Tutorial Machine Learning in Solid Mechanics (Winter term 2022–2023)

TECHNISCHE UNIVERSITÄT DARMSTADT

Task 2: Hyperelasticity I

Dominik K. Klein, M.Sc. , Yusuf Elbadry, M.Sc. , Prof. Dr. Oliver Weeger

November 15, 2022

In this task you will learn about (i) hyperelasticity, (ii) the generalisation of NNs, and (iii) construction principles for the fulfillment of mathematical conditions with NNs. For this, make use of the code you implemented for "Task 1: Feed-Forward Neural Networks" of this tutorial.

1 Data preparation

In the first step, data for the calibration and testing of the NN models is to be prepared. For the calibration of material models, the corresponding datasets usually consist of stress-strain tuples which are received by physical experiments. However, in the following example data was generated *synthetically*, meaning by evaluating an analytical hyperelastic potential, and the resulting dataset

$$D = \{(\boldsymbol{F}_1, \boldsymbol{P}_1, W_1), (\boldsymbol{F}_2, \boldsymbol{P}_2, W_2), \dots\} \tag{1}$$

consists of triplets for the corresponding deformation gradient $\boldsymbol{F}$, the first Piola-Kirchhoff stress $\boldsymbol{P}(\boldsymbol{F})$ and the strain energy density $W(\boldsymbol{F})$. For the material models to be programmed, $\boldsymbol{F}$ will be the input, while $\boldsymbol{P}$ (and sometimes W) will be the output of the model.

1.1 Data import

Visit the GitHub repository CPShub/TutorialMLinSolidMechanics and go to the folder "02_hyperelasticity_I/data". We now consider the two folders which contain the calibration and test data, respectively. The data is stored in Voigt notation in text files, with the notation stored in a "Readme" file.

Implement a python script which imports $\boldsymbol{F}$, $\boldsymbol{P}$ and W from the text files. Reshape both $\boldsymbol{F}$ and $\boldsymbol{P}$ in such a way that you receive matrices, i.e.

$$\boldsymbol{F} = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} . \tag{2}$$

This data storage will faciliate the following tasks. Then, find a suitable visualization to examine the different load paths of the dataset (load path meaning e.g. uniaxial tension or the mixed test case).

Which physical experiments are represented in the load paths of the calibration dataset? What describes the test cases better: "interpolation" or "extrapolation"?

1.2 Analytical potential

Consider the transversely isotropic hyperelastic potential [4]

$$W(\boldsymbol{F}) = 8\, I_1 + 10\, J^2 - 56 \log(J) + 0.2\left(I_4^2 + I_5^2\right) - 44 \tag{3}$$

with the invariants

$$I_1 = \operatorname{tr} \boldsymbol{C}\,, \quad J = \det \boldsymbol{F}\,, \quad I_4 = \operatorname{tr}\left(\boldsymbol{C}\,\boldsymbol{G}_{\mathrm{ti}}\right)\,, \quad I_5 = \operatorname{tr}\left(\operatorname{Cof} \boldsymbol{C}\,\boldsymbol{G}_{\mathrm{ti}}\right)\,. \tag{4}$$

Here, $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$ denotes the right Cauchy-Green tensor and $\operatorname{Cof}\boldsymbol{C} = I_3\boldsymbol{C}^{-1}$ its cofactor. Furthermore, the transversely isotropic structural tensor is given by

$$\boldsymbol{G}_{\mathrm{ti}} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}\,. \tag{5}$$

At first, implement all of above invariants in TensorFlow. Check your implementation with the data given in the folder "02_hyperelasticity_I/data/invariants", where the values of the invariants for the calibration and test datasets are given. Then, implement the potential from Eq. (3) and check your code with the data imported above. Finally, implement the evaluation of the first Piola-Kirchhoff stress

$$\boldsymbol{P} = \frac{\partial W(\boldsymbol{F})}{\partial \boldsymbol{F}} \tag{6}$$

by using TensorFlows "GradientTape" function, and evaluate it with the data imported above.

2 Neural network model – a naïve approach

2.1 Model implementation

We want to formulate a model which maps the deformation gradient $\boldsymbol{F}$ to the first Piola-Kirchhoff stress $\boldsymbol{P}$. A naïve approach to this task is to use a FFNN which has the six independent components of the right Cauchy-Green tensor $\boldsymbol{C}$ as input and yields the nine components of $\boldsymbol{P}$ as output. Implement this model, which we denote by M^S.

As this model uses the right Cauchy-Green tensor as input quantity, it fulfills the objectivity condition. What other conditions does the model fulfill?

2.2 Model calibration

Calibrate the model to the calibration data imported in Sec. 1.1. Is the model able to interpolate the calibration dataset? Is the model able to predict the unseen load cases of the test dataset? Does the model behave physically sensible?

2.3 Loss weighting strategy

The calibration dataset consists of several different load paths. In general, the stress values $\boldsymbol{P}$ of different load paths may have different orders of magnitude, which affects the contributions of the load paths to the loss function. A load path with smaller stress values will contribute less to the loss function, and thus will be less considered in the parameter optimization process. At worst, the NN won't learn at all from a load path with very small stress values. This can be counteracted by weighting the contributions of different load paths to the loss function: load paths with smaller absolute stress values will be weighted more, while load paths with larger absolute stress values will be weighted less. By this, all different load paths will be equally considered in the parameter calibration process.

