On the use of neural networks in the modeling of yield surfaces

Preprint · June 2023

CITATIONS
0

READS
447

1 author:

Stefan Cristian Soare
Top-Technology
36 PUBLICATIONS **429** CITATIONS

SEE PROFILE

On the use of neural networks in the modeling of yield surfaces

Stefan C. Soare[a,*]

[a]*Technical Univ. of Cluj-Napoca, Memorandumului 28, Cluj-Napoca, 400144, , Romania*

Abstract

We investigate feed forward (densely connected) neural networks as a modeling framework for yield surfaces. We show that homogeneity and smoothness place strong restrictions on the type of activation function. We also show a simple recipe for designing networks with default convex output. It is found that shallow networks in combination with homogeneous activation functions are identical with the FACET yield function. In data-driven contexts such as virtual testing or interpolation from mechanical data, small sized networks, amenable to deployment in finite element codes, can provide a powerful modeling framework capable of handling arbitrary data. Finite element simulations of cylindrical cup drawing show promising results, the overall run-time of small sized shallow networks being comparable with that of traditional yield functions. We also formulate and tackle the problem of mapping the convexity domain of a yield function. This reduces the problem of calibrating yield functions that are not convex by default to a much simpler optimization problem subject to a single constraint. Code and data are made available at: `https://github.com/stefanSCS/nnYS`

Keywords: Neural network, Yield function, Optimization, Convexity, Tension-compression asymmetry

1. Introduction

Neural networks[1] (NN) are universal fitting machines, capable of handling huge amounts of highly heterogeneous data. Their calibration - referred to as *training* in the jargon of data/computer science - is explicitly geared toward *generalization*, i.e., toward making reasonable predictions (interpolation) of unseen data, trying as much as possible to avoid *overfitting*, i.e., an overly accurate fitting of undesirable patterns in data, James et al (2023). In general, overfitting is abhorrent in machine learning, and this is so for good reasons: the input data may be corrupted in various ways[2] and what really matters when making new predictions is the discovery (fitting) of its underlying probabilistic distribution.

On the other hand, in metal plasticity the data is the result of carefully orchestrated and performed experiments, following standardized procedures whenever possible. Randomness does occur, but its sources are traceable and, in principle, quantifiable (e.g., samples may slightly vary in dimensions, heterogeneity, crystallographic texture, processing etc). Furthermore, physically based models, ranging from ab initio calculations to crystal (grain) level continuum plasticity, can then incorporate the experimental data and based on it simulate experiments that may be expensive or even impossible to perform in reality. This is the framework of *virtual testing*, where arbitrary amounts of data can be generated and the goal is to find models that reproduce it as accurately as possible, e.g., Kraska et al (2009), Esmaeili et al (2019), Roters et al (2019). Thus the paradigm here is opposite to that of machine learning: *Overfitting is desirable*. To further clarify, any realization/outcome of an experiment is statistically significant for the estimation of the process window of a manufacturing process, e.g., Harsch et al (2016), Pereira et al (2021). For example, in order to be able to obtain good estimates of the height of deep-drawn cylindrical cups via numerical simulation, the underlying constitutive model must fit as accurately as possible the mechanical properties of the blank-sheet. In turn, these are obtained by mechanical testing and the result of each test will feature a certain spread/variance. In a study of the variance of the cup-height any individual outcome of a mechanical test becomes relevant, and various combinations of mechanical tests are fitted in order to evaluate their influence on the deep drawn cup via simulation. Most often, however, mechanical data are reported in the form of averages, thus removing any fluctuations/noise. In this case the statement 'overfitting is desirable' becomes transparent, since, by a well known result in statistics, the expected value of a random variable is its best estimator in a least squares sense (and hence must be fitted as accurately as possible).

In this study (Sections 3-6) we investigate the applicability of feed forward (densely connected) NNs to the modeling of the yield surfaces of metals under quite general conditions of anisotropy and load reversal (a)symmetry. The focus is on feasibility, practicality, *deployment* - The actual usage of the model in practice, i.e., in finite element (FE) simulations; And lastly, but not the least, on reproducibil-

*stefancsoare@gmail.com

[1]The literature on this topic is vast. We can only mention a few defining references: Strang (2019) and Calin (2020) for an introduction to its mathematics; Géron (2022) and Chollet (2021) for an overview of the machine learning landscape and software.

[2]The famous MNIST data set of handwritten digits is perhaps the archetypal example. One does not want a badly written '5' be systematically interpreted as '6'.

ity. This is important, since our framework is essentially *deterministic* while randomness plays a significant role in the traditional approach to the training of NNs[3].

Some of the recent contributions to the problem take a holistic approach and train NN-models of the yield function by fitting solutions of the evolution equations of classical plasticity, Vlasis and Sun (2021), Fuhg et al (2023). While this is the most general framework, the resulting NN-models are not homogeneous (thus requiring a mapping of large domains in stress space) and some of them feature deviations from convexity. Nevertheless, the cited works achieve quite good results with networks comprised of hundreds of nodes (with a rough count of parameters in the thousands). A more pragmatic approach, adopted here, is to directly model a yield surface level[4]. This was recently attempted by Nascimento et al (2023). However, besides not being homogeneous, the reported NN is comprised of roughly 6000 nodes, the resulting model thus employing over seven *million* parameters. Such networks are appropriate for digesting heterogeneous data with high dimensional feature spaces but are a clear overkill when applied to the modeling of convex bodies of a certain regularity. Furthermore, large networks are unlikely to be effective in FE-simulations.

In what follow we conduct a rational analysis of the main ingredients of a densely connected NN in the context of modeling yield surfaces and the first point we make is about the data. Densely connected NNs are very amorphous modeling mediums - clay like - and therefore require large amounts of data for proper calibration. Their applicability then is best suited to environments where data sets of arbitrary size can be generated. These are usually virtual testing environments, as discussed above, but one may also employ data generation tools based on the interpolation of mechanical tests or of any discrete combination of data, e.g. Vegter and Boogaard (2006), Soare (2023). Here the interest is in the *expressiveness* of a NN-model, i.e., in its modeling power, and hence it is sufficient for our purposes to substitute crystal plasticity or interpolation based generated data with samples extracted from conventional yield surface models.

A second point is about *validation*. This is a crucial step for any machine learning project where data is expected to be noisy and/or fluctuate randomly (with unknown probabilities). The available data is then split into two sub-sets, one reserved for training, the input data, and the other for validation, the test data[5]. For our problem, the targets consist of regular convex surfaces where random localized variations are not expected. Thus we fit all the available data. Then the validation step in our case will consists of an assessment of the model's predictions of directional properties (yield stresses and r-values), yield surface levels, convexity and symmetry.

The second problem considered here (Section 7) is that of the calibration of yield functions that are not convex by default. These functions are usually polynomial, e.g. Hill (1950)/Ch.12, Soare et al (2008), or trigonometric/harmonic expansions, e.g. Raemy et al (2017), Soare (2023). One approach to their calibration is via constraint optimization, Soare (2023) and Soare and Diehl (2023), where a set of linear constraints is used to construct a local approximation of the convexity domain of the yield function - the set of parameters for which it is convex. This process of approximation is analogous to that of approaching a circle by a sequence of approximating polygons. Important classes of metals, e.g. aluminun alloys, are very close to the boundary of such convexity domains and hence accurate modeling of these materials requires an accurate description of this boundary: the number of constraints is then inherently large. Here we consider the problem at its most fundamental level: *To map the convexity domain of a yield function that is convex only for a subset of its parameters*. In its general form, as proposed here, the problem appears to be intractable within the framework of the classical approximation theory, which is most suitable for low dimensional spaces. Instead, we shall approach it as a classification problem, where parameters associated with convex/non-convex yield surfaces are labeled with '0'/'1', and use a feed forward densely connected NN to model the boundary between these two subsets.

2. Densely connected NNs

This is the simplest network architecture where the modeling function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is defined as a sequence of functional compositions of identical computational units - the neurons. Specifically, for an input $\boldsymbol{x} \in \mathbb{R}^n$, the first layer of the network consists of C_1 computational units defined for $k = 1, ..., C_1$ by:

$$a_{1|k} = \Phi(z_{1|k}),\ z_{1|k} = b_{1|k} + \sum_{j=1}^{n} W_{1|kj}\, x_j \tag{1}$$

The first index points to the layer, here the first layer, and is separated from the rest of the indices by a vertical bar. Any number of layers can then be stacked on top of the first layer. The L-th layer would then consist of C_L neurons, defined for $k = 1, ..., C_L$ by

$$a_{L|k} = \Phi(z_{L|k}),\ z_{L|k} = b_{L|k} + \sum_{j=1}^{C_{L-1}} W_{L|kj}\, a_{L-1|j} \tag{2}$$

[3]Hyperparameters, such as the learning rate, are often dependent on the training loops (*epochs*), which in turn employ optimization engines that rely on random selection of data batches and stochastic gradient descent algorithms.

[4]Here we discuss *regression* models. The modeling of yield surfaces can also be regarded as a *classification* problem. This is the approach taken by Hartmaier (2020) who employed a support vector representation of the boundary of the elastic domain.

[5]Our terminology, rooted in mechanical engineering parlance, is slightly different from that employed in ML. The latter refers to a small subset of the input data, used for the tuning of hyper-parameters, as validation data. Here validation and test (unseen) data are one and the same.

Finally, the (scalar) output of the network is defined by

$$f(\boldsymbol{x}) = b_{\Lambda|1} + \sum_{j=1}^{C_{\Lambda-1}} W_{\Lambda|j}\, a_{\Lambda-1|j} \tag{3}$$

Here Λ is the index of the final (output) layer and hence $L = 1, ..., \Lambda$. One says the network is Λ-layers deep. The first $\Lambda - 1$ layers, between input and output, are called *hidden* layers. The network is called *dense* because any computational unit receives input from all units in the previous layer. The b's are referred to as *biases* and the W's as *weights*. Collectively they form the *trainable* parameters of the network.

In our case, $\boldsymbol{x}$ stands for $\boldsymbol{\sigma}$, the Cauchy stress, or for $\boldsymbol{\sigma'} = \boldsymbol{\sigma} - (\mathrm{tr}(\boldsymbol{\sigma})/3)\boldsymbol{I}$ the Cauchy stress deviator. The function $\Phi : \mathbb{R} \longrightarrow \mathbb{R}$ is the *activation* function of a neuron. A vast amount of empirical evidence now favors the ReLU (rectified linear unit) function:

$$\Phi(z) = \max(0, z) \tag{4}$$

Let us note, however, that ReLU is not differentiable at $z = 0$. One may argue that in numerical implementations a perfect zero is rarely (if ever) encountered. Nevertheless, piece-wise linearity leads to abrupt variations in surface gradients, a feature which may be irrelevant in the classification/regression problems usually tackled by machine learning but which is crucial to metal plasticity where return mapping algorithms rely essentially on yield surface gradients.

3. Two fundamental assumptions: Homogeneity and smoothness

In phenomenological plasticity, the yield surface of a metal/alloy is defined as the boundary of the elastic domain, Hill (1950). It is assumed that the yield function describing this boundary is first order positive homogeneous with respect to some reference stress. The latter may be regarded as a 'center' of the elastic domain. For most applications, including the present work, this reference stress can be safely taken at the origin of the stress space.

Besides mathematical convenience, homogeneity may be justified based on the linear elastic response featured by any metal/alloy. Irrespective of its previous processing history, the material will respond elastically along a proportional loading path up to a certain path dependent limit - the yielding stress. Thus the elastic domain is necessarily a *star* domain, call it $\mathcal{E}$. To $\mathcal{E}$ then there corresponds a unique homogeneous function f that has the boundary $\partial\mathcal{E}$ as its level set $f(\partial\mathcal{E}) = 1$.

ReLU is homogeneous but not smooth. The requirements of homogeneity and smoothness[6] lead us naturally to investigate activation functions defined by

$$\Phi_{AN}(z) = z^{N-1}\max(0, z) = \begin{cases} z^N, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases} \tag{5}$$

The above will be employed in contexts where tension-compression asymmetry is present. For centro-symmetric plastic properties we shall use

$$\Phi_{SN} = |z|^N \tag{6}$$

with $N \geq 2$ an integer number[7]. Homogeneity also demands that the biases of the network vanish: $b_{L|k} = 0$, for $L = 1, ..., \Lambda$, $k = 1, ...C_L$. The output of a dense NN activated by Φ_{AN} or Φ_{SN} is then a (piecewise) homogeneous polynomial function of degree N^Λ.

When $\Phi = \Phi_{SN}$, the whole network becomes a parametrization of a subset of the space of homogeneous polynomials of degree N^Λ, and of the space of piecewise polynomials if $\Phi = \Phi_{AN}$. The similarity with previous such parametrizations is apparent: the linear transformation approach to generating yield functions also leads to polynomial expressions, Soare and Barlat (2010), Soare and Diehl (2023). To further illustrate this point, it is instructive to look at the particular case of a shallow network, having only one hidden layer. The yield function defined by the network, via eqs.(1) and (3), is then simply

$$f(\boldsymbol{\sigma'}) = \sum_{j=1}^{J} W_j\, \Phi\left(\boldsymbol{w}^{(j)} \cdot \boldsymbol{\sigma'}\right) \tag{7}$$

When Φ is Φ_{SN} in eq.(6), the centro-symmetric FACET yield function proposed by Van Houtte et al (2008) in their eq.(50) is recovered. When Φ is Φ_{AN} in eq.(5), the asymmetric FACET proposed in eq.(53) of the cited reference is recovered. If all $W_j \geq 0$, the formula in eq.(7) defines a *convex* function, since it is the result of the summation of a sequence of convex terms (each term being the composition of a convex real function - the activation - with a linear transformation). The total number of nodes is $J = C_1$ and the weights of the j-th node have been assembled into a vector/tensor $\boldsymbol{w}^{(j)}$. Note that because of homogeneity the weights of the output layer are redundant (they can be multiplied into the rest of the weights without decreasing the dimension of the parameter space) and hence they could all be set to one in eq.(3), irrespective of the number of layers. The modeling power is then in direct relation with the width J of the hidden layer: more nodes lead to more parameters being combined into the monomial coefficients. This obviously works only up to a saturation limit, defined by the degree of the activation function (the number of independent parameters being limited by the number of monomials in the final polynomial expression).

[6] By *smoothness* we understand C^2 smooth functions. We should note that metal plasticity also employs 'multisurface' yield functions featuring potential corners. These locations of non-smoothness are usually motivated on physical grounds (e.g., crystal plasticity), in contrast with the type of random jumps associated with non-smooth activation functions.

[7] We use integers to facilitate comparison with polynomial modeling frameworks. However, N can be any real number ≥ 2.

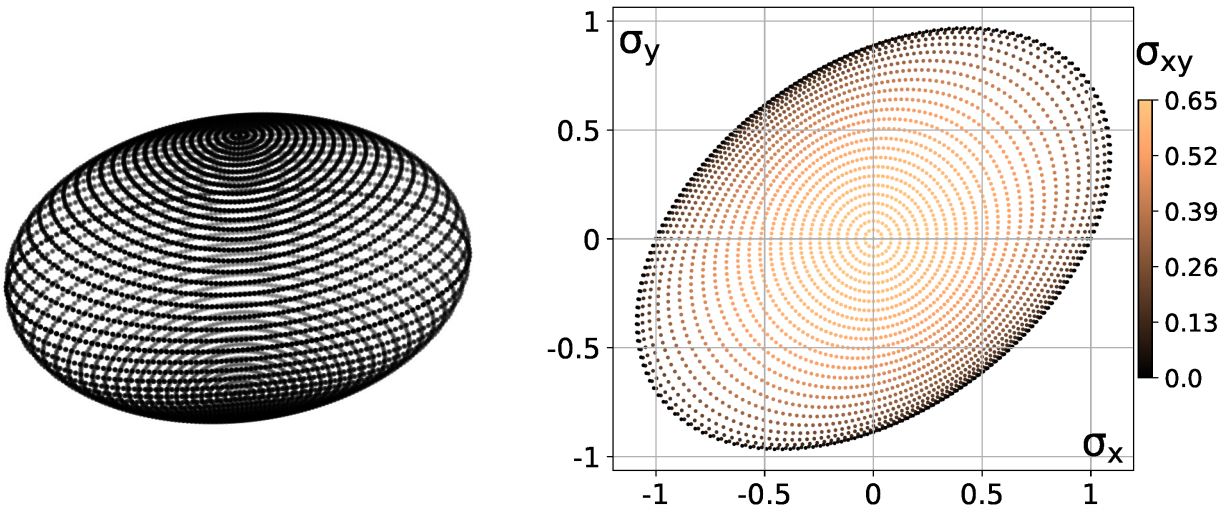

Figure 1: **Left:** A cloud of 6512 samples extracted from a Hill'48 model of AA6022-T4 (Left) and its top view with corresponding numerical map (Right).

