TDDE56 – Foundations of AI and ML
Examiner: Daniel Gnad

Linköping University
Project by: Robert Eriksson, Daniel Leidermark
Solid Mechanics (SOLMEK)

Group Project: Elasto-Plastic Stress and Strain Prediction

The finite element (FE) method is a powerful tool for solving structural problems in mechanical engineering. FE software is frequently used to establish stresses and strains in loaded structures. However, FE analyses can be quite time consuming, especially when plasticity is included in the analysis.

Results from elasto-plastic FE simulations will be used as basis for this project. However, it is not necessary to be particularly familiar with the FE method or plasticity modeling. For those interested, some theory is provided in the appendix.

Aim of the Project We would like to find out if it is possible to train a machine-learning model that, at least to some extent, can replace time-consuming elasto-plastic FE stress and strain analyses. A database of results from several FE analyses is provided. The database contains a selection of a few geometries, load types and mesh sizes; Figure 1 shows an overview. The aim of the project is to figure out if a model can be learned to predict stresses and strains in a loaded structure and to see if the model can generalize in terms of variation in geometry, load distribution, and mesh density.

Problem Description

A two-dimensional plate with a hole is loaded by the total applied force F_0. The stress field around the hole will be higher due to the stress concentration. An elasto-plastic material model is used, which will give plastic deformation around the hole for higher loads. The dimensions of the plate are: W=100 mm, H=200 mm, t=2 mm, R=10 mm, and a=50 mm.

There is a *baseline* case where the total applied force is evenly distributed as a line load, $q = \frac{F_0}{W}$ (in N/mm), see Figure 2. The baseline case has a high mesh density (giving more accurate results) and contains 1000 data sets for loads between 70 N and 70 kN.

First Steps & Research Questions

(1) Familiarize yourself with the provided data and Jupyter notebook. Execute the notebook, understand the code and plots that are generated. The data folder contains an important Readme file. Note that we preprocessed the data, so that it is easier to load.

Observe that there are four different measures to predict. The notebook we provide learns to make predictions for the equivalent (von Mises) stress (the "Seqv" column in the data). In the following steps, train models for at least one additional measure. How do you need to adapt the learning pipeline for that?

