to fully determine if fun model is better at predicitng damage paramters than the other.

6.8 **Correlation and** Principal Component Analysis (PCA)

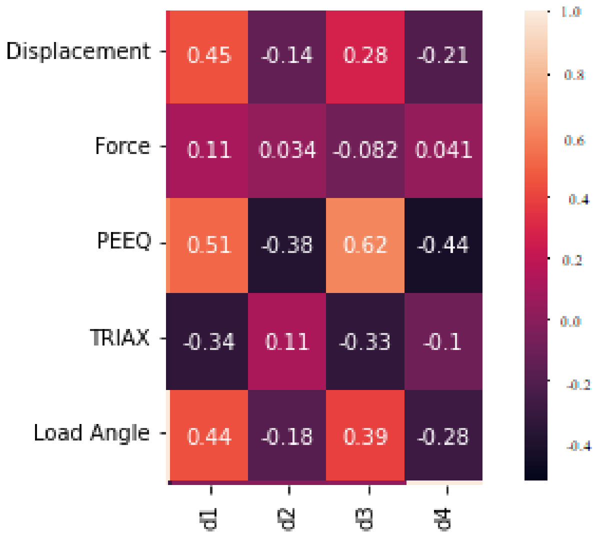
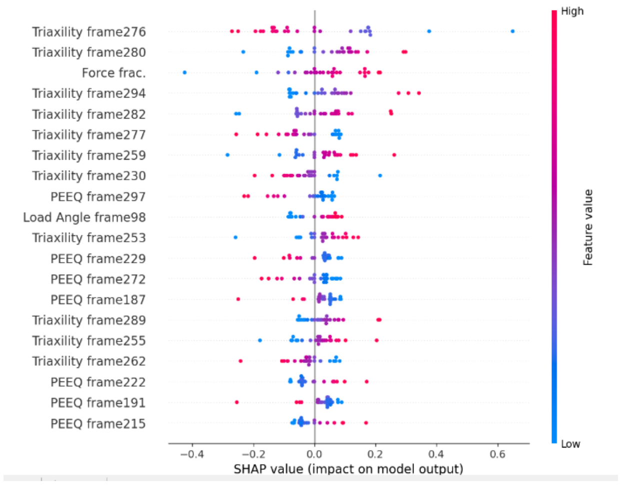
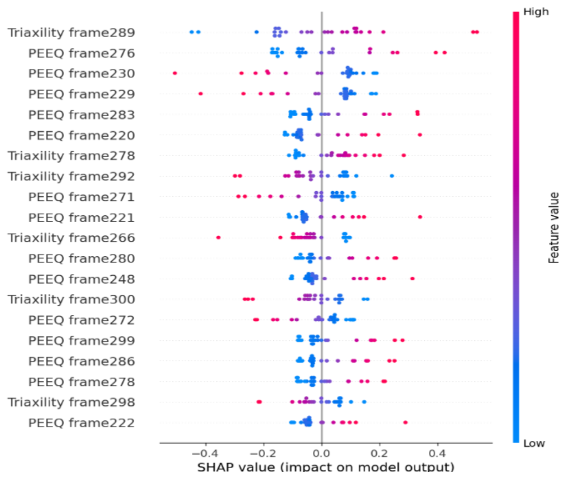


Figure 31: Correlation matrix betweem damage paramters (d1-4) and local + global data

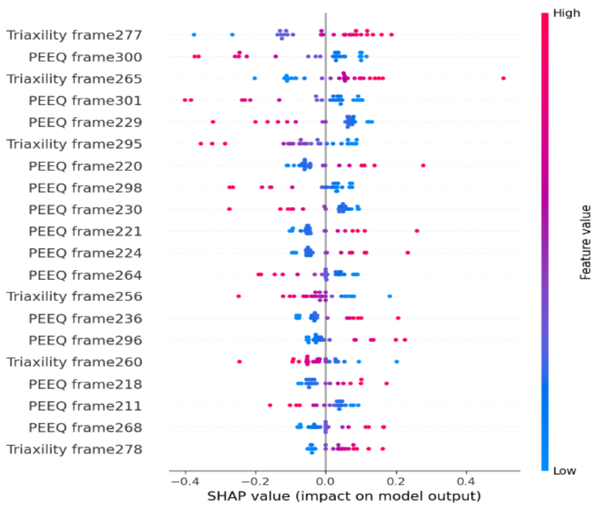
One important metric used to decide on the features used was the linear correlation between the features and the damage parameters. The correlation matrix on Fig. 31 shows that force is the only feature that has comperatively low linear corelation (the absolute value is at most 0.1) for all damage parameters. The same Figure also shows that Displacement, PEEQ, Triaxility and Lode Angle have useful linear correlation (absolute value is at least 0.1) for all geometries. As the relationship between the features and the damage parameters is more complex than linear it is useful to check for other types of correlations.