4. Methodology and a first example: Fitting Hill'48

To be worthy of consideration for practical applications, any newly proposed yield function must be proven capable of at least approximating, if not reproducing, the quadratic of Hill (1948). In this section we expose the methodology employed for implementing and testing NNs, using Hill's function as a toy example.

As noted in the Introduction, we use analytical yield surface models as substitute for data. More precisely, we use (first order positive homogeneous) plane stress yield functions $f^{\mathcal{T}}$ (with $\mathcal{T}$ standing for *target*) in the context of isotropic hardening where all level sets are scaled versions of the surface defined by $f^{\mathcal{T}}(\boldsymbol{\sigma}) = 1$ and the modeling space is the 3d vector space $\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \sigma_{xy}]$. Then we extract a number of N_σ samples from the surface $f^{\mathcal{T}}(\boldsymbol{\sigma}) = 1$ as follow:

1) First, we generate a pseudo-uniform distribution of N_σ points $\boldsymbol{\tau}^{(k)}$, $k = 1, ..., N_\sigma$, on the unit sphere of the stress space by using the algorithm in Soare (2023).

2) Then the actual data set (cloud) of stress points is defined by $\boldsymbol{\sigma}^{(k)} = \rho^{(k)}\boldsymbol{\tau}^{(k)}$, with $\rho^{(k)} = 1/f^{\mathcal{T}}\left(\boldsymbol{\tau}^{(k)}\right)$.

In the simplest scenario, we then fit the NN model f to the point cloud, i.e., we calibrate its weights by minimizing the Euclidean distance between predictions and targets:

$$\sum_{k=1}^{N_\sigma}\left[f\left(\boldsymbol{\sigma}^{(k)}\right) - 1\right]^2 \tag{8}$$

Historically, metal plasticity has favored models requiring only few data points/experiments[8]. On the other hand, it is well known that the performance of NNs depends crucially on data: The more, the better. This is also true when fitting a yield surface model to a point cloud: The cloud must be dense enough to allow for a good rendering of the surface gradients as well. Thus we have no qualms in using data sets consisting of thousands of samples. This places no restrictions on the practicality of the method: Once properly calibrated, a virtual testing framework can generate arbitrary amounts of data; The same is true in frameworks where data is generated based on interpolation tools from mechanical tests. An additional reason for such a dense sampling will become apparent in Appendix-B, where some of the NN-models generated here will be compared in finite element simulations with their source model.

As a concrete example, we use a Hill'48 model of AA6022-T4 generated based on the three r-values $r_0 = 0.8$, $r_{45} = 0.37$ and $r_{90} = 0.54$, experimental data from Tian et al (2017). The samples extracted from this model are shown in Fig.1.

We shall show next several NN-models of this dataset. But first, some information about the computing environment is in order. We used `TensorFlow` as implementation medium, a popular open source `python` package released by Google. All randomness was made reproducible by using constant seeds. We used the gradient descent algorithm `RMSprop` as optimizer, with the learning rate =1.0e-4 (and default values for the rest of its parameters). The typical number of epochs was about 500 (with variations, depending on the model) with a batch-size of 16 data points. These parameters allowed for relatively stable descents. To each model there correspond two notebooks[9], one showing the

[8]For various reasons: For analytical simplicity; Experiments are expensive and/or difficult to conduct; After calibration, models are typically deployed in finite element simulations where efficiency and performance are essential.

[9]Jupyter notebooks are versatile interactive interfaces to `python` kernels that can be run in most browsers. There are several options for running such notebooks, perhaps the easiest being Google's Colab. The NNs employed here are tiny by comparison with the average data science project, so they can easily run on desktops. On Linux platforms one can install the entire software chain. On Windows one can use Docker.

fit of the model and the other its validation. All notebooks (files bearing the extension `.ipynb`) can be accessed at the GitHub repository specified at the end of this article.

4.1. The perils of ReLU

Our first NN-model has only one hidden layer consisting of 600 neurons, each activated by ReLU (notebooks `nb00_*`; see also Appendix-A/Table-3). The calibrated/fitted NN-model is shown in Fig.2. At a first look, it appears to be quite a satisfactory model: the NN-model almost overlaps the analytical model.

As validation of any NN-model reported here we employ the following three criteria:

V.1 Prediction of generalized directional properties $\overline{\sigma}(q,\theta)$ and $r(q,\theta)$, for the drawing zone corresponding to, say, $q \in [-0.2, 0]$ (with the stress ratio q defined in Appendix-A).

V.2 Orthotropic symmetry

V.3 Convexity

The predictions of the directional stresses of ReLU based NN-model are also good, Fig.3/Left, as one would expect from the near overlap in Fig.2. However, the predicted r-values, although good on average, feature an undesirable saw-tooth profile, Fig.3/Right. This is caused by the sharp variations in gradient direction caused by the constant slopes of ReLU. To alleviate this, Vlasis and Sun (2021) used multiplication layers in their ReLU based networks. On the other hand, Nascimento et al (2023) used a similar, raw architecture, as employed here, albeit with 6000 neurons spread over four hidden layers. Adding more layers does not solve the saw-tooth profile problem, as shown in notebooks `nb00B_fitHill` and `nb00B_Valid`.

The NN-models defined by eqs.(1)-(3) are not orthotropic by default. Orthotropy is a theoretical assumption, based on the geometry of the manufacturing process of sheet metal (rolling), meant to simplify the constitutive models based on analytical formulas. If, instead of using directly the network output f, we use the symmetrized

$$f_O(\sigma_x,\sigma_y,\sigma_{xy}) = \frac{1}{2}\Big[f(\sigma_x,\sigma_y,\sigma_{xy}) + f(\sigma_x,\sigma_y,-\sigma_{xy})\Big] \quad (9)$$

then the yield function f_O is automatically orthotropic (see also footnote-10). We will not use this approach here, since, from a modeling perspective, it is far more interesting to investigate the fitting of a monoclinic model to orthotropic data. Thus, the orthotropic symmetry is dictated here by the data (This is similar to obtaining transverse isotropic models from traditional orthotropic yield functions, e.g., Hill'48, Yld2004-18p, PolyN, etc, when the input data is 'isotropic', i.e., uniform uniaxial properties, etc.). As numerical measure we employ the absolute difference along the z-coordinate of stress points on the NN-model surface. Thus, for any two such points, having the same x and y-coordinates, i.e., $\boldsymbol{\sigma}^{(1)} = \left[\sigma_x,\sigma_y,\sigma_{xy}^{(1)}\right]$ and $\boldsymbol{\sigma}^{(2)} = \left[\sigma_x,\sigma_y,\sigma_{xy}^{(2)}\right]$:

$$|\Delta\sigma_{xy}| = \left|\sigma_{xy}^{(1)} - \sigma_{xy}^{(2)}\right| \quad (10)$$

The verification of the orthotropic symmetry of the ReLU based NN-model is shown in Fig.2/Right. For aluminum alloys, where the reference yield surface $f(\boldsymbol{\sigma}) = 1$ is scaled by factors in the range of 300-500MPa, a maximum $|\Delta\sigma_{xy}| < 0.005$ is acceptable (bounding the top-bottom asymmetry to being less than 2.5MPa). For the ReLU model we have $\max|\Delta\sigma_{xy}| \le 0.0028$, and hence it passes the symmetry test.

Our third validation criterion is convexity. This is perhaps the closest in intent to validation as employed more broadly in machine learning: In our problem, violation of convexity usually points to lack of data. Usually, yield functions are C^2-smooth and hence their convexity can be verified by probing their Hesiann or the Gauss curvature of their level sets at the locations of a dense pseudo-uniform grid on the unit sphere. In the case of a ReLU based NN the Hessian is zero (almost) everywhere and hence to check convexity we have to employ directly the definition. Specifically, for each location $\boldsymbol{\tau}$ of a pseudo-uniform grid on the unit sphere we construct three unit directions $\boldsymbol{u}^{(i)}$, $i = 1,2,3$, in the tangent plane of the unit sphere at $\boldsymbol{\tau}$ such that

$$\boldsymbol{u}^{(1)} + \boldsymbol{u}^{(2)} + \boldsymbol{u}^{(3)} = \boldsymbol{0}$$

From homogeneity, the stress state in the direction of $\boldsymbol{\tau}$ on the yield surface $f(\boldsymbol{\sigma}) = 1$ is $\boldsymbol{\sigma} = \boldsymbol{\tau}/f(\boldsymbol{\tau})$. Similarly, for the (not necessarily unit) directions

$$\boldsymbol{\tau}^{(i)} = \boldsymbol{\sigma} + \mu\,\boldsymbol{u}^{(i)},\ i = 1,2,3$$

the corresponding stress states on the yield surface are $\boldsymbol{\sigma}^{(i)} = \boldsymbol{\tau}^{(i)}/f(\boldsymbol{\tau}^{(i)})$. The function f is convex if and only if it satisfies the Jensen inequality

$$f\left(\boldsymbol{\sigma}^{(M)}\right) \le \frac{1}{3}\sum_{i=1}^{3} f\left(\boldsymbol{\sigma}^{(i)}\right) = 1 \quad (11)$$

where $\boldsymbol{\sigma}^{(M)} = \left(\boldsymbol{\sigma}^{(1)} + \boldsymbol{\sigma}^{(2)} + \boldsymbol{\sigma}^{(3)}\right)/3$. When μ is small, $\boldsymbol{\sigma}^{(M)}$ is very close to the grid stress $\boldsymbol{\sigma}$ and hence the above inequality becomes an effective test for local convexity.

Here we used $\mu = 0.0025$ and calculated $f(\boldsymbol{\sigma}^{(M)})$ on a pseudo-uniform grid of 9352 points. The results for the ReLU model are shown in Fig.4/Left. The color bar shows the minimum and maximum limits of $f(\boldsymbol{\sigma}^{(M)})$. The notable feature is that virtually all points are dark red, a value corresponding to 1, i.e., the inequality in eq.(11) is an equality at these points. In other words, the ReLU model is a piecewise linear approximation of the target model, which correlates well with the saw-tooth profile of the r-values in Fig.3 and with the symmetry map in Fig.2/Right. By comparison, the same plot for the analytical model from which the target data was extracted shows a very different pattern, Fig.4/Right: It has smooth transitions and it never reaches the linearity threshold ($\max f(\boldsymbol{\sigma}^{(M)}) < 1$).

4.2. A network with four neurons

In the previous subsection we used only one hidden layer. Adding more layers does not improve upon the final model,

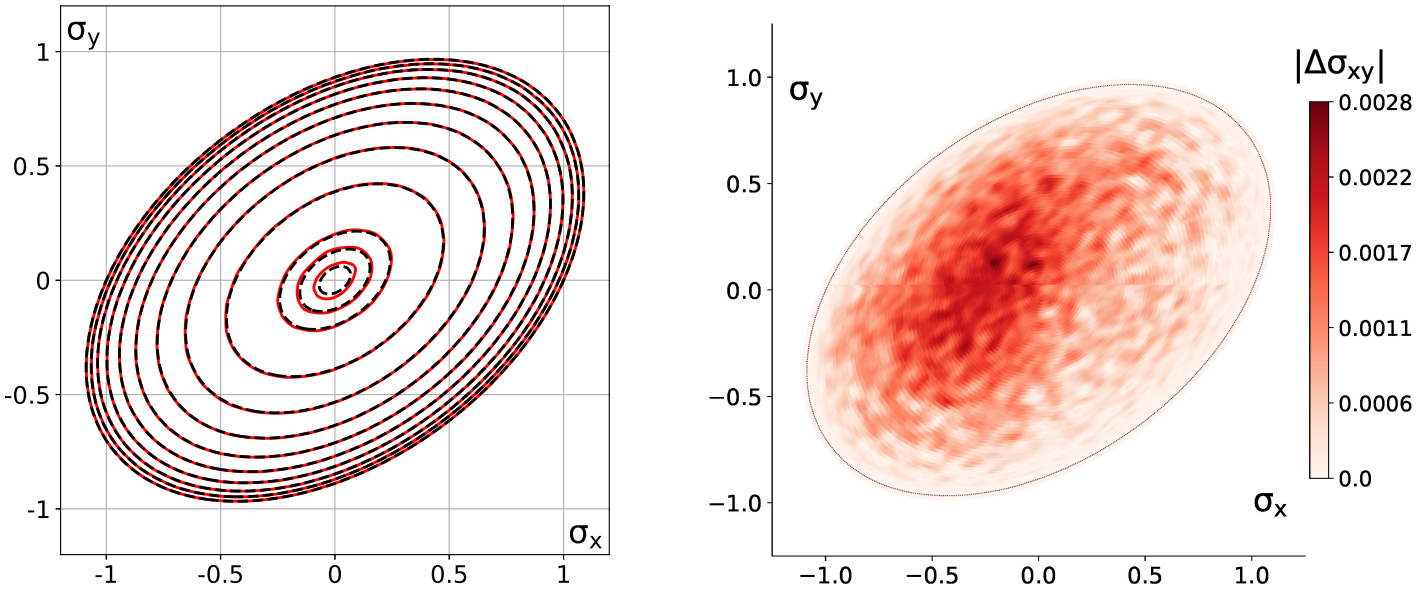

Figure 2: **(nb00)** Left: σ_{xy} = *constant* sections of a ReLU-based NN-model (red) of Hill'48 data (black). Right: Verification of the orthotropic symmetry of the NN-model.