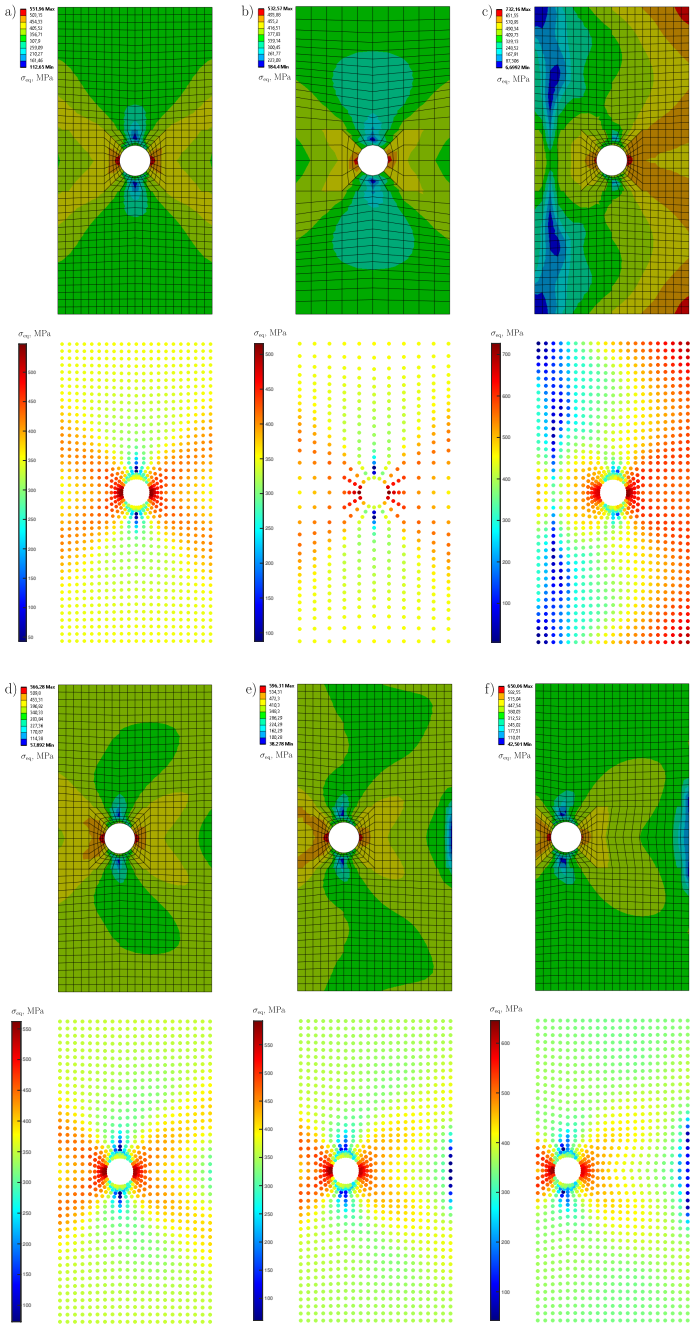

Figure 1: Overview of the FE analyses used to generate data: a) the baseline case (hole at center, uniform load); b) baseline case but with coarser mesh; c) linearly distributed load; d)–f) off-center hole (uniform load). For each subfigure, the top image shows a screenshot from the FE postprocessor and the bottom shows the nodal data available in the database.

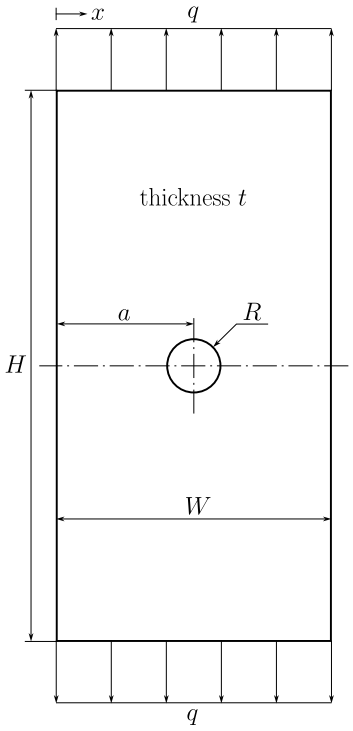

Figure 2: The loaded plate; the baseline case.

(2) Use the baseline data to train a first model and then investigate how accurate its predictions are. Experiment with an arbitrary split of the baseline data into training/test data, and more systematic splitting. Does the accuracy change if you train only on the extreme force values, i.e. minimum, maximum, and some value(s) in between compared to a random split? What does that imply for the generation of training data?

(3) Linearly distributed line load:
For the same geometry as the baseline case, the line load is instead applied as linearly increasing with x (see Figure 2 for definition of x) as

$$q(x) = \frac{2F_0 x}{W^2}$$

Can you train a model with data from uniformly applied line load that predicts stresses and strains for linearly distributed applied line load? What if you train on (a subset of) the linearly distributed load data?

(4) Different mesh size:
Using the same geometry and load as the baseline case, stresses and strains were also calculated for a much coarser mesh size. Since FE results are mesh dependent, the

stresses and strains calculated on a coarse mesh are different from the baseline case. A coarse mesh will speed up the FE analysis but will reduce the accuracy of the results.

Can you train a model that predicts the fine-mesh values (i.e. the results from the baseline case) from data generated from a much coarser mesh?

(5) Variation in hole position:
For the load case of uniformly applied line load, the hole has been moved to three other positions: $a = 40\,\text{mm}$, $a = 30\,\text{mm}$ and $a = 20\,\text{mm}$ (see Figure 2 for definition of a).

Can you train a model with data from the hole at $a = 50\,\text{mm}$ (i.e. at the center) to predict stresses and strains for other positions of the hole? Can the available data be used to learn a model that can predict stresses and strains from an arbitrary position of the hole? Again, play around with the train/test data split. Do you observe any differences?

(6) Be creative:
The Ansys FE model used to generate the database is available (see the Readme file for details), so you can use it to generate your own data if there is something else you would like to see if it can be predicted using an ML model. Use your imagination.