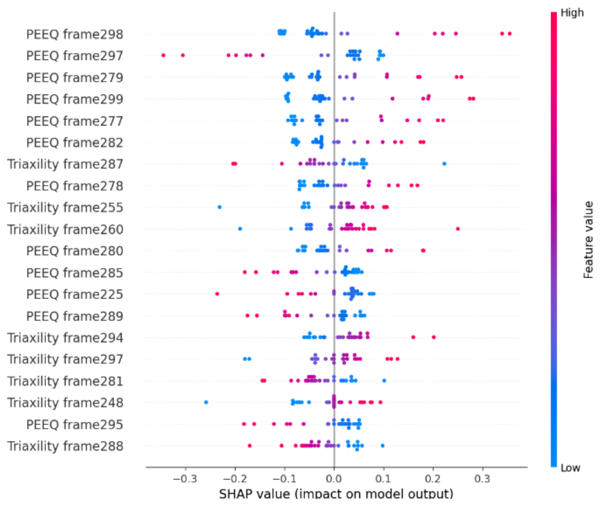
a)



b)



c)



d)

Figure 32: Shap summary plots for model 3 for damage parameters: a) d1; b) d2; c) d3; d) d4.

For model 3 we use a large amount of features as we include all the frames that are part of the loading path for each geometry and the force-displacement at fracture. In order to see which frames where the most important in the resulting output for model 3 we used the SHAP (SHapley Additive exPlanations) python library. SHAP is a game theoretic approach used to explain the output of machine learning models [41]. We used the SHAP Kernel Explainer in order to explain our multi layer perceptron model (model 3). The results from the SHAP summary plot can be seen in figure 32 for each of the damage parameters. From the figure we can see all the most important features for each damage parameter sorted by the importance in the model prediciton. One important factor to notice is that for PEEQ, Trixaility and Lode angle the most important frames are the last frames close to fracture.

6.9 Challenges and Improvements

We had to face several obstacles during the entire course our project for which we had to find unique solutions for compromissions. The G4 model in each of the machine learning model gave inaccurate results which were very relatively inaccurate for the purposes of our experiment and gave radically different results when compared to the other specimen. Some reasons for the bad results could be the lack of data for the G4 model as most of them did not fracture for the Data collection phase. Another reason could be a fault in the Abaqus model which prevents the proper FEM Simulation of the G4 model.

Another challenge we faced was the low amount of training data that we had at our disposal for our machine learning model. Although we had 120 jobs, as not all of them fractured, hence we were only able to use about 90 or so datapoints. Each job took several hours to complete and thus took significant time and resources. In the end, this severely limited the amount of data we had and affected how well our machine learning models could have performed.

Another obstacle we faced early on was the abandonment of the Johnson Cook learning Model that we had in favor of the enHIll48 model due to the problems in the Abaqus software. This caused a setback in the speed of our deliverables.

In general, our machine learning models were able to predict the fracture point of many of the specimen to a relatively accurate degree. Some models gave better results for some specimen while other remained the same. A number of potential improvements could be considered to our methodology in order to give even more reliable results.

The Swift-Voce curve was a crucial aspect of our project ;however, in order to achieve more accurate results, one could consider optimizing it even further. This could involve trial and error with different optimization values, but one could also consider using Machine Learning to find out optimized parameters.

Another major improvement could be brough by increasing the number of datapoints that are available for machine learning. One has to consider the significant amount of time and effort it takes to get usable results from Abaqus, and so new ways would have to be investigated to get reliable results quicker. The jobs that would be run with different parameters would also have to cover a wider range of parameter values so that the machine learning model has more data to work with. It would also be helpful to select parameter values which give a higher probability of fracture occurring in order to make sure more data is usable. In our experiment we used different datapoints for different specimen, however using the same parameter set for different specimen and then using that data may help us get better results as the machine learning will be better able to tell how each paramter set affects each set of experimental data.