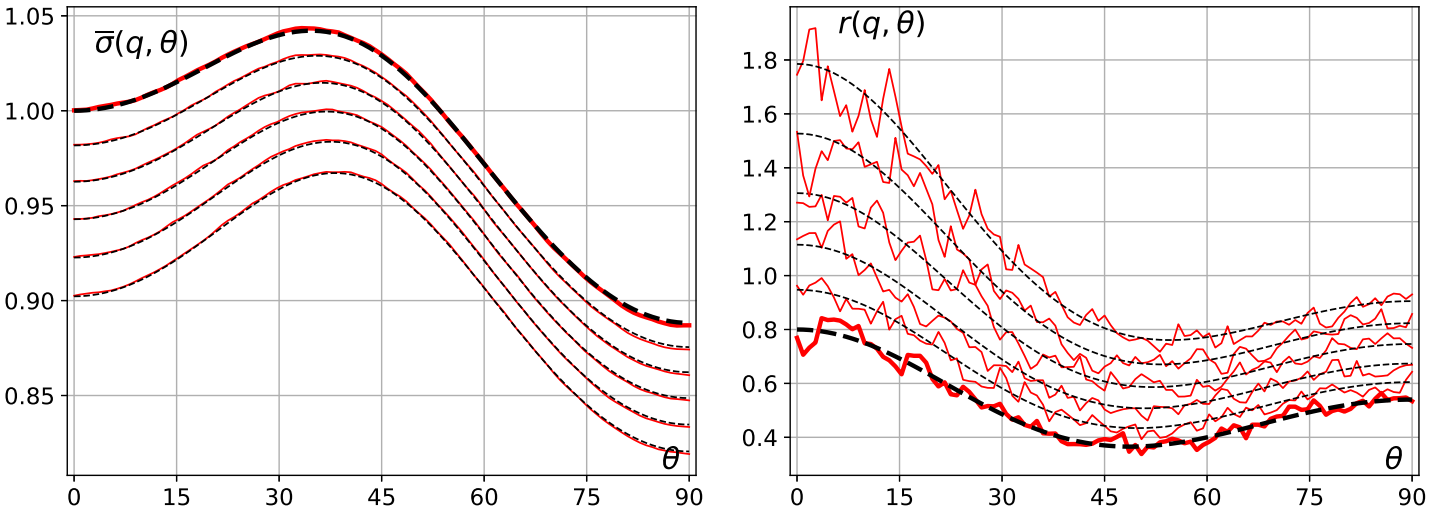

Figure 3: **(nb00)** Predictions of the ReLU-based NN-model (red) of Hill'48 data (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

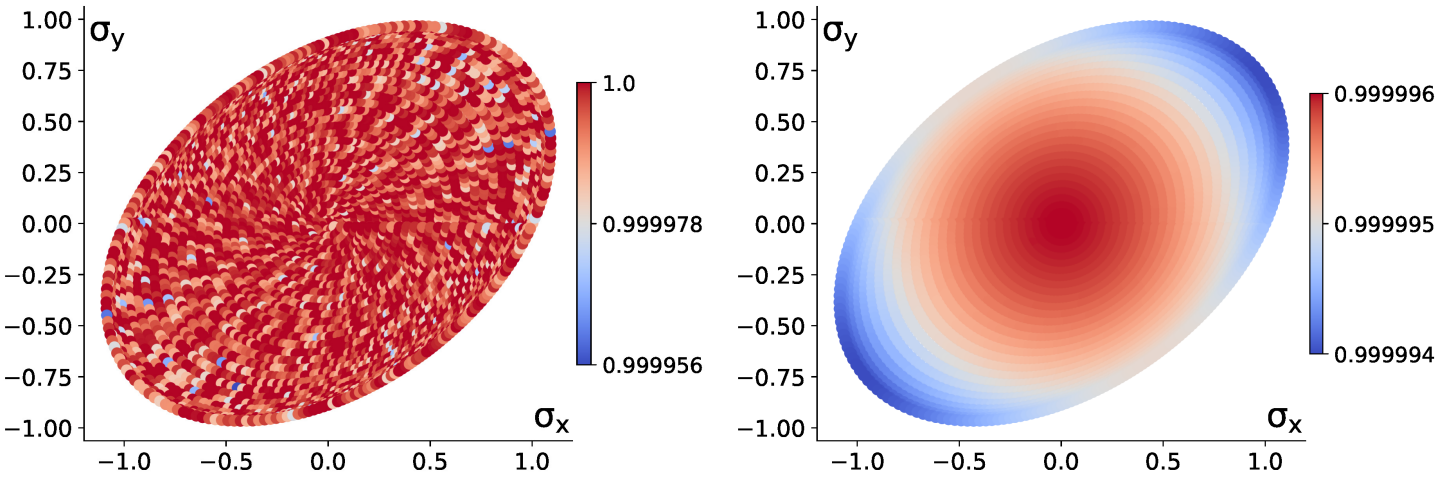

Figure 4: **(nb00)** Verification of convexity via eq.(11): of the ReLU-based NN-model of Hill'48 data (Left) and of Hill'48 analytical model (Right).

as shown by notebooks `nb00B_*`, Appendix-A/Table-3 and Figs-1,2,3 in Appendix-C: the saw-tooth profile is actually amplified. A solution to this problem is to enrich the feature space by considering input data of the form

$$\boldsymbol{x} = [\sigma_x^2, \sigma_x\sigma_y, \sigma_y^2, \sigma_{xy}^2] \tag{12}$$

In combination with ReLU, this not only solves the smooth-

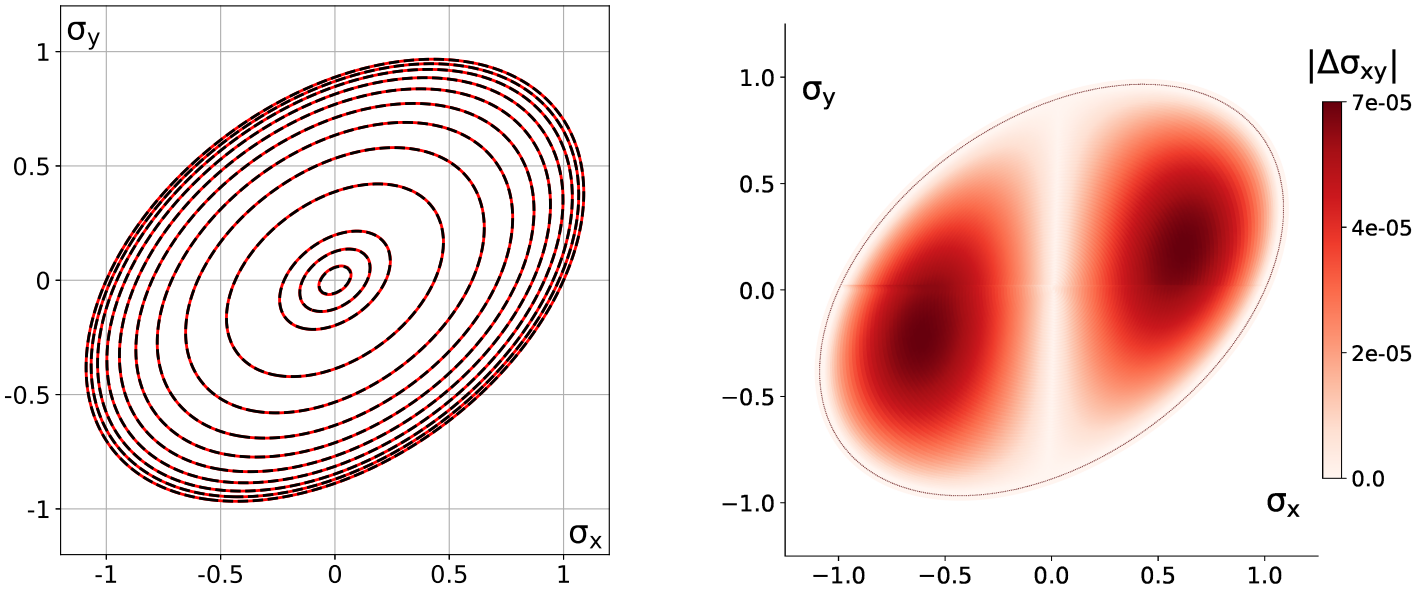

Figure 5: **(nb01)** Left: σ_{xy} = *constant* sections of a Φ_{S2}-based NN-model (red) of Hill'48 data (black). Right: Verification of the orthotropic symmetry of the NN-model.

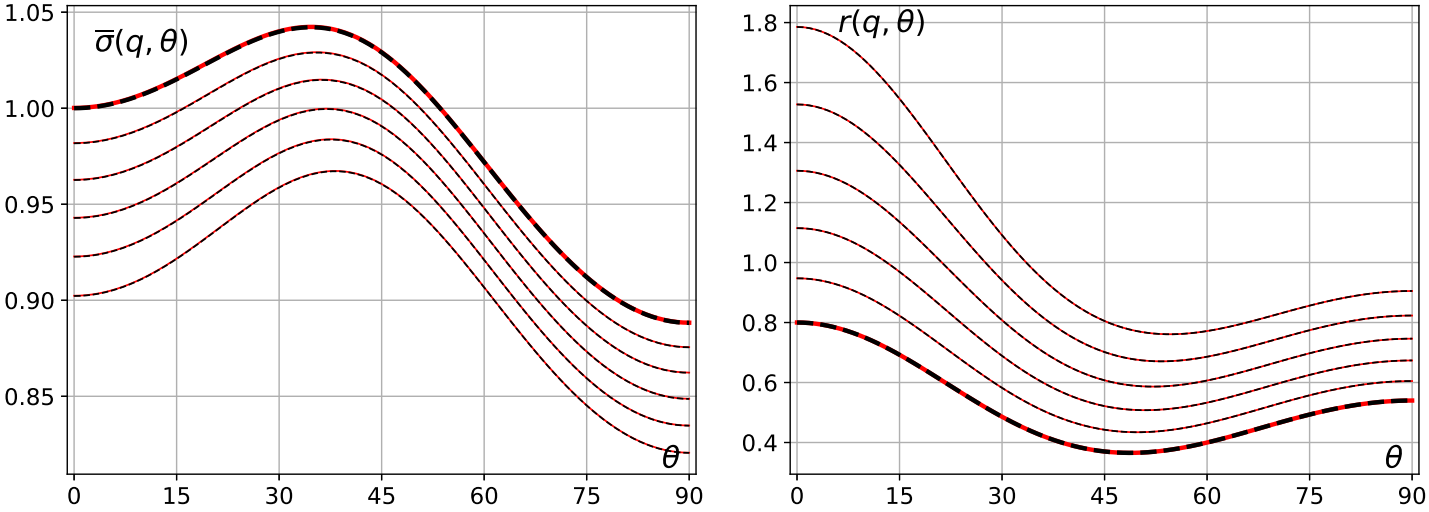

Figure 6: **(nb01)** Predictions of the Φ_{S2}-based NN-model (red) of Hill'48 data (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

ness problem, but also automatically ensures centro and orthotropic symmetries[10]. More generally, one can consider the entire set of monomials of degree N, or of harmonic monomials of arbitrary degrees. Then one effectively recovers PolyN, Soare and Diehl (2023), or SHYqp, Soare (2023). We do not pursue this approach here since the algorithms proposed in the cited references are much more efficient for these models.

As discussed in Section-4, one can solve the smoothness problem by employing the power or piecewise power functions in eqs.(6) and (5). Then parameter saturation can be achieved with smaller networks. For example, by using only one hidden layer with only four neurons activated by the quadratic Φ_{S2} one can easily replicate Hill'48, as shown in Figs.5 and 6 (notebooks `nb01_*`).

As shown by Fig.5/Right, the model is orthotropic, the symmetry gap being ≤ 0.00007. Since here the model associated with f is the quadratic

$$f(\boldsymbol{\sigma}) = a_1\sigma_x^2 + a_2\sigma_x\sigma_y + a_3\sigma_y^2 + a_4\sigma_{xy}^2 + a_5\sigma_x\sigma_{xy} + a_6\sigma_y\sigma_{xy}$$

it is instructive to correlate Fig.5/Right with the magnitude of the above coefficients. In terms of weights, the networks reads (we set the weights of the output layer equal to 1), eq.(7):

$$f = \sum_{j=1}^{4} \left(\boldsymbol{w}^{(j)} \cdot \boldsymbol{\sigma}\right)^2$$

After training we obtain the optimal vectors $\boldsymbol{w}^{(j)}$ and identifying the above two quadratics obtains: $a_1 = 0.999704$, $a_2 = -0.888635$, $a_3 = 1.267243$, $a_4 = 2.398172$, $a_5 = -0.000375$, and $a_6 = 0.000004$. Then $a_5 \approx 0$ and $a_6 \approx 0$

[10] One can also employ a vector of invariants of the stress tensor as input, as done, for example, in Klein et al (2021), Fuhg et al (2022) and Kalina et al (2023) to model the strain energy of a hyperelastic material. However, in order to enforce convexity via the methodology discussed in the next section, one has to consider *subsets* of invariants that are convex functions, which may impact the modeling range.

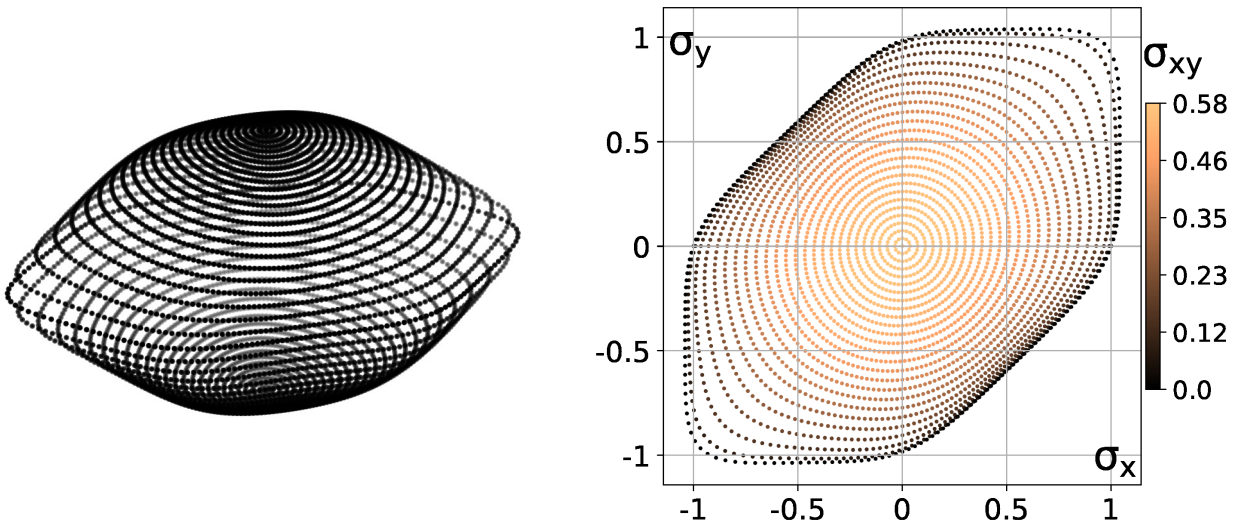

Figure 7: A cloud of 6512 samples extracted from a Poly8 model of AA6016-T4 (Left) and its top view with corresponding numerical map (Right).

and hence we recover the classical four parameter plane stress Hill'48.

`TensorFlow` provides a rich API for automatic differentiation. We made use of it to calculate the gradient and the hessian of all smooth NN-models featured here. These then allow for an easy calculation of the Gaussian curvature K_G via the formula (see Appendix-D for a proof and an overview of the relation between curvature and convexity)

$$K_G(\boldsymbol{\sigma}) = \frac{1}{|\nabla f|^{m+2}} (\boldsymbol{H}^* : \nabla f) \cdot \nabla f \tag{13}$$

where $\boldsymbol{H}^*$ is the cofactor matrix of the hessian, and $m = 2$ for plane stress and $m = 4$ for triaxial stress. The stress $\boldsymbol{\sigma}$ is on the the level set $\Gamma = \{\boldsymbol{\sigma} : f(\boldsymbol{\sigma}) = 1\}$, which is (strictly) convex if for all of its points there holds $K_G > 0$. Upon verification of the K_G featured by the four neurons model on a dense grid on the level set Γ, we obtained $K_G \geq 0.963 > 0$, and hence the model is convex also, thus passing all validation criteria V.1-V.3.

Finally, a note about the input data is in order. Once a suitable network architecture has been identified one may then proceed to a more structured inquiry about the necessary input. The most basic question is whether the size of the training data set can be reduced. The four neurons network is specialized to fitting quadratics and hence the answer is yes: notebooks `nb02_*`, Appendix-A/Table-3, show the results of training the same network on a cloud of only 72 points (see also Figs-4,5,6 in Appendix-C).

5. A second example: Fitting AA6016-T4

As a more realistic example we next consider the problem of modeling the plastic properties of the aluminum alloy AA6016-T4 with experimental data reported in Habraken et al (2022). As data generator we use the Poly8 model of this alloy reported in Soare and Diehl (2023) from which the point cloud shown in Fig.7 was extracted.

We experimented with two types of NN-architectures to fit this data set, each architecture being largely determined by the activation function employed. The first type,

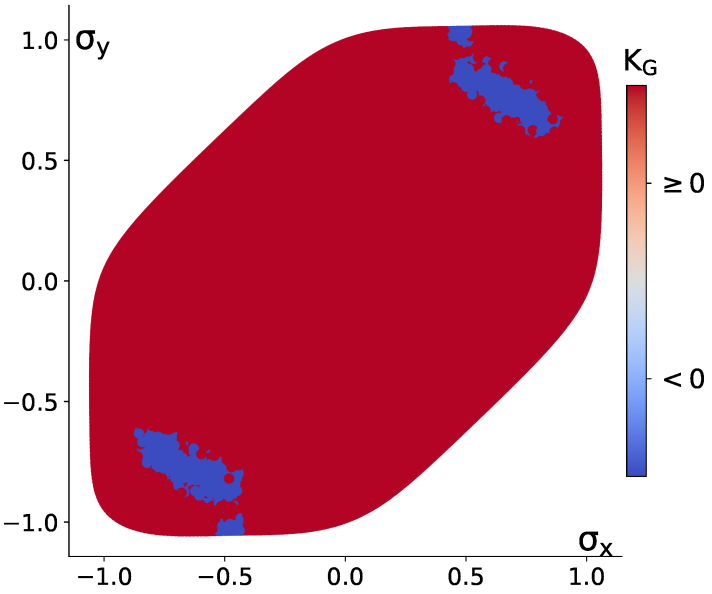

Figure 8: **(nb10)** Verification of the convexity of the NN-model (K_G is the Gauss curvature of the yield surface).

described by the rows `nb07`, `nb07B`, `nb08` and `nb10` in Appendix-A/Table.3, consists of several hidden layers of equal width, with nodes activated by Φ_{S2}. The output of these networks is a polynomial of order 8, i.e., Poly8, for the first three, and of order 16, i.e., Poly16, for the last. One can see from the corresponding rows in Appendix-A/Table-3, that the first three models provide quite good approximations of the original Poly8 model. However, all four NN-models fail the convexity test, the model in `nb07` being the closest to the original model and, with $K_G \geq -0.0008$, featuring the least deviation from convexity (see Figs-7 to 17 in Appendix-C for the results in notebooks `nb07`, `nb07B`, `nb08` and `nb10`).