Practical Advice

This section provides advice that might be useful when working with the project. One of the main points is that we recommend to use Google Colab for the project. If you prefer to develop and run the code locally on your machine, then Visual Studio Code is a great alternative.

Google Colab Colab provides an easy-to-use interface, no local installation is required, and it is connected to well-performing hardware using Google's server infrastructure. You can easily import the Jupyter notebooks we are providing in Colab by uploading the *.ipynb file(s) as a "new notebook".

You will probably need to upload some data files to load them in the notebook. We recommend to connect Colab to your Google Drive and upload the files there, so they are easily available. Google provides a good documentation on how to do this here. Essentially, you only need to add a new code block (probably at the top of your notebook, after the includes) that could for example look as follows:

```
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')

data = pd.read_csv("/content/drive/MyDrive/tdde56-data/data.csv")
```

This assumes that you have uploaded the file "data.csv" to the folder "tdde56-data" in your Google Drive. When executing this code cell for the first time, you will be asked to confirm that Colab is allowed to access your Drive.

Python Libraries You will probably need one or more of the following Python libraries when working on your project. While these libraries were already used in the first part of the course, we next provide links to the documentation pages as well as tutorials that help you make better use of them in a larger project. In general, if you have any issues with one of the libraries, it is a good idea to google your problem, e.g. search for an error message that you might get, or search for usage examples of a certain library.

- Numpy: https://numpy.org/, numpy tutorial.
- Pandas: https://pandas.pydata.org/, pandas tutorial.
- Sklearn: https://scikit-learn.org, sklearn tutorial.
- Matplotlib: https://matplotlib.org/, matplotlib tutorial.

A Appendix

This appendix contains some theory regarding plasticity and the finite element method which might be useful.

Stress and strain Total strain, ε, can be split into elastic strain, ε^{e}, and plastic strain, ε^{p}, according to

$$\varepsilon = \varepsilon^{\mathrm{e}} + \varepsilon^{\mathrm{p}}$$

Stress and strain are symmetric tensors. Stress, for example, has the tensor components σ_{x}, σ_{y} and τ_{xy} in 2D. An alternative description of stress is the *equivalent stress* (von Mises stress), σ_{eq}, which reduces the tensor components to a single scalar number. For 2D and plane stress, we get

$$\sigma_{\mathrm{eq}} = \sqrt{\sigma_{\mathrm{x}}^2 + \sigma_{\mathrm{y}}^2 - \sigma_{\mathrm{x}}\sigma_{\mathrm{y}} + 3\tau_{\mathrm{xy}}}$$

In a similar way, we can also define the equivalent strains: $\varepsilon_{\mathrm{eq}}$, $\varepsilon_{\mathrm{eq}}^{\mathrm{e}}$ and $\varepsilon_{\mathrm{eq}}^{\mathrm{p}}$. The database contains values of $\varepsilon_{\mathrm{eq}}$, $\varepsilon_{\mathrm{eq}}^{\mathrm{e}}$, $\varepsilon_{\mathrm{eq}}^{\mathrm{p}}$ and σ_{eq} at each node.

The elasto-plastic model The material used in the FE analyses is modeled as elasto-plastic. The material model was chosen as a nonlinear isotropic hardening power law. This means that the yield limit, σ_{Y}, increases from its initial value, σ_0, with increasing plastic deformation. The rate of hardening is determined by the exponent N. The current yield limit, σ_{Y}, is obtained by solving

$$\frac{\sigma_{\mathrm{Y}}}{\sigma_0} = \left(\frac{\sigma_{\mathrm{Y}}}{\sigma_0} + \frac{3G}{\sigma_0}\bar{\varepsilon}^{\mathrm{p}}\right)^N$$

where G is the shear modulus and $\bar{\varepsilon}^{\mathrm{p}}$ is the accumulated equivalent plastic strain. The values used for this material is: $E = 200\,\mathrm{GPa}$, $\nu = 0.3$, $\sigma_0 = 300\,\mathrm{MPa}$ and $N = 0.15$. The stress–strain curve is shown in Figure 3.