Finally, experimenting with the Machine learning model itself can be an efficient way to improve on the experiment. This could mean experimenting with different number of hidden layers and neurons for the artificial neural network, but it could also mean trying out different machine learning models altogether and testing the data with them. Different Neural networks could be used in order to predict the parameters which would predict the locus fracture point. One other improvement could be to use an award system place so that the machine learning model is better able to recognise the bad and good paramter sets and so that it has a better way to being able to predict good paramter sets.

7 Conclusion

7.1 Overview

It is evident that machine learning can be useful in predicting the damage parameters. However, the data structure for the machine learning must be considered carefully as it has a direct impact on the quality of the prediction.

The aim of this report was to use machine learning to optimize the damage parameters for QP1000. For this 120 data points or simulations were collected and out of that only 90 were used. These simulation data points were used to train several artificial neural networks called multi-layer perceptron. Using the trained ML model, predicted label or damage parameter were then used for further visual validation. G2 and G3 had the best predicted curve compared to their experimental data. However, due to errors in compatibility and the new software used for simulations G4 was unable to finish with most of the predicted damage parameters. For G0-G2 they were quite like each other. Hence, with limited data the ML models were able to give a good estimation of damage parameter. However, further fine tuning is required to be able to get a final optimal damage parameters for QP1000.

In conclusion, this report was able to achieve the aim partly. This is shown through the fact that machine learning can indeed be used to achieve a good degree of accuracy, while reducing computational resources. Although there were shortcomings, these can be minimized and improved upon.

7.2 Next Steps

Aside from focusing on the improvements highlighted in the improvements sections the next steps from this experiment would be to test machine learning models from other damage models as well. Aside from the enHill model, other models could give better information about the fracture points and may be better suited for machine learning analysis.

One further step could be to try this methodology on other materials as well. Our project focused on the QP1000 material, and all the specimens were made from that. Hence, it could be worthwhile to test the machine learning approach on specimen of other materials as well to increase the scope of this project. At the same time, focusing on other geometries as well could prove to be fruitful. G4 in our experiment proved to be the result which gave the most unreliable results so it could be worthwhile to focus on specimen which can be run better in Abaqus and produce results which train our machine learning models on a range of different geometries.

Using our methodology in new areas while refining it can ensure that we are better able to build models which have high commercial and practical use that can increase productivity in the material sciences field.

7.3 Remote Repository

Aside from the Appendix below, one can aslo use the public repository our team made on GitHub in order to investigate our results more carefully and use the scripts we have made for this entire project as well as the recordings we prepared detailing how we conducted our project. [39]

8 Reflection

This project was completed as a Bachelor project for the course titled “COE-C3010 - Computational Engineering Project.” It was completed by the students of Aalto University curretnly completing their Bachelors in Computational Engineering. It was done under the supervision of doctoral researcher Zinan Li. We would like to thank her for her continued assistance in our project and her mentoring through the entire process. We would also like to acknowledge Assistant Professor Junhe Lian for organising this course and offering us the opportunity to partcipate and learn from the experience. We would like to acknowledge Esko Järvinen who was our coordinator for the CSC service and the entire staff of CSC for their assistance. We would also like to thank all the other individuals invovled who helped organise this course.

From this course, we were able to dive deeper into the field of Material Sciences and gain real world experience about how to conduct research and experiment at the academic level. We would encourage anyone interested in material sciences to take this course and encourage the organisers of this course to expand the range and scope of this course even more to encourage more people to participate.

From this project, we were able to sharpen our skills of research and analysis and were able to better understand the methodology involved in preparing publications for the academic field.

For the future we strive to build on the foundation set up by this course in our future studies and we thank everyone invovled for their help in our project.

9 References

[1] Sandra B., Mohammad Z., Patrick H. , Julien M., Hans-Peter G., Thomas A. René H., Machine learning assisted calibration of a ductile fracture locus model, (https://www.sciencedirect.com/science/article/pii/S026412752100157X)

[2] M. Dunand, D. Mohr
Hybrid experimental–numerical analysis of basic ductile fracture experiments for sheet metals

[3] Abbassi, Fethi and Belhadj, Touhami and Mistou, Sebastien and Zghal, Ali
Parameter identification of a mechanical ductile damage using Artificial Neural Networks in sheet metal forming. (2012) Materials & Design, vol. 45 . pp. 605-615.