To better illustrate the issue, we show here the model with the largest deviation from convexity, i.e., `nb10` (See also Section 6.3 of Appendix-D, where this model is reanalyzed using the `Autograd` library). Fig.8 depicts the zones where the deviation from convexity happens. The same pattern is observed for the models in `nb07`, `nb07B` and `nb08` as

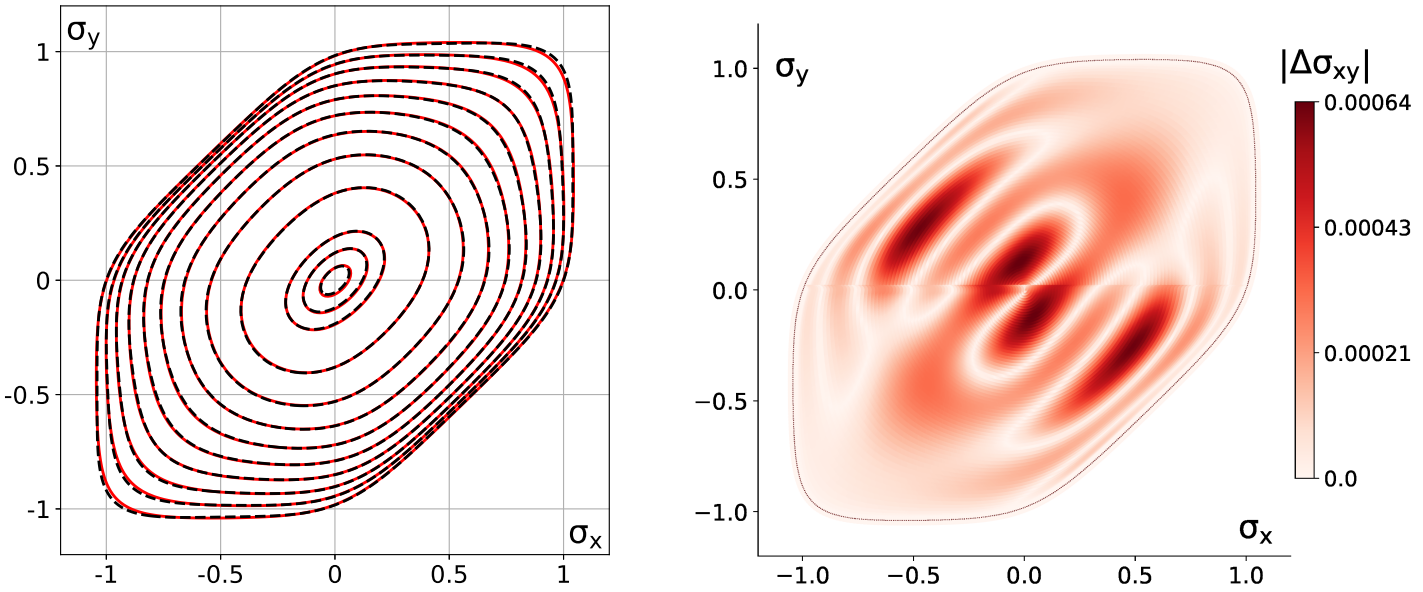

Figure 9: **(nb14)** Left: σ_{xy} = *constant* sections of the NN (red) and Poly8 (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

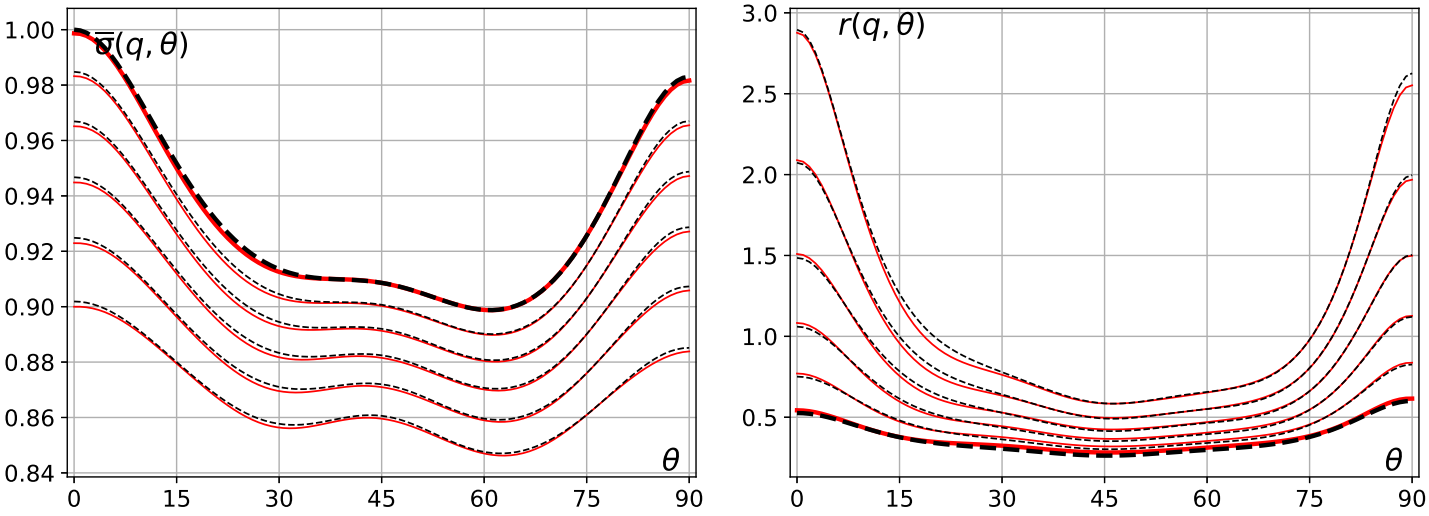

Figure 10: **(nb14)** Predictions of the NN-model (red) and Poly8 (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

well: The deviation from convexity is manifested in the areas where data is least dense, i.e., near the balanced-biaxial points, Fig.7/Right. Adding more data will probably solve the problem but this is not a procedure guaranteed to work reliably, each and every time, for any kind of data.

The best solution is to work with NN-models that are by default convex with respect to their input layer. One approach was discussed by Fuhg et al (2023). Here we take a much simpler approach in the framework of feed forward densely connected NNs. The first example is the model `nb13` described in Appendix-A/Table-3. It consists of three layers: The first is activated by Φ_{S8}; the second layer is activated by Φ_{A2} and is constrained to have non-negative weights ($W_{2|kj} \geq 0$); finally, the output also is subject to the same constraints ($W_{3|j} \geq 0$). The analytical expression of the output of this network is

$$f(\boldsymbol{\sigma}) = \sum_{j=1}^{C_3} W_{3|j} \Phi_{A2} \left(\sum_{k=1}^{C_2} W_{2|jk} \Phi_{S8}(\boldsymbol{w}^{(k)} \cdot \boldsymbol{\sigma}) \right) \qquad (14)$$

where, to shorten the notation we denoted $\boldsymbol{w}^{(k)} = [W_{1|kp}]$. Then, by a well-known result, Theorem 5.1 in Rockafellar (1970), the output is convex. From Appendix-A/Table-3 and Figs-18,19 in Appendix-C, one can see that this model provides an approximation similar to that of the previous NN-models but with the added benefit of default convexity.

The second example employs an even simpler architecture described in row `nb14` of Appendix-A/Table-3: One hidden layer, activated by Φ_{S10}, the weights of the output layer being constants (all set =1). The resulting analytical expression of the output is of the kind defined by eq.(7), and hence is convex. Fitting this model on the data leads to the results in Figs.9 and 10.

6. A third example: Fitting AZ31B

As a final example we show the modeling of a material featuring strength differential effect, i.e., tension-compression asymmetry. This type of asymmetry is largest for metals with hexagonal close packed (HCP) lattices, such as Ti and

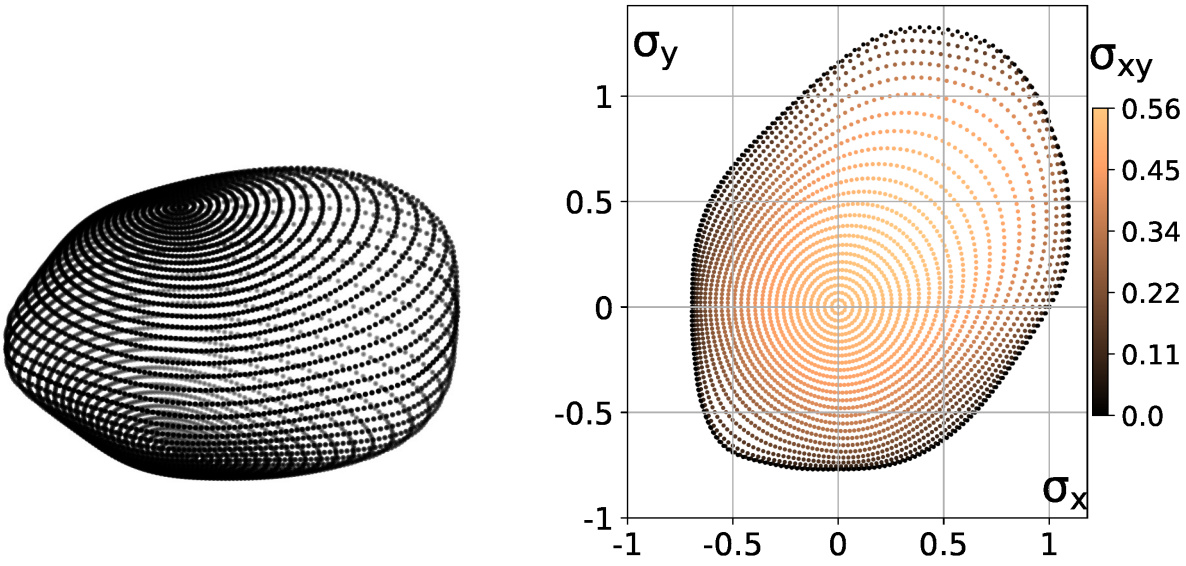

Figure 11: A cloud of 6512 samples extracted from a SHYqp model of AZ31B (Left) and its top view with corresponding numerical map (Right).

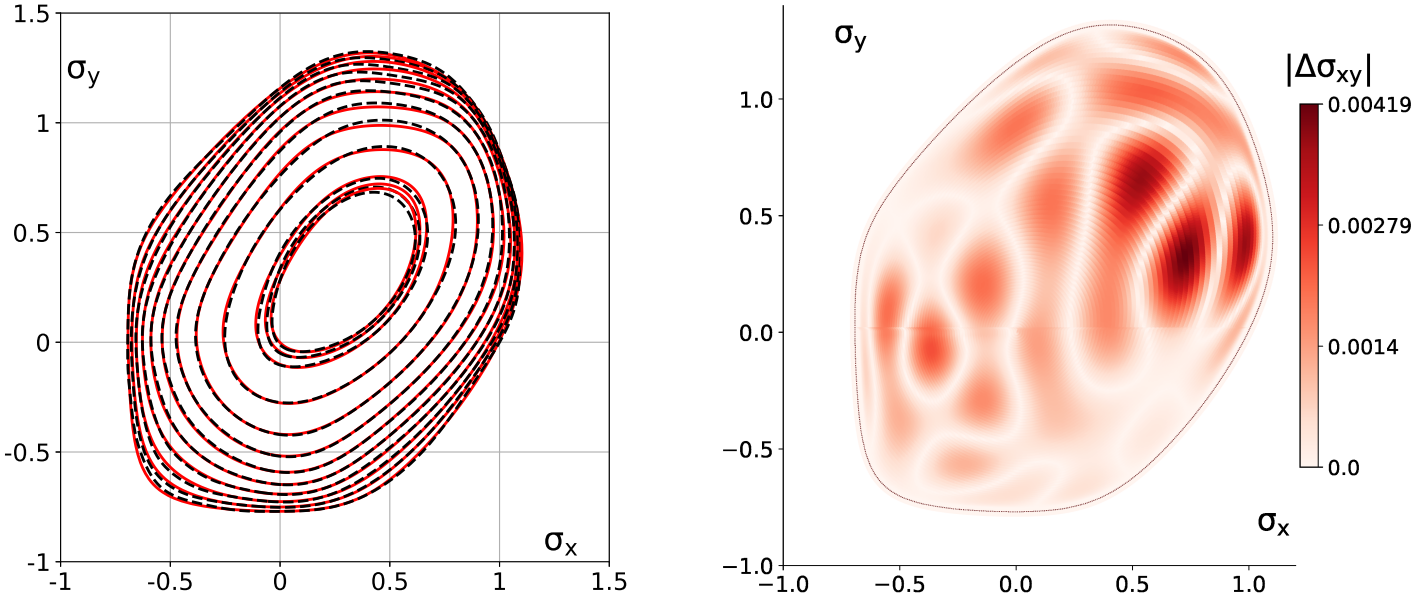

Figure 12: **(nb15)** Left: σ_{xy} = *constant* sections of the NN (red) and SHYqp (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

Mg-based alloys. As specific material we use the magnesium alloy AZ31B and a SHYqp model of its yielding properties reported in Soare (2023) (model-C/Fig.22 of cited ref), which was built using the experimental data reported in Lou et al (2007).

The samples extracted from the SHYqp model are shown in Fig.11. For a NN-model to capture the asymmetry with respect to the surface center (here the origin of the stress space) at least its first layer must be activated by 'asymmetric' functions, i.e., of the type in eq.(5). The structure of the upper layers is then somewhat arbitrary and only extensive experimentation can detect advantages of some architectures over others. In general, as discussed in the previous section, if the output of the NN-model is to be convex with respect to its input, then the activation functions of all layers must be convex and monotonic, i.e., again of the type in eq.(5), and the weights of all layers $L \geq 2$ must be non-negative.