Applying nodal forces The applied total load has been recalculated and applied as nodal forces. The total applied load (in N), F_0, is first recalculated into line load (in N/mm), q. For constant applied load, we get

$$q = \frac{F_0}{W}$$

where W is the width of the plate. For linearly increasing load, we get

$$q(x) = \frac{2F_0 x}{W^2}$$

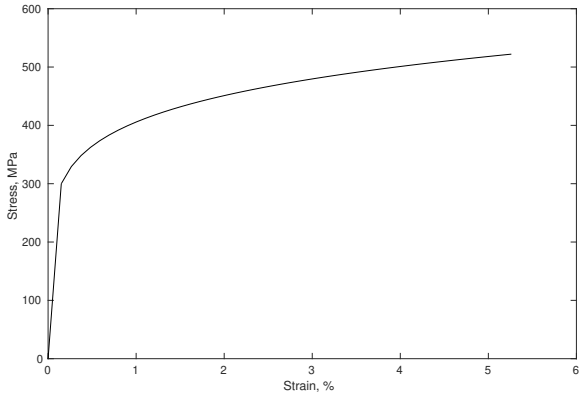

Figure 3: The stress–strain curve of the modeled material.

This is then split into equivalent point loads at each node. At the corner nodes, we must apply a somewhat lower load since the corner nodes only have one "neighbour" that helps carrying the load. For constant load, we get

$$
\begin{aligned}
F_1 &= \frac{q\,(x_2 - x_1)}{2} \\
F_i &= \frac{q\,(x_i - x_{i-1})}{2} + \frac{q\,(x_{i+1} - x_i)}{2}\,,\ i = 2, 3, \ldots, n-1 \\
F_n &= \frac{q\,(x_n - x_{n-1})}{2}
\end{aligned}
$$

and for linearly increasing load, we get

$$
\begin{aligned}
F_1 &= \frac{(2q_1 + q_2)\,(x_2 - x_1)}{6} \\
F_i &= \frac{(q_{i-1} + 2q_i)\,(x_i - x_{i-1})}{6} + \frac{(2q_i + q_{i+1})\,(x_{i+1} - x_i)}{6}\,,\ i = 2, 3, \ldots, n-1 \\
F_n &= \frac{(q_{n-1} + 2q_n)\,(x_n - x_{n-1})}{6}
\end{aligned}
$$

where $F_1,\ F_2,\ \ldots,\ F_n$ are nodal forces and n is the total number of nodes on the edge. Figure 4 illustrates the procedure.

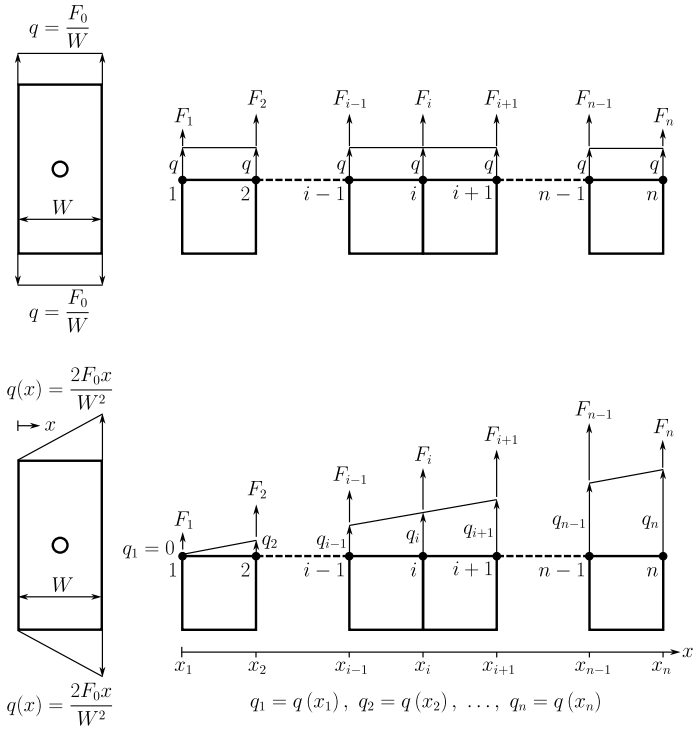

Figure 4: Illustration of how total applied force, F_0, is applied through nodal forces, $F_1,\ F_2,\ \ldots,\ F_n$.