[4] Freudenthal, A.M., 1950. The Inelastic Behaviour of Engineering Materials and Structures. John Wiley & Sons Inc, New York.

[5] Datsko, J., 1966. Material Properties and Manufacturing Processes. John Wiley & Sons Inc, New York.

[6] Cockcroft, M.G., Latham, D.J., 1968. Ductility and the workability of metals. J. Inst. Met. 96, 33–39.

[7] Rice, J.R., Tracey, D.M., 1969. On the ductile enlargement of voids in triaxial stress fields. J. Mech. Phys. Solids 17 (3), 201–217. https://doi.org/10.1016/0022-5096 (69)90033-7.

[8] McClintock, F.A., 1968. A criterion for ductile fracture by the growth of holes. J. Appl. Mech. 35 (2), 363–371. https://doi.org/10.1115/1.3601204.

[9] Hancock, J.W., Mackenzie, A.C., 1976. On the mechanisms of ductile failure in high-strength steels subjected to multi-axial stress-states. J. Mech. Phys. Solids 24 (2–3), 147–160. https://doi.org/10.1016/0022-5096(76)90024-7.

[10] Bridgman, P.W., 1952. Studies in Large Plastic Flow and Fracture, vol. 115 (Bande).

[11] Baltic Sandra, Magnien Julien, Gänser Hans-Peter, Antretter Thomas, Hammer René, coupled damage variable based on fracture locus: Modelling and calibration, International Journal of Plasticity, Volume 126, 2020, https://www.sciencedirect.com/science/article/pii/S0749641919305406

[12] Gurson, A.L., 1977. Continuum theory of ductile rupture by void nucleation and growth: Part I - yield criteria and flow rules for porous ductile media. J. Eng. Mater. Technol. 99 (1), 2–15. https://doi.org/10.1115/1.3443401 .

[13] Tvergaard, V., Needleman, A., 1984. Analysis of the cup-cone fracture in a round tensile bar. Acta Metall. 32 (1), 157–169. https://doi.org/10.1016/0001-6160(84) 90213-X.

[14] Nahshon, K., Hutchinson, J.W., 2008. Modification of the Gurson model for shear failure. Eur. J. Mech. A Solid. 27 (1), 1–17. https://doi.org/10.1016/j.euromechsol.2007.08.002 .

[15] Tutyshkin, N., Müller, W.H., Wille, R., Zapara, M., 2014. Strain-induced damage of metals under large plastic deformation. Theoretical framework and experiments. Int. J. Plast. 59, 133–151. https://doi.org/10.1016/j.ijplas.2014.03.011 .

[16] Malcher, L., Andrade Pires, F.M., Cesar de Sa, J.M.A., 2014. An extended GTN model for ductile fracture under high and low stress triaxiality. Int. J. Plast. 54, 193–228. https://doi.org/10.1016/j.ijplas.2013.08.015.

[17] Brünig, M., Gerke, S., Schmidt, M., 2017. Damage and failure at negative stress triaxialities: experiments, modeling and numerical simulations. Int. J. Plast. 102 https://doi.org/10.1016/j.ijplas.2017.12.003.

[18] Reddi, D., Areej, V.K., Keralavarma, S.M., 2019. Ductile failure simulations using a multi-surface coupled damage-plasticity model. Int. J. Plast. https://doi.org/10.1016/j.ijplas.2019.02.007.

[19] Lemaitre, J., 1985. A Continous Damage Mechanics Model for Ductile Fracture.

[20] Lemaitre, J., Chaboche, J.-L., 1990. Mechanics of solid materials. Cambridge University Press, Cambridge.

[21] Torki, M.E., 2019. A unified criterion for void growth and coalescence under combined tension and shear. Int. J. Plast. https://doi.org/10.1016/j.ijplas.2019.02.002.

[22] Scales, M., Chen, K., Kyriakides, S., 2019. Material response, localization, and failure of an aluminium alloy under combined shear and tension. Part I experiments. Int. J. Plast. https://doi.org/10.1016/j.ijplas.2019.04.004.

[23] Yao, D., Duan, Y.C., Li, M.Y., Guan, Y.P., 2021. Hybrid identification method of coupled viscoplastic-damage constitutive parameters based on BP neural network and genetic algorithm. Eng. Fract. Mech. 257, 108027 https://doi.org/10.1016/j.engfracmech.2021.108027.