We used the simplest architecture to model the data for this material: One hidden layer with 60 neurons activated by Φ_{AN} in eq.(5) with $N = 10$. The output is then exactly described by eq.(7), i.e., this network is an instance of the asymmetric FACET, and hence its output is a convex function. We performed three tests with this network. In the first, we trained the network on the data in Fig.11 for 450 epochs. The results of this test (notebooks `nb15_*`) are shown in Figs.12-14. Overall, the NN-model passes most validation tests with one exception. The NN predicted r-values deviate significantly from their targets. In the second test, notebooks `nb16_*` in Appendix-A/Table-3, we augmented the data set by extracting 9352 samples from the target model SHYqp. We then used the weights obtained in the first test as initial guess and retrained the model on the augmented data for another 400 epochs. The results, Figs-20 and 21 in Appendix-C, do not show a significant improvement over the first test, the r-values featuring the

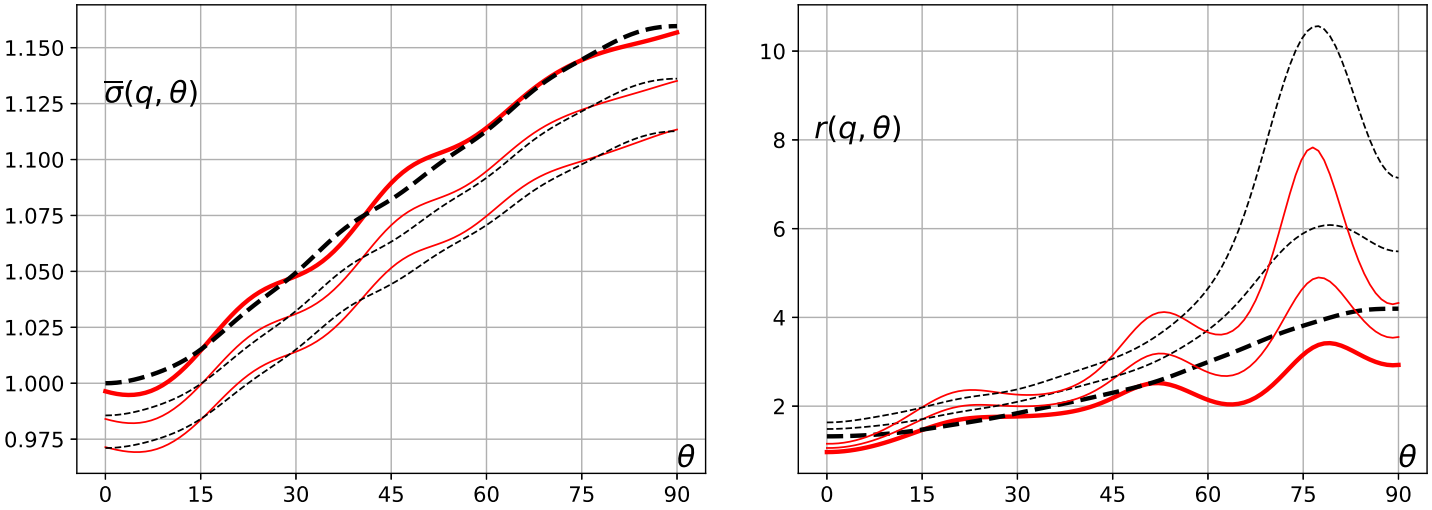

Figure 13: **(nb15)** Predictions of the NN-model (red) and SHYqp (black) for tensile tests: Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

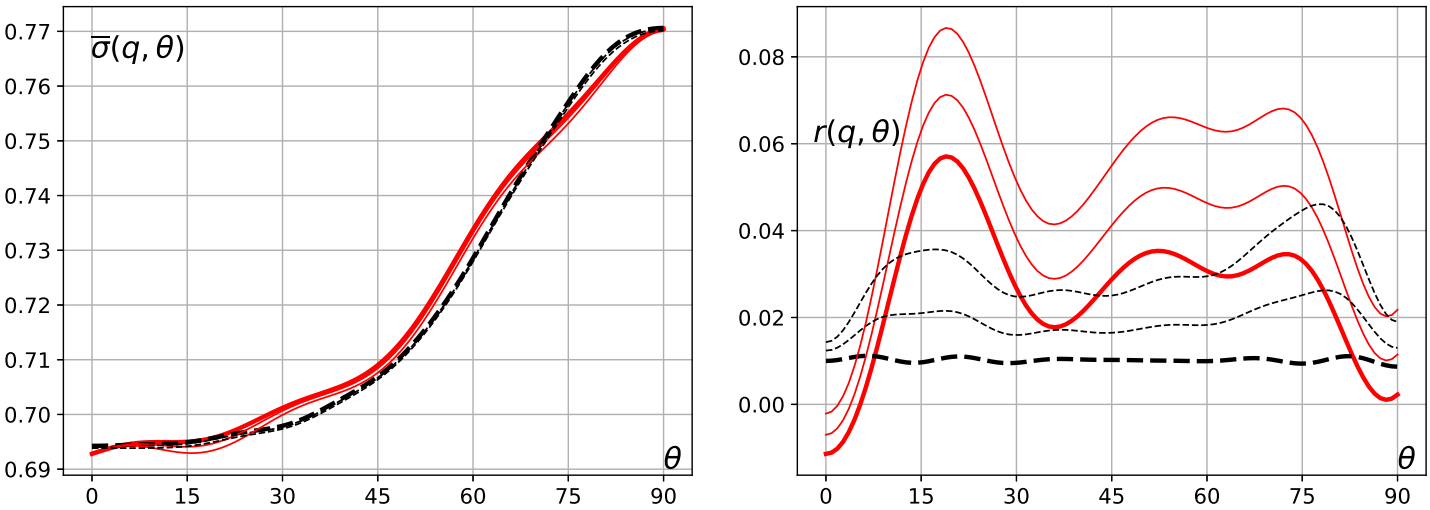

Figure 14: **(nb15)** Predictions of the NN-model (red) and SHYqp (black) for compression tests: Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

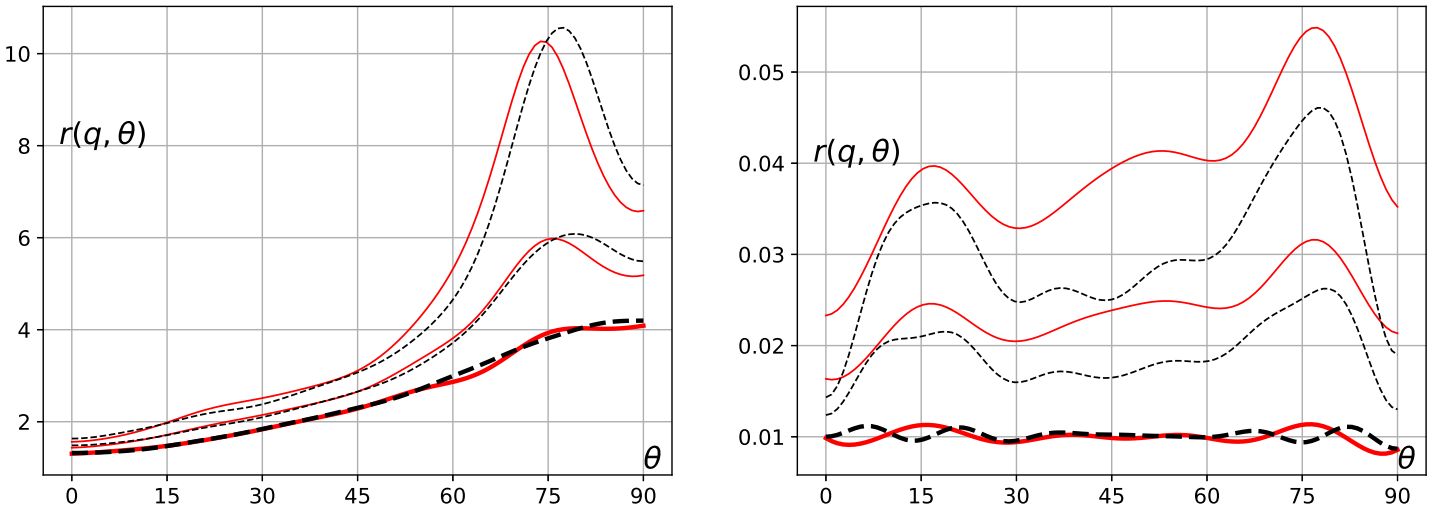

Figure 15: **(nb17)** r-values predictions of the NN-model (red) and SHYqp (black) for tensile (Left) and compression (Right) tests. Thick lines correspond to uniaxial ($q = 0$).

same oscillatory pattern as in Figs.13/Right, Fig.14/Right.

Much better results can be achieved by incorporating into the loss function information about the field of exterior normals of the yield surface by using generalized r-values, Soare and Diehl (2023). In this way, in conjunction with networks having smooth and convex output, the size of the input data set can be drastically reduced. We show here the benefits of this approach by regularizing the loss function

with the uniaxial r-values only. Thus we augment the loss function with the following term:

$$\begin{aligned}&\Delta_r = \\ &\lambda \sum_\theta \left[a_\theta \frac{\partial f}{\partial \sigma_{xy}}\left(\boldsymbol{u}^t_\theta\right) - b^t_\theta \frac{\partial f}{\partial \sigma_x}\left(\boldsymbol{u}^t_\theta\right) - c^t_\theta \frac{\partial f}{\partial \sigma_y}\left(\boldsymbol{u}^t_\theta\right)\right]^2 \\ &+\lambda \sum_\theta \left[a_\theta \frac{\partial f}{\partial \sigma_{xy}}\left(\boldsymbol{u}^c_\theta\right) - b^c_\theta \frac{\partial f}{\partial \sigma_x}\left(\boldsymbol{u}^c_\theta\right) - c^c_\theta \frac{\partial f}{\partial \sigma_y}\left(\boldsymbol{u}^c_\theta\right)\right]^2\end{aligned} \tag{15}$$

Here $\lambda > 0$ is a scaling factor and $\theta \in \{k\pi/18 \,|\, k = 0, 1, ..., 9\}$. The superscript '$t$' indicates a uniaxial tension test on samples cut parallel with the (Ox, Oy)-plane along stress directions at an angle θ from the Ox-axis (which usually is the rolling direction for sheet metal) defined by

$$\boldsymbol{u}^t_\theta = [\cos^2\theta, \sin^2\theta, \cos\theta\sin\theta]$$

while the superscript 'c' indicates a uniaxial compression test along directions $\boldsymbol{u}^c_\theta = -\boldsymbol{u}^t_\theta$. The coefficient functions are defined by

$$a_\theta = \sin\theta\cos\theta,\ b^\bullet_\theta = r^\bullet_\theta + \sin^2\theta,\ c^\bullet_\theta = r^\bullet_\theta + \cos^2\theta$$

where a '$\bullet$' stands for 't' or 'c', and where r^t_θ and r^c_θ are the target r-values in tension and compression, respectively (here calculated based on the SHYqp model).

In the third test, in notebooks `nb17_*`, we used the weights of the first NN-model (`nb15`) as initial guess and retrained the model for another 400 epochs on the original data set in Fig.11 by using the regularized loss function in eq.(15) with $\lambda = 0.0025$. Overall, the results are similar with those of the first two tests, Appendix-A/Table-3, but show a significant improvement in the fit of r-values, Fig.15 (and Figs-22,23 in Appendix-C).

7. Mapping the convexity domain of a yield function

A long standing problem in metal plasticity is the calibration of yield functions that are not convex by default. The difficulty lies in the lack of tractable descriptions of the convexity domain - the set of parameters for which the function is convex - and, in general, the problem is solved either by trial and error or, more systematically, by constraint optimization. Here, we tackle directly the essence of the problem, which we state in the form: Given a yield function $f = f(\boldsymbol{\sigma}, \boldsymbol{a})$, to find the subset of parameters $\boldsymbol{a} \in V$ for which f is convex. More precisely, we denote by

$$\mathcal{D}_C = \{\boldsymbol{a} \,|\, \boldsymbol{\sigma} \longrightarrow f(\boldsymbol{\sigma}, \boldsymbol{a}) \text{ is convex}\} \tag{16}$$

Here $\boldsymbol{a}$ stands for any set of parameters, e.g. a vector, a tensor, or a list of combinations of the two, and V for the underlying parameter space. In order for f to be an effective yield function, its convexity domain must feature some minimal regularity properties: $\mathcal{D}_C$ must be at least simply connected and its boundary $\partial\mathcal{D}_C$ piecewise C^1-smooth.

Then our problem consists in mapping the boundary of $\mathcal{D}_C$. One such mapping can be realized by constructing a function $\boldsymbol{a} \longrightarrow g(\boldsymbol{a}) \in \mathbb{R}$, over the parameter space, with the property that

$$\mathcal{D}_C = \{\boldsymbol{a} \,|\, g(\boldsymbol{a}) \le 0\} \tag{17}$$

We seek smooth functions g for then $g(\partial\mathcal{D}_C) = 0$. The calibration problem of the yield function f is then reduced to an optimization problem with a *single* constraint:

$$\begin{cases} \min \Phi_f(\boldsymbol{a}) \\ \text{subject to } g(\boldsymbol{a}) \le 0 \end{cases} \tag{18}$$

where Φ_f stands for a cost function measuring some distance between the predictions of f on some dataset and its targets.

Assuming a homogeneous yield function f, its level sets are uniquely determined by its 'normalized' yield surface, i.e., for a fixed set of parameters $\boldsymbol{a}$, the set $\mathcal{Y}_f(\boldsymbol{a}) = \{\boldsymbol{\sigma} \,|\, f(\boldsymbol{\sigma}, \boldsymbol{a}) = 1\}$. Hence $f(\cdot, \boldsymbol{a})$ is convex if $\mathcal{Y}_f(\boldsymbol{a})$ is a convex set. Denoting by $K_G(\boldsymbol{a}, \boldsymbol{\sigma})$ the curvature field of $\mathcal{Y}_f(\boldsymbol{a})$, the g function can be defined as:

$$g(\boldsymbol{a}) = -\min_{\boldsymbol{\sigma} \in \mathcal{Y}_f(\boldsymbol{a})} K_G(\boldsymbol{a}, \boldsymbol{\sigma}) \tag{19}$$

The goal is to find a good enough approximate of the above in a tractable functional form. For high dimensional spaces, such as the parameter space of a yield function, this is in general a difficult problem in the context of classical theories of approximation where the central paradigm is a Hilbert functional space generated by orthogonal basis functions, e.g. Courant and Hilbert (1953), Atkinson and Han (2012). Such basis functions in high dimensional spaces are computationally expensive and sensitive to the accumulation of roundoff errors. Neural networks, on the other hand, seem particularly well suited in this case, due to their fitting capabilities and their relatively simple computational schemes. They do require, however, solving several specific challenges: Generating the data set; Finding a proper network architecture and fitting the dataset; Solving efficiently the optimization problem in eq.(18). We take a first step here and consider the simplest case, that of the plane stress orthotropic fourth order homogeneous polynomial - Poly4:

$$f(\boldsymbol{\sigma}) = \sigma_x^4 + a_1\sigma_x^3\sigma_y + ... + a_4\sigma_y^4 + ... + a_8\sigma_{xy}^4 \tag{20}$$

As with any PolyN, the first coefficient is always $a_0 = 1$, ensuring by default normalization of the yield function along the rolling direction. The parameter space is then $V = \mathbb{R}^8$. The resulting problem is small enough to be handled with widely available computational hardware and yet, as shown next, it already features most of the challenges posed by the general problem.

7.1. Generating the data

We use the fact that $\mathcal{D}_C$ is a convex set, Soare and Barlat (2010), and hence for any interior point $M \in \mathcal{D}_C$ any half-line passing through it will intersect $\partial\mathcal{D}_C$ in exactly one

point. We choose M to be the PolyN replica of the von Mises function. In case of Poly4 this is

$$M = [-2, 3, -2, 1, 6, -6, 6, 9]$$

Then we generate a pseudo-uniform distribution of $N_u = 10^4$ points on the unit sphere of $\mathbb{R}^8$ centered at M, using the method of Muller (1959) and Marsaglia (1972). For each such generated unit direction $\boldsymbol{u}$ we find the intersection between the half line $L(\boldsymbol{u}) = \{M + \lambda \boldsymbol{u} \,|\, \lambda > 0\}$ and $\partial\mathcal{D}_C$ by using the bisection method: we first find (by trial and error) a point $P \in L(\boldsymbol{u})$ for which the corresponding Poly4 yield function is not convex; Then, a point $\boldsymbol{p} \in (M, P)$ is found by bisection such that the minimum Gauss curvature K_G of its corresponding yield surface is close to zero, i.e., $K_G \in$[1e-5,5e-4). The process is repeated for the opposite direction $L(-\boldsymbol{u})$.

For each unit direction $\boldsymbol{u}$ we record all bisection steps with the corresponding curvatures thus obtaining a set $\mathcal{S}_T$ of points distributed in the vicinity of $\mathcal{D}_C$, both inside and outside of $\mathcal{D}_C$, for each point having also recorded the minimum of the Gauss curvature K_G of its corresponding yield surface. The resulting set $\mathcal{S}_T$ consists of $N_T = 267647$ points and was used as input dataset to fit a NN-model of $\partial\mathcal{D}_C$. For validation purposes, we also generated a second pseudo-uniform distribution on the unit sphere consisting of 5000 points to obtain a set $\mathcal{S}_V$ consisting of $N_V = 133927$ points in the vicinity of $\partial\mathcal{D}_C$.