Here, we want to use a weighting strategy using (a discrete approximation of) the L^2 norm. For this, implement a function which calculates the norm of each load path according to

$$w = \frac{1}{\#(D)} \sum_j \left\| \boldsymbol{P}^j \right\|\,, \tag{7}$$

where $\|\cdot\|$ denotes the Frobenius norm, $\#(D)$ denotes the number of elements in the load path, the index j goes through all elements in the load path, and $\boldsymbol{P}^j$ denotes a single stress tensor of the load path. Use the inverse of w of each load path as its loss weight in the calibration process. This is implemented by the argument `sample_weight` in the "model.fit(...)" function. Calibrate the model again using the loss weighting strategy, and use the loss weighting strategy for all following model calibrations.

3 Physics-augmented neural network model

3.1 Model implementation

Now, the physics-augmented NN model based on invariants, which we denote by W^I, is to be implemented. The model architecture is as follows:

- The model uses the vector of invariants $\boldsymbol{\mathcal{I}} = (I_1,\, J,\, -J,\, I_4,\, I_5)$ as input for an ICNN, with the invariants as defined in Eq. (4).
- The scalar-valued output of the ICNN is then used as a hyperelastic potential W.
- Differentiating the ICNN w.r.t. the deformation gradient $\boldsymbol{F}$ yields the first Piola-Kirchhoff stress $\boldsymbol{P}$.

Compared to a standard ICNN, the first hidden layer of the ICNN used in this application must be further restricted. How does this restriction look like? Note that this also explains why both J and $-J$ are used in the invariant vector. Hint: Have a look at the polyconvexity condition. Which physical / mathematical conditions are fulfilled by which part of the model architecture?

3.2 Model calibration

Calibrate the model to the calibration data imported in Sec. 1.1.

- Is the model able to interpolate the calibration dataset?
- Is the model able to predict the unseen load cases of the test dataset?
- Examine the stress / energy prediction of the model in the reference configuration $\boldsymbol{F} = \boldsymbol{I}$.
- Calibrate the model (a) using only the energy, (b) using only the stress, and (c) using both. For all three cases, evaluate both stress and energy prediction of the model.
- Use different load paths for the calibration of the model, e.g. only uniaxial tension. Is the model still able to extrapolate to the other cases?

4 Concentric sampled deformation gradients

In the previous tasks we observed an excellent generalization behavior for the physics-augmented model, while the naïve approach generalized very poorly. This does *not* imply that the naïve approach is not able to generalize at all. It only shows that with the considered setting, in particular, the model's hyperparameters and the available calibration data, the model showed a poor generalization behavior.

In Sec. 1 deformation paths such as uniaxial tension were used, which are commonly applied in experimental investigations. For this, only a fairly small amount of deformation gradient paths was considered. Now, a calibration dataset consisting of significantly more deformation gradients is used, sampled in a wider range. As uniform sampling of $\boldsymbol{F}$ in $\mathbb{R}^{3\times3}$ would quickly exceed a reasonable dataset size while possibly containing large deformations outside a relevant range, $\boldsymbol{F}$ is sampled in a physically sensible range using an algorithm proposed by Kunc and Fritzen [3], see [2, Appendix B] for a short introduction.

In "02_hyperelasticity_I/data/concentric" you find the deformation gradients generated with this method. Use the analytical model implemented in Sec. 1.2 to generate the corresponding stress tensors. The dataset again consists of different load paths, which are stored in different text files. Out of all load paths, randomly choose different amounts of load paths for the calibration dataset. Calibrate both models to the chosen calibration dataset and use the remaining load paths as test dataset. Increase the amount of load paths in the calibration dataset until both models are able to predict the test dataset. Repeat this investigations five times in order to circumvent the influence of the random choice of load paths for the calibration dataset.

What differences do you observe in the generalization of the models? What difference do you observe in the required network size / calibration dataset size? What is the smallest possible architecture for each model? What is the *inductive bias*, c.f. Haussler [1], of both NN models?

References

[1] D. Haussler. "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework". In: *Artificial Intelligence* 36.2 (1988), pp. 177–221. DOI: 10.1016/0004-3702(88)90002-1.

[2] D. K. Klein, R. Ortigosa, J. Martínez-Frutos and O. Weeger. "Finite electro-elasticity with physics-augmented neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 400 (2022), p. 115501. arXiv: 2206.05139.

[3] O. Kunc and F. Fritzen. "Finite strain homogenization using a reduced basis and efficient sampling". In: *Mathematical and Computational Applications* 24.2 (2019), p. 56. DOI: 10.3390/mca24020056.

[4] J. Schröder, P. Neff and V. Ebbing. "Anisotropic polyconvex energies on the basis of crystallographic motivated structural tensors". In: *Journal of the Mechanics and Physics of Solids* 56 (2008), pp. 3486–3506. DOI: 10.1016/j.jmps.2008.08.008.