[24] Pütz, F., Shen, F.H., Knemann, M., Münstermann, S., 2020. The differences of damage initiation and accumulation of DP steels: a numerical and experimental analysis. Int. J. Fract. 226, 1–15. https://doi.org/10.1007/s10704-020-00457-z.
[25] Shen, F., Münstermann, S., Lian, J., 2020. Investigation on the ductile fracture of high strength pipeline steels using a partial anisotropic damage mechanics model. Eng. Fract. Mech. 227, 106900 https://doi.org/10.1016/j.engfracmech.2020.106900.
[26] Lian, J., Sharaf, M., Archie, F., Münstermann, S., 2013. A hybrid approach for modelling of plasticity and failure behaviour of advanced high-strength steel sheets. Int. J. Damage Mech. 22, 188–218. https://doi.org/10.1177/1056789512439319.
[27] Zinan Li, Aalto Universtiy, 16 Oct 2022,
https://mycourses.aalto.fi/pluginfile.php/1844169/mod_folder/content/0/2022%20_Topic%205.pdf?forcedownload=1
[28] Dan Yao, Shilong Pu, Muyu Li, Yingping Guan, Yongchuan Duan,
https://www.sciencedirect.com/science/article/abs/pii/S0020768322003031?via%3Dihub
[29] Yuanil Bai, 2008. A new model of metal plasticity and fracture with pressure and Lode dependence.
https://www.sciencedirect.com/science/article/abs/pii/S0749641907001246?via%3Dihub
[30] J.R. Rice, DM. Tracey. On the ductile enlargement of voids in triaxial stress fields. Journal of the Mechanics and Physice of Solids. Volume 17, Issue 3, June 1969, Pages 201-217.
https://www.sciencedirect.com/science/article/abs/pii/0022509669900337
[31] Gordon R.Johnson, Wiliam H. CooK. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. Engineering Fracture Mechanics, Volume 21, Issue 1, 1958, Pages 31-48.

https://www.sciencedirect.com/science/article/abs/pii/0013794485900529?via%3Dihub
[32] Gui Li, Saisai Cui. Meso-mechanics and damage evolution of AA5182-O aluminum alloy sheet Based on the GTN model. Engineering Fracture Mechanics, Volume 235, August 2020, 107162.

https://www.sciencedirect.com/science/article/abs/pii/S0013794420307451?via%3Dihub

[33] Junhe L, Fuhui, S, Xiaoxu J, Deok-Chan A, Dong-Chul C, Sebastian M, Wolfgang B. An evolving non-associated Hill48 plasticity model accounting for anisotropic hardening and r-value evolution and its application to forming limit prediction. Material and Design, Volume 203, May 2021, 109604. https://www.sciencedirect.com/science/article/pii/S0020768317301580

[34] Sandra B, Mohammad Zhian A, Patrick H, Julien M, Hans-Peter G, Thomas A, René H. Machine learning assisted calibration of a ductile fracture locus model. Materials and Design, Volume 203, May 2021, 109604. https://www.sciencedirect.com/science/article/pii/S026412752100157X

[35] CSC – Tieteen Tietotekniikan Keskus (2022) Wikipedia. Wikimedia Foundation. Available at: https://fi.wikipedia.org/wiki/CSC_%E2%80%93_Tieteen_tietotekniikan_keskus .

[36]-Putty (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/PuTTY .

[37] hobs (2021) stackexchange. How to choose the number of hidden layers and nodes in a feedforward neural network? Avaliable https://stats.stackexchange.com/users/15974/hobs

[38] Bastos, N., Prates, P.A. and Andrade-Campos, A. (2022). Material Parameter Identification of Elastoplastic Constitutive Models Using Machine Learning Approaches. *Key Engineering Materials*, 926, pp.2193–2200. doi:10.4028/p-zr575d.

[39] MachineLearningDuctileLocus(2020), Naveed J., Shahinas E., Shreyas G.. Available at https://github.com/JayZAalto/MachineLearningDuctileLocus

[40] S Münstermann, J Lian, F Pütz, M Könemann and V Brinnel Comparative Study on Damage Evolution during Sheet Metal Forming of Steels DP600 and DP1000 Comparative Study on Damage Evolution during Sheet Metal Forming of Steels DP600 and DP1000 - IOPscience

[41] Scott M. Lundberg, Su-In Lee (2017) A Unified Approach to Interpreting Model Predictions https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

10 Apendix