The center of $\mathcal{S}_T$ is (here the reported numerical values are rounded to two digits but the precise values can be found in the data repository)

$$C = \frac{1}{N_T} \sum_{p \in \mathcal{S}_T} \boldsymbol{p} = \\ [-1.61, 3.45, -1.77, 1.86, 6.11, -5.92, 6.19, 9.13] \tag{21}$$

and the spread of the data about it is

$$\Delta_T = \left[\max_{p \in \mathcal{S}_T} |p_i - C_i| \,\middle|\, i = 1, ..., 8 \right] = \\ [8.42, 60.72, 100.28, 123.27, 78.45, 72.87, 125.63, 246.01] \tag{22}$$

The spread is quite large and nonuniform. Then the normalized dataset $\mathcal{S}_N$ defined by

$$\mathcal{S}_N = \left\{ \left[\frac{p_i - C_i}{(\Delta_T)_i} \right]_{i=1,...,8} \,\middle|\, \boldsymbol{p} \in \mathcal{S}_T \right\} \tag{23}$$

will be used for fitting purposes. More about the structure of $\mathcal{S}_T$ will be shown next where its NN-model is discussed.

Before closing this sub-point though, some notes on the convexity of the yield surface $\mathcal{Y}_f(\boldsymbol{p})$ corresponding to a point $\boldsymbol{p} \in \mathcal{S}_T$ are in order. Given a parameter point $\boldsymbol{p} \in \mathbb{R}^8$, the Gauss curvature K_G of the corresponding Poly4 yield surface is calculated on a pseudo-uniform dense grid (consisting of more than 10^5 points) on the unit sphere of the 3d stress space $[\sigma_x, \sigma_y, \sigma_{xy}]$. Then the minimum value of K_G is recorded. If this is positive, the point $\boldsymbol{p}$ generates a convex surface, otherwise not-convex. The whole procedure is relatively fast[11] since the formula for K_G is explicit for hypersurfaces described as level sets $f(\boldsymbol{\sigma}) = c$. Hence this approach is expected to scale well to any plane stress yield function and also to yield functions in 6d (or 5d if pressure independence is explicitly accounted for).

7.2. The NN-model

Since each point in $\mathcal{S}_T$ has associated with it a scalar measure of convexity - K_G - there are two possible approaches: A regression model, or a binary classification model. In this work we used the latter[12] with the set of corresponding labels set to 0 if $K_G \geq 0$ and to 1 if $K_G < 0$.

Homogeneity is no longer a restriction in this case (and hence we will employ trainable biases) but smoothness is again crucial for our problem because our algorithm for solving the problem in eq.(18) relies on calculating the gradient ∇g on the boundary of the convexity domain. This again eliminates ReLU as activation function. Instead, in order to preserve as much as possible from its merits, we employed a smoothed approximation defined by

$$\Phi_R(z) = \begin{cases} \dfrac{1}{k} \exp(kz), & z < 0 \\ z + \dfrac{1}{k}, & z \geq 0 \end{cases} \tag{24}$$

The above is C^1 smooth and for our problem[13] we used it with $k = 100$.

We experimented with various architectures (width and depth) of densely connected NNs. Here too the observation made in the previous sections appears to remain valid: For our problem at least, and in the context of (feed-forward, aka sequential) densely connected networks, depth does not bring noticeable improvements over wide shallow architectures. So we settled for a network with three layers: Two hidden layers, each with 200 units (nodes) activated by Φ_R in eq.(24) and the output layer with a single unit activated by the sigmoid function, i.e., $z \longrightarrow 1/[1 + \exp(-z)]$. The output of the network, call it $m(\boldsymbol{p})$, is evenly spread about the threshold value $t = 0.5$. Then the NN-model predicts that $\boldsymbol{p}$ is the set of parameters of a convex yield surface if $m(\boldsymbol{p}) <= 0.5$, and not convex otherwise. The g-function describing the convexity domain is then defined by

$$g(\boldsymbol{p}) = m(\boldsymbol{p}) - t \tag{25}$$

with the threshold value t =0.5. Since the range of m is within $[0, 1]$, the range of g is within [-0.5,0.5].

[11] We used the the free-tier of Google-Colab for this task. It gives access to machines with $\approx$ 12GB of RAM. On one such machine, we generated the whole dataset $\mathcal{S}_T$ and its corresponding K_G's in about 5 hours

[12] We thus emphasize that we do not predict the Gauss curvature K_G. It is clear that, by categorical labeling, some information is lost. A regression model, capable of satisfactory predictions of K_G, would require a lot more input data and a larger model.

[13] We also experimented with other smoothed versions of ReLU, e.g. ELU of Clevert et al (2016), but Φ_R outperformed them in terms of model accuracy and convergence rate.

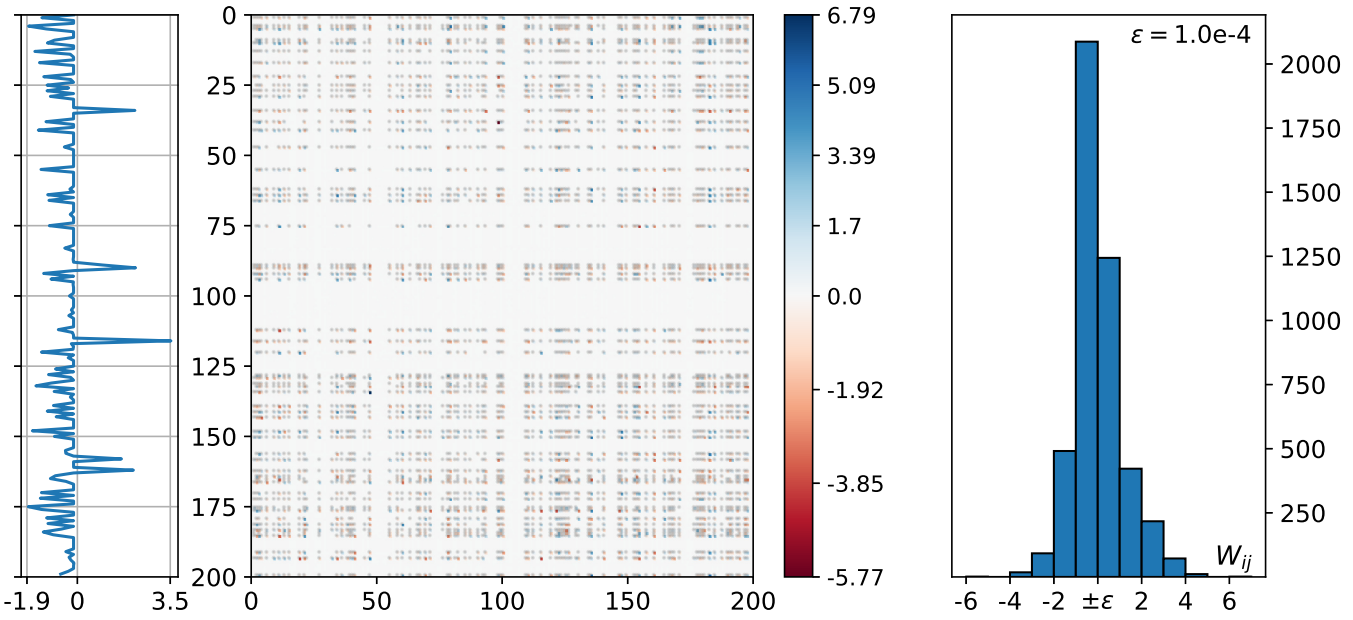

Figure 16: Second layer of the NN-model of the convexity domain of Poly4. **Center:** Heat map of the 200×200 matrix W of weights (a row represents a computational unit). **Left:** Corresponding biases. The peaks correlate with the rows (units) featuring larger (in absolute value) weights. **Right:** Histogram showing the distribution (numbers) of weights $|W_{ij}| \geq \epsilon$. Out of 40000 matrix entries, only 4656 have magnitude larger than ϵ.

This network was implemented with `Tensorflow` and fitted to $\mathcal{S}_T$ using the Adam gradient descent based optimizer, with batches of 8 samples, in three stages: 100 epochs at a learning rate of 1.e-3, followed by 80 epochs at learning rate 1.0e-4, and finally 30 epochs at learning rate 1.0e-5. The final accuracy achieved on $\mathcal{S}_T$ was $\approx$ 86.7%. The whole fitting process took about 5 hours on a free-tier Google-Colab machine.

Several comments are in order. No NN-architecture was able to predict a better accuracy than[14] 87%. This percentage seems to be the upper limit for this simple structure (densely connected). As Fig.16 shows, the matrix of the second layer is quite sparse and hence only a small fraction out of the total of 42201 parameters is actually relevant for the final model (the model being amenable to size reduction -aka pruning - before a potential deployment). Fig.17 shows a more detailed account of the accuracy of the final NN-model. Its accuracy is preserved on the validation dataset (confirming the regularity of the underlying data: the convexity domain itself is convex). From the histogram on the right of Fig.17 one can observe that most of the accuracy loss happens, as expected, in the (-1,1) interval of curvatures, i.e., near the boundary of the convexity domain. A quantitative assessment of this is shown in Fig.18 where eq.(25) is evaluated on points near the boundary of the convexity domain. Ideally, the scattering would be confined to a small band around zero, with most points of positive curvature falling below zero and most points of negative curvature falling above zero. Our NN-model does place most points of positive curvature bellow the threshold, but it also concentrates there quite a significant number of points of negative curvature. The magnitude of the scatter is measured by the variation of g which is quite large (considering the thinness of the band around the boundary), filling the interval $[-0.4, 0.4]$.

For the center C of the input dataset, defined in eq.(21), we have $g(C) = -0.5$. Table-1 and the first three rows of Table-2 show the predictions of g for Poly4-models of various materials reported in previous works. Notably, most models are classified as convex with a value $g(\boldsymbol{a}) \approx -0.5$, the same as the center C. This gives us a 'visual' hint on the overall shape of g: it features a flat bottom, where most of its depiction of the convexity domain lies, and a steep increase -an abrupt wall - at the boundary of the convexity domain, as suggested by the variation of g in Fig.18. Also notable are the two models with IDs 15 and 15B. Model 15B is convex and is also predicted convex by g. Yet it features zones of very small curvature (its minimum K_G being $\approx$2.9e-4), being close to the true boundary of the convexity domain. However, it is located on the flat bottom of g (since $g(15B) = -0.5$) and hence well within the predicted convexity domain. This implies that there are false positives near model 15B. This is confirmed by model 15: Its parameters do not generate a convex yield surface, the latter having a negative minimum curvature of $K_G \approx -0.014$, the zones of negative curvature featuring as blue patches in Fig.19-Left, but g predicts it as being convex.

[14]The true accuracy of the model on the $\mathcal{S}_T$ dataset is actually 88.1%. However, from Fig.17-Left, one can observe a slight imbalance of $\mathcal{S}_T$, the subset of non-convex points (comprised of 143707 points) having approximately 20000 points more than the subset of convex points (comprised of 123940 points). This leads to a slightly worse fitting of the boundary precisely in the zones of interest for actual sheet metal materials. To alleviate this, we duplicated 20000 points in the subset of convex points, those associated with the smallest curvature, i.e., near the boundary. This effectively doubles the weights of these points and leads to a better representation of the boundary, albeit with a slight loss in the overall score, i.e., the reported 86.7%.

Category	Predicted Convex	Predicted Not-Convex	Row Totals
True Convex	119957 (96.8%) 58876 (94.9%)	3983 (3.2%) 3170 (5.1%)	123940 62046
True Not Convex	31493 (21.9%) 16170 (22.5%)	112214 (78.1%) 55711 (77.5%)	143707 71881
Column Totals	151450 75046	116197 58881	267647 133927

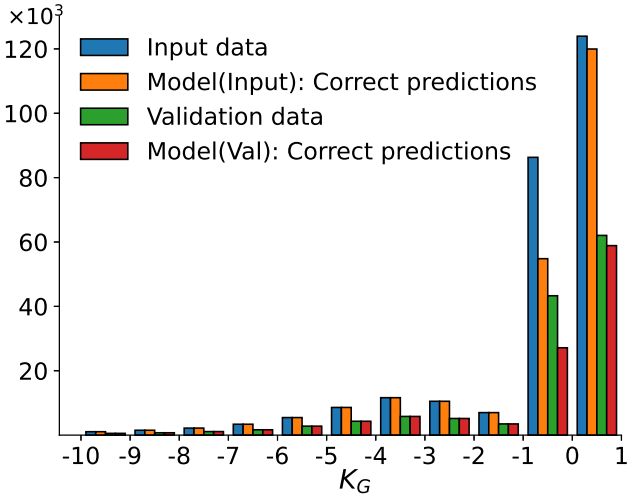

Figure 17: Accuracy assessment of the NN-model of the convexity domain of Poly4. **Left:** Confusion table. Each data cell records two values: Top = Input (fitted) dataset; Bottom = Validation dataset. Percentages refer to 'Row Totals'. The accuracy of the model on the input data is the percentage of the total correct predictions, i.e., (119957+112214)/267647≈ 86.7%. Similarly, the accuracy of the model on the validation dataset is ≈ 85.6%. **Right:** Histogram showing the number of points in the input and validation datasets according to the corresponding Gaussian curvatures and the counts of the correct model predictions (of the sign of the Gaussian curvature).

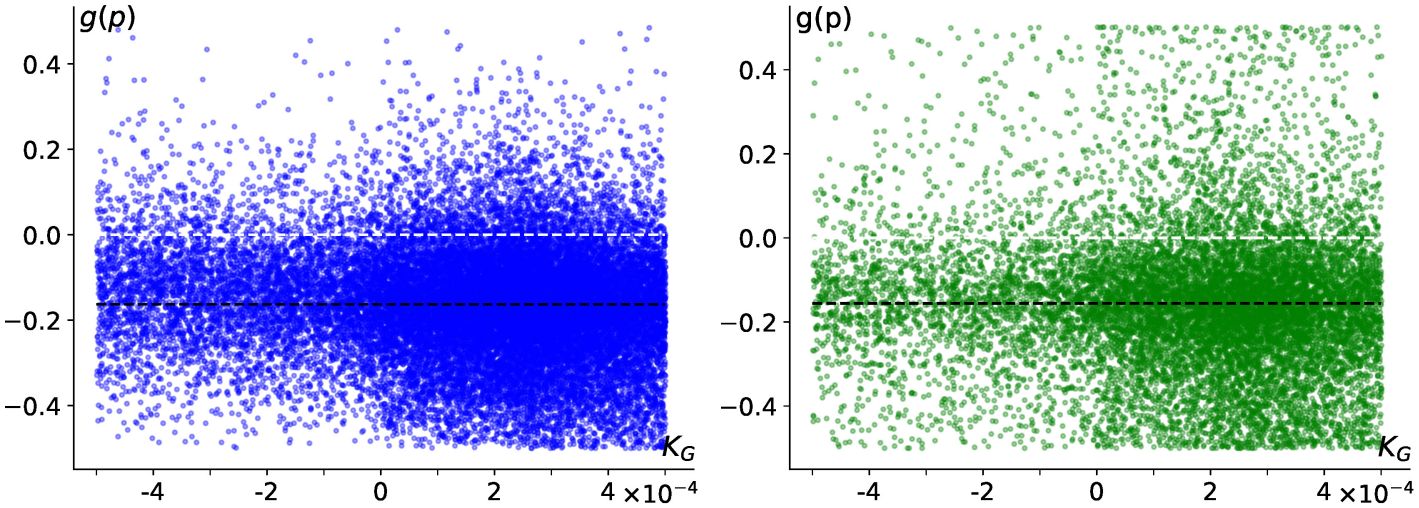

Figure 18: NN-model of the convexity domain of Poly4: Predictions on a thin band around the boundary, i.e., points $\boldsymbol{p}$ in the input (Left) and validation (Right) datasets for which $K_G \in [-\epsilon, \epsilon]$, where ϵ =5.0e-4. The black dashed lines show the average values of the NN-model m, see eq.(25), on the thin band around the boundary as sampled by the input and validation datasets: 0.337 and 0.344, respectively.

7.3. Solving the optimization problem

Having a description of the convexity domain, we are now in the position to tackle the optimization problem in eq.(18). In order to simplify the search for a solution, we assume that the yield function depends linearly on its parameters. This condition is satisfied in general by functions expressed as series expansions in terms of basis functions, and by any PolyN in particular. This, in conjunction with the euclidean distance between predictions and data, renders the cost function Φ_f in eq.(18) quadratic with respect to its parameters $\boldsymbol{a}$:

$$\Phi_f(\boldsymbol{a}) = \frac{1}{2}(\boldsymbol{K} : \boldsymbol{a}) \cdot \boldsymbol{a} - \boldsymbol{k} \cdot \boldsymbol{a} + k_\delta \tag{26}$$

where the positive definite matrix $\boldsymbol{K}$, the vector $\boldsymbol{k}$ and the constant k_δ are completely determined by the data and weights, e.g. Soare and Diehl (2023). A global optimizer $\boldsymbol{a}_0$ then exists, given by the formula

$$\boldsymbol{a}_0 = \boldsymbol{K}^{-1} : \boldsymbol{k} \tag{27}$$

where the inverse $\boldsymbol{K}^{-1}$ may be interpreted in generalized form (as singular value decomposition inverse) if Φ_f is not strictly definite, e.g. when data is not sufficient to match the number of parameters. $\boldsymbol{a}_0$ is taken as initial guess for our solution search algorithm. If $g(\boldsymbol{a}_0) \leq 0$, the corresponding yield surface is (predicted) convex and $\boldsymbol{a}_0$ is accepted as solution.

When $g(\boldsymbol{a}_0) > 0$, the global optimizer is outside the convexity domain and the algorithm proceeds as follow. First, the unique intersection point $\boldsymbol{b}$ between the boundary of the convexity domain $g^{-1}(0)$ and the segment joining the center C with $\boldsymbol{a}_0$ is found by a sequence of bisections. With $\boldsymbol{n} = \nabla g(\boldsymbol{b})/|\nabla g(\boldsymbol{b})|$, i.e., the unit vector along the exterior normal of the boundary at $\boldsymbol{b}$, the search direction is then

$$\boldsymbol{\delta} = -\nabla\Phi_f + \left(\nabla\Phi_f \cdot \boldsymbol{n}\right)\boldsymbol{n} \tag{28}$$

i.e., the projection of the descent direction $-\nabla\Phi_f$ on the tangent plane of the boundary at $\boldsymbol{b}$. The new approximation

to the solution is then

$$\boldsymbol{a}_1 = \boldsymbol{b} + \lambda\boldsymbol{\delta} \tag{29}$$

with the scaling factor $\lambda > 0$. If $g(\boldsymbol{a}_1) \leq 0$, i.e., $\boldsymbol{a}_1$ is inside the convexity domain (which may happen if the latter has local straight edges), the next solution update is made using the descent direction of Φ_f. Otherwise, the tangential component is employed at the intersection point of the segment $(C, \boldsymbol{a}_1)$ with the boundary, etc.

The algorithm stops when either the magnitude of the projection becomes too small, i.e. $|\boldsymbol{\delta}| < \epsilon$, or some maximum number of iterations is consumed. Typical values for λ are in $[0.1, 0.3]$. If the quadratic in eq.(26) is semi-definite (e.g., when certain parameters are not uniquely determined by the data, which usually happens in case of insufficient data) the algorithm takes an oscillating path along a long valley. One can alleviate this by increasing the scaling factor $\lambda \in [1.0, 3.0]$.

Finally, let us recall, based on the findings in the previous subsection, that the NN-description of the convexity domain is just an approximation, which may contain, in the worse case scenario, a non-negligible subset of false positives (non-convex functions classified as convex). The optimization algorithm may then end at a false positive solution. In particular, it may happen that $\boldsymbol{a}_0$ itself is a false positive. In this case, let $\tau = m(\boldsymbol{a}_0) \geq 0$. To ensure a convex solution, the optimization problem is solved on the smaller domain

$$\mathcal{D}_\tau = \{\boldsymbol{a} \in V \,|\, m(\boldsymbol{a}) - \epsilon_\tau \leq 0\} \subset \mathcal{D}_C$$

where ϵ_τ is a number in the interval $(0, \tau)$. This turns $\boldsymbol{a}_0$ into a true negative relative to the convexity subdomain $\mathcal{D}_\tau$. The threshold ϵ_τ can be determined with a few trial-and-error steps, until the solution is a true positive. The entire procedure is summarized in Box-1.

In Box-1, $\boldsymbol{a}_0$ can be the unconstrained estimate in eq.(27), or a false positive outcome of the second `IF`-branch. We illustrate the optimization procedure by modeling the aluminum alloy AA2090-T3, for which the following data was reported in Aretz and Barlat (2004): (uniaxial) directional yield stresses $\overline{\sigma}_\theta$ and r-values r_θ, for $\theta = k\pi/12$, $k \in 0, 1, ..., 6$, and balanced biaxial yield stress $\overline{\sigma}_b$ and r-value r_b. With the stress data normalized (i.e., $\overline{\sigma}_0 = 1$), the remaining 15 data points cannot be matched exactly by the 8 parameters of Poly4. In such cases one proceeds with the next best option which is to better fit a subset of the data. This was the approach taken in Soare et al (2008) where the highly customized (semi-explicit) solution ID#3 in Table-2 was obtained by matching exactly $\{\overline{\sigma}_0, \overline{\sigma}_{45}, \overline{\sigma}_{90}\}$, $\{r_0, r_{45}, r_{90}\}$ and $\overline{\sigma}_b$, the one remaining parameter being optimized to fit the rest of the data (subject to convexity constraints).

Box.1: Optimization procedure

`Set:` $\boldsymbol{a}_0, k_G = K_G(\boldsymbol{a}_0), \tau = m(\boldsymbol{a}_0)$

`IF` $(\tau \leq 0.5)$ `AND` $(k_G \geq 0)$

  `Solution =` $\boldsymbol{a}_0$

`ELSE`

  `Set initial guess:` $\boldsymbol{a} = \boldsymbol{a}_0$

  `IF` $(\tau > 0.5)$

$$\texttt{Solve:} \begin{cases} \min\limits_{\boldsymbol{a}} \Phi_f(\boldsymbol{a}) \\ m(\boldsymbol{a}) - 0.5 \leq 0 \end{cases}$$

  `ELSE`

$$\texttt{Set:}\ \epsilon_\tau \in (0, \tau)$$

$$\texttt{Solve:} \begin{cases} \min\limits_{\boldsymbol{a}} \Phi_f(\boldsymbol{a}) \\ m(\boldsymbol{a}) - \epsilon_\tau \leq 0 \end{cases}$$

This kind of customization is made possible by the linear dependence of PolyN on its parameters, reflected in the quadratic nature of the associated cost function in eq.(26). Any explicit solution to the fitting problem is then obtained via eq.(27) with appropriate weights. The solution ID#4/Table-2 was obtained by using the overall weights 0.1 and 0.9 for all r-value data points and for all yield stresses, respectively. It coincides with the initial guess $\boldsymbol{a}_0$ given by eq.(27), since, in this case, the constraint $g(\boldsymbol{a}_0) \leq 0$ is satisfied and also $K_G(\boldsymbol{a}_0) > 0$. The corresponding yield surface model is shown in Figs-24,25 in Appendix-C: it features an undesirable oscillation in both stresses and r-values near $\theta = 15^o$.

One can improve sol ID#4 by customizing the weights. A simple yet very effective way of doing this is by repeating data, i.e., while the weights for stresses and r-values are uniform, one can increase the weight of a specific point by adding (sampling) it to the cost function multiple times. Solution ID#5 was obtained by using the same overall weights as above, with r_0 sampled 3-times, $\overline{\sigma}_{15}$ 11-times, r_{45} 5-times, $\overline{\sigma}_{45}$ 3-times, and r_{90} and $\overline{\sigma}_{90}$ each sampled 3-times (the rest of the points being sampled only once). The resulting solution, ID#5 in Table-2, is shown in Figs-26,27 in Appendix-C: the improvement is obvious, this solution being quite close to sol ID#3. Even more remarkable, this solution is still the initial guess $\boldsymbol{a}_0$ (i.e., explicit), since $g(\boldsymbol{a}_0) \leq 0$ and $K_G(\boldsymbol{a}_0) > 0$. This success is explainable by the reduced flexibility of Poly4 (which will, in most cases, be convex if the input data is conducive to a convex shape).

Trying to further improve the prediction of r_{45}, r_{90} and $\overline{\sigma}_b$, we used the same input as for ID#5 but with r_{45} sampled 8-times, r_{90} sampled 5-times, and $\overline{\sigma}_b$ sampled 2-times. This time the corresponding initial guess, reported as ID#6 in Table-2, and shown in Figs-28,29,30 in Appendix-C, is no longer convex. It is a false positive, since the NN-model classifies it as convex. With the threshold $\epsilon_\tau = 0.0865$, the first and second `ELSE`-branches of the algorithm in Box-1 are activated to produce solution ID#7/Table-2, shown here in Fig-19/Right and Fig-20, which is convex and also cor-

Table 1: Parameters of several Poly4 models reported in Tong and Alharbi (2017). The IDs were kept as in the cited reference with one exception: '15B', reported in Tong (2018) as a correction to '15'. Column K_G records the global minimum Gauss curvature. Column $g(\boldsymbol{a})$ records the prediction of the NN-model in eq.(25) with the threshold $t = 0.5$.

ID	Material	$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$	K_G	$g(\boldsymbol{a})$
10	Brass	-1.8644, 2.9499, -2.1475, 1.1347, 6.375, -6.1573, 6.9275, 10.3633	0.081	-0.5
11	Copper	-1.7363, 2.5304, -1.9759, 1.2236, 6.3702, -6.5197, 7.057, 11.4808	0.065	-0.5
12	Aluminum	-1.3422, 2.4433, -1.8689, 0.9905, 6.0261, -1.6115, 6.5337, 6.5264	0.186	-0.5
13	A5052-O	-1.6744, 1.9748, -1.4843, 1.0, 5.8452, -2.0841, 5.6551, 6.4509	0.191	-0.5
14	A5182-O	-1.6608, 2.287, -1.9816, 1.0281, 5.7451, -4.7334, 6.122, 8.643	0.099	-0.5
15	A6016-T4	-1.7273, 2.3675, -1.9619, 1.2945, 6.3148, 0.0996, 7.1386, 5.0025	-0.014	-0.499
15B	A6016-T4	-1.7273, 2.3675, -1.9619, 1.2945, 6.3016, 0.0289, 7.0354, 5.1532	2.9e-4	-0.5
16	AISI 430	-1.7401, 2.699, -2.1818, 1.0, 5.7401, -5.8931, 6.1818, 9.1941	0.079	-0.5
17	AISI 409	-2.2143, 3.0112, -2.2456, 1.0, 6.2143, -6.7541, 6.2456, 9.7428	0.072	-0.5
18	DD Steel	-2.2456, 3.6695, -2.6918, 1.0347, 5.9369, -6.1607, 6.4525, 7.9083	0.115	-0.5
19	AISI 304	-1.9167, 3.197, -2.3152, 1.2518, 6.4377, -6.9885, 7.3398, 11.0854	0.071	-0.5
20	IF Steel	-2.7179, 4.177, -3.0688, 1.0327, 6.8832, -9.6978, 7.2993, 11.8839	0.047	-0.5
21	440W HSS	-1.6331, 2.5308, -2.034, 1.0495, 5.5837, -5.5725, 6.0835, 8.9923	0.081	-0.5

Table 2: Parameters of several Poly4 models: IDs 1, 2 and 3, reported in Soare et al (2008); IDs 4 to 7 obtained by solving the problem in eq.(18). Column K_G records the global minimum Gauss curvature. Column $g(\boldsymbol{a})$ records the prediction by the NN-model in eq.(25) with the threshold $t = 0.5$.

ID	Material	$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$	K_G	$g(\boldsymbol{a})$
1	AA2008-T4	-1.8579, 2.9549, -2.0742, 1.4946, 6.56, -4.1447, 7.949, 8.1031	0.1822	-0.5
2	Steel(NUM'93)	-2.5663, 3.6988, -2.4392, 0.8784, 5.7851, -7.6630, 5.8435, 8.2863	0.0696	-0.5
3	AA2090-T3	-0.6984, 1.4969, -2.3838, 1.4568, 4.8808, -1.0150, 8.7095, 23.4498	0.001	-0.4164
4	AA2090-T3	-0.7622, 1.3920, -2.2382, 1.4859, 2.2711 , 1.1774, 10.0092, 23.3333	0.001	-0.4982
5	AA2090-T3	-0.6779, 1.3931, -2.3281, 1.5014, 4.9212, -0.8580, 8.8894, 23.4782	5.4e-5	-0.3746
6	AA2090-T3	-0.6779, 1.4165, -2.3563, 1.4968, 5.0471, -1.4173, 9.2527, 23.5159	-0.026	-0.2873
7	AA2090-T3	-0.6826, 1.4252, -2.3541, 1.4974, 5.0515, -1.4367, 9.2395, 23.4547	0.0001	-0.4134

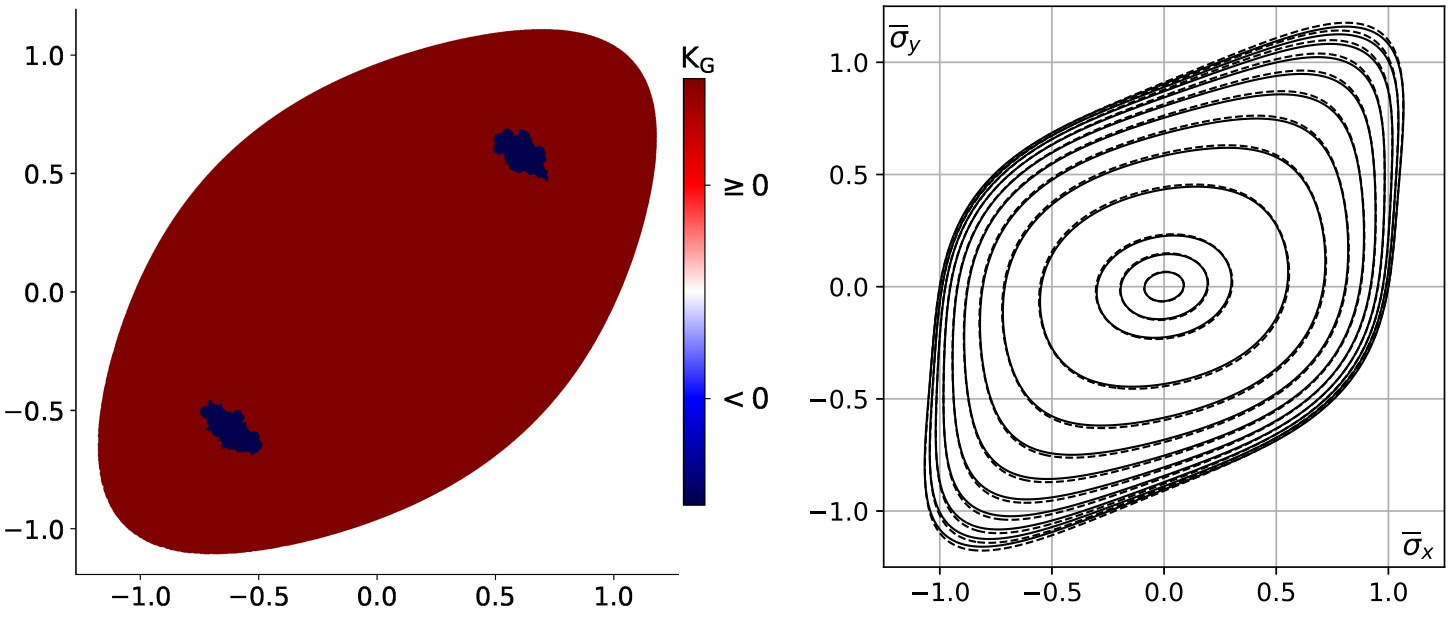

Figure 19: **Left:** Verification of the convexity of model ID#15/Table-1. **Right:** $\sigma_{xy} = const$ sections through the yield surfaces of the two Poly4 models shown in Fig.20: ID#7-solid, ID#3-dashed.

rectly classified by the NN-model.

8. Conclusions

8.1. General comments

Homogeneity is a fundamental property of yield functions in metal plasticity. By using non-zero biases, NN-models reported previously in the literature are not homogeneous.

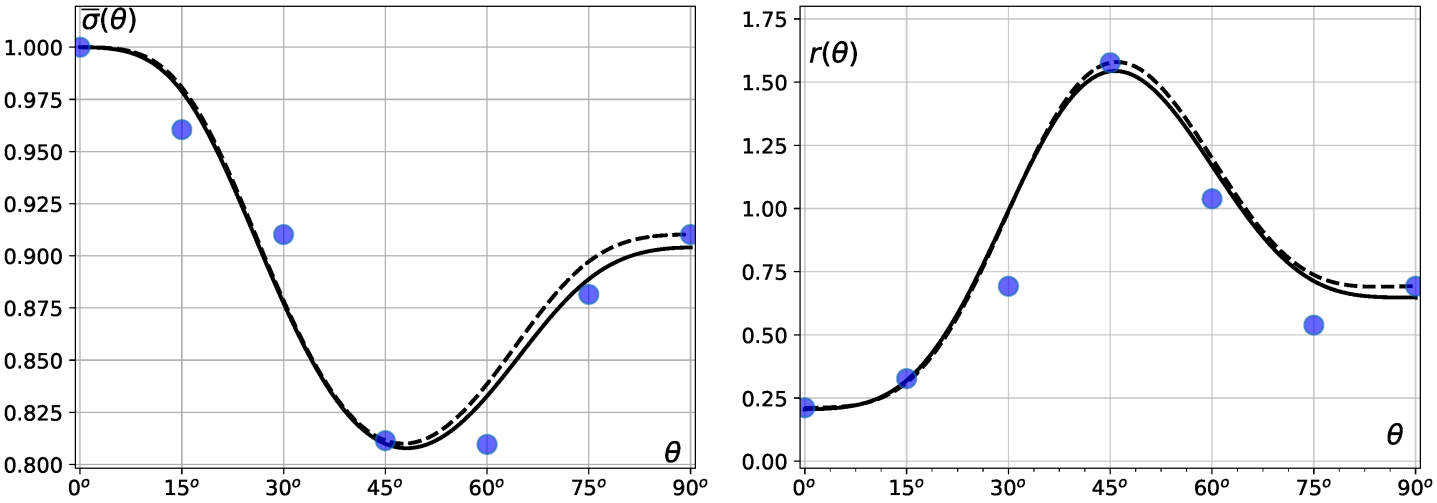

Figure 20: Directional yield stress (Left) and r-values (Right) of the Poly4 model of AA2090-T3 obtained by solving the problem in eq.(18): row ID#7 in Table-2 shown with solid line. Comparison with the customized Poly4 model reported in Soare et al (2008) - row ID#3 (dashed). The markers show the experimental data reported in Aretz and Barlat (2004).

Such NN-models of the yield function cannot be easily fit to the traditional directional properties of sheet metal, and must either map a large portion of the stress space, e.g., Vlasis and Sun (2021), Fuhg et al (2023), or require solving a nonlinear equation for each evaluation of the status of the deformation (elastic/plastic), e.g., Nascimento et al (2023), a task completed by a homogeneous function in just one evaluation. Thus NN-models in plasticity should use homogeneous activation functions and zero biases.

Smoothness is another fundamental property required of yield functions for two basic reasons. The first is that yield surface gradient determines the flow direction and hence it has a significant impact on the predicted deformations. The second is that virtually all numerical integration routines used in finite elements codes rely on various forms of the return mapping algorithm, which in turn relies on yield surface gradients (and hessians in case of implicit codes) to calculate the stress state from displacements. Then the requirement of smoothness, in the context of a direct modeling of yield surface level sets, places strong restrictions on the usage of ReLU, currently one of the most popular activation functions in the machine learning community. As shown here, Fig.3/Right, unmodulated usage of ReLU leads to nonphysical oscillations (saw-tooth like) in the predicted r-values.

One can still use ReLU if the input space is enriched with additional features, as in eq.(12), a method often used in machine learning for classification problems. However, in the present context this method leads to raw polynomial (or harmonic polynomial) yield functions for which superior calibration algorithms are already available, Soare (2023) and Soare and Diehl (2023). A better approach is to employ smoothed ReLUs or power functions, as proposed in eqs.(5) and (6), respectively. This effectively bridges dense NNs with the traditional approach to the modeling of yield functions since the output of networks employing such activation functions is a particular form of polynomial or piecewise polynomial function. In the particular case of a shallow network (one hidden layer), the FACET yield criterion of Van Houtte et al (2008) is recovered. More generally, the mechanism of over-saturation of parameters used by NNs is also employed by yield functions based on multiple linear transformations, Plunket et al (2008), Soare and Barlat (2011), Aretz and Barlat (2013).

It may be concluded that dense NNs with smooth and homogeneous activation functions provide a powerful modeling framework, being capable of handling arbitrary data. As shown here, good results can be achieved even with small-sized networks, which are amenable for deployment in finite element codes (Appendix-B). Automatic convexity can be easily enforced and, by their very recursive nature, NNs are expected to scale well to the 5d sub-space of general deviatoric stress states.

8.2. Mapping a convexity domain

As trade-off, for the increased modeling power of NNs, one has to accept a high degree of over-parametrization. In particular, NN-models, in general, are not amenable to analytical calculations, optimization being the sole mean of manipulating their parameters (unless a network is trivially simple, such as the four neurons network depicted in Section-4). On the other hand, yield function models formulated as classical series expansions in terms of basis functions (e.g., polynomial, trigonometric/harmonic expansions), do have some attractive properties with respect to their parameters: these are independent, the dependence on them is linear, and particular data points can be fitted exactly, via explicit formulas. However, by comparison with the NN-models discussed here, convexity is not as easy to enforce, the convexity domain (the set of parameters for which the expansion is convex) being a (convex) cone[15]

[15]If the yield function is normalized with the yield stress along, say, the rolling direction, only a plane section through this cone is of interest, this section being a bounded set.

whose shape is, in general, not as simple as that featured by a densely connected NN (for which the convexity domain is a Cartesian product with cones of positive parameters). The question then arises of whether is possible and feasible to describe in some analytical form the convexity domain of a given expansion.

This was the second problem studied here: to map the convexity domain of a yield function that is not convex by default. As a specific example we used Poly4, the simplest non-quadratic polynomial yield function, with a relatively low dimensional parameter space (only 8 independent parameters). We approached it as a classification problem and, using a shallow feed forward densely connected NN, were able to characterize the convexity domain of Poly4 with an accuracy close to 87%. This level of accuracy is only marginally satisfactory since it incorporates into the predicted convexity domain a non-negligible amount of false positives (not convex predicted convex) located in zones of interest for the modeling of sheet metal. Nevertheless, in the case of the alloy A2090-T3 the NN-model of the convexity domain of Poly4 was able to guarantee the convexity of the solutions obtained by minimizing the distance between data and Poly4 predictions. The methodology, data generation scheme and optimization algorithm described here are general and applicable to a wide range of analytical formulations in terms of basis functions: In principle, the calibration of any such yield function can be reduced to that of solving an optimization problem subject to a single constraint.

8.3. NNs vs traditional yield functions.

Finally, a comparison between NN yield surface models and the (more or less) traditional approach based on series expansions (the latter incorporating as particular cases virtually all yield functions employing small sets of parameters, e.g., Hill'48, Yld'91, Yld2000-2D, BBC2005, Yld2004-18p, etc) may be useful for guiding potential applications and future research. This is made in terms of input data, calibration, modeling power and simulation performance.

The amount of data used here for the calibration of NN-models may appear to be excessive by comparison with the small datasets typically employed in both academic and industrial environments to describe the plasticity of sheet metal: these are based mostly on the uniaxial mechanical properties of the sheet, being comprised, at a bare minimum (in the case of a tension-compression symmetric material), of six data points - the three yield stresses $\{\sigma_0, \sigma_{45}, \sigma_{90}\}$ and the corresponding r-values $\{r_0, r_{45}, r_{90}\}$. One usually supplements this minimal dataset by adding the balanced-biaxial yield stress, σ_b, based on experiments or estimation. Seven data points are clearly insufficient for the calibration of both NN and (high order) series expansion models. At a more fundamental level, one cannot uniquely determine a surface, the yield surface, by specifying only one curve on it - the curve traced by the set of (all) uniaxial experiments. Particular instances of yield functions compensate this lack of uniqueness by making assumptions about the overall shape of the yield surface: Hill'48 is the class of ellipsoids, Yld'2004-18p is the class of o particular set of transformations of the more general shapes provided by the Hershey-Hosford isotropic function, etc. More generally, an algorithm for generating a large class of *proto-models* of a yield surface based on the interpolation of the minimal dataset has been developed by Soare (2023). By contrast with the traditional yield functions, proto-models fit all the available data exactly and their accuracy increases with added information (e.g., from other mechanical tests or from virtual testing). Then, by sampling a proto-model one can generate any amount of data to be used for calibration.

Regarding modeling power, NN and series expansion models are similar (as illustrated here, since all data was extracted from series expansion models). This should not come as a surprise: the level sets of (even-degree) homogeneous polynomials are dense in the set of centrally symmetric and convex surfaces, e.g., Varjú (2007), while spherical harmonics are dense in the wider set of asymmetric surfaces, e.g., Atkinson and Han (2012), Groemer (1996).

Regarding simulation performance, fast evaluation algorithms for homogeneous polynomials have been shown to yield simulation times comparable with Hill'48 for degrees as high as 12, Soare and Diehl (2023). Due to their simple iterative evaluation scheme, shallow NN-models, if moderate in size, are not expected, in general, to be far from this benchmark. This is confirmed here in Appendix-B, where an algorithm for the evaluation of the output, gradient and hessian of a NN-model is presented. Finite element simulations of uniaxial tensile tests and of cylindrical cup-drawing, show that NN-models feature simulation runtimes that are comparable with those of traditional yield functions. Furthermore, the cup height profiles reported in Fig.21 of Appendix-B are almost overlapping, demonstrating that a *simulation should be agnostic to the yield function employed*, if the latter properly fits the data that is relevant to the simulation. This is the premise of a *data-driven* constitutive framework. By contrast, finite element simulations based on yield functions with small numbers of parameters (essentially identified by uniaxial mechanical properties) show significant variability in results, depending on the underlying assumptions about the shape of the yield surface (as discussed above).

The discussion so far and, in fact, the whole context of this work have been limited to plane stress and isotropic hardening, where NN and series expansion models, as argued in the preceding paragraphs, may be regarded as similar in terms of modeling and simulation performance. Extensions to general (aka 3d) stress states and non isotropic hardening (also referred to as anisotropic, or differential, or distortional hardening) pose significant challenges for either approach. NN models do have an advantage here, since their extension from plane stress to 3d stress states is straightforward and with almost no additional computational cost, whereas for (high order) series expansion mod-

els, at least in their canonical form, the computational cost increases significantly as the number of stress space dimensions increases to 6. In either case, the data generating scheme is by far the most important ingredient. Proto-models can still be constructed (in case of isotropic hardening) by considering sequences of plane-stress sections through the 5d yield surface. However, each such section requires at least 7 data points which, for sections not parallel to the rolling plane, can only be estimated by using virtual testing methods. In turn, these rely on crystal plasticity models whose predictive power depends significantly on their proper calibration on the available mechanical data. The calibration of crystal plasticity models is still a matter of active research, e.g. Sedigiani et al (2020), any practical solution to the problem being relevant also to the problem of phenomenological modeling of anisotropic hardening via yield functions[16]. In its most general form, the latter problem is complicated by the fact that yield surfaces, in their canonical interpretation (as elasticity domains), cannot be directly inferred from the long excursions into the plastic domain exhibited by the loading paths usually employed in mechanical/virtual testing since each such path alters the internal structure (texture, hardening levels, grain boundaries, etc) of the material, and hence the yield surface, in its unique way, Hill (1994). The modeling problem then reformulates to that of discovering appropriate evolution laws for the parameters of the yield function. Series expansions models are likely to be easier to work with since their parameters are independent. The problem is simplified considerably if one adopts the approximation of yield surfaces as level sets of the expended plastic work, Hill and Hutchinson (1992). In this case, the anisotropic hardening problem is trivially reduced to the one studied here, where the evolution of the yield surface can be parametrized by a sequence of yield surfaces.

Data: Code and data can be accessed at the repository: `https://github.com/stefanSCS/nnYS`

9. Appendix-A: NN-tables

We record in Table-3 the neural networks used in sections 4-6 of this article. A hidden layer is described by a pair (C, Φ), where C is the number of neurons (nodes) and Φ is their activation function (the same for all nodes in a layer). A network consisting of a stack of n identical hidden layers is described by the notation $n(C, \Phi)$. If the network is a stack of n heterogeneous layers, then we denote this by $(C_1|\Phi_1)|...|(C_n, \Phi_n)$. As discussed in Sections-3,4, all biases are set =0. Then the total number of parameters of a network can be calculated with the formula $\sum_{k=1}^{n} C_k C_{k-1} + C_n$ where C_0 is the number of input nodes, here = 3, i.e., the dimension of the space of plane stresses. For example, the network in `nb00B_fitHill` has a total of $60\times3+ 30\times60+ 30\times30+ 30\times30 + 30 = 3810$ parameters.

Validation is performed by reporting: $\min K_G$, the minimum of the Gaussian curvature calculated on a dense grid on the yield surface; $\Delta_{xy} = \max|\Delta\sigma_{xy}|$, as defined by eq.(10); and $\Delta\sigma$ and ΔR. The last two measure the euclidean distance between directional predictions (stresses and r-values) and corresponding data (generated here by sampling an analytical model). Specifically,

$$\Delta\sigma = \left\{ \sum_q \sum_\theta \left[\overline{\sigma}^p(q,\theta) - \overline{\sigma}^d(q,\theta) \right] \right\}^{1/2}$$

$$\Delta R = \left\{ \sum_q \sum_\theta \left[r^p(q,\theta) - r^d(q,\theta) \right] \right\}^{1/2}$$

where the superscript p indicates predictions of the NN-model, and the superscript d indicates data. The whole yield surface can be parameterized by stress states of the form, Soare and Diehl (2023):

$$\boldsymbol{\sigma} = \sigma(q,\theta) \begin{bmatrix} \cos^2\theta + q\sin^2\theta \\ \sin^2\theta + q\cos^2\theta \\ (1-q)\sin\theta\cos\theta \end{bmatrix}$$

where $\sigma(q,\theta)$ and $q\sigma(q,\theta)$ are the major and minor principal stresses of a biaxial test performed on a sample cut out from the sheet at an angle θ from the rolling direction. To sample the drawing zone of a yield surface, here we used $q \in \{0.0, -0.04, -0.08, -0.12, -0.16, -0.2\}$ and $\theta \in \{k\pi/28 | k = 0, 1, 2, ..., 14\}$ for all NN-models except the ones constructed for AZ31B (based on its SHYqp model) where the r-values in tension increase rapidly with decreasing q. In order not to obscure the uniaxial case in figures, we limited in this case the variation of q to $\{0.0, -0.025, -0.05\}$.

10. Appendix-B: Implementing NNs in FE codes

Here we show the deployment of a NN-model in constitutive subroutines. These require the evaluation of the yield function values and of its first derivatives (and, in implicit FE-codes, of its second order derivatives). Given the recursive nature of NNs, the evaluation algorithm parallels the forward and back-propagation schemes used in their training (calibration). The context here is that of Sections-2 and 3 (no biases). The computational flow in eqs.(1-3) can then be represented more compactly in the form

$$\begin{aligned} \boldsymbol{z}_L &= \boldsymbol{a}_{L-1}^T \boldsymbol{W}_L \\ \boldsymbol{a}_L &= \overline{\boldsymbol{\Phi}}_L(\boldsymbol{z}_L) \end{aligned} \tag{30}$$

[16] An alternative, black-box approach aims at learning a general $\boldsymbol{\sigma}_t = \mathcal{F}(\boldsymbol{\epsilon}_{t_0,t})$ relation between the current stress $\boldsymbol{\sigma}_t$ and the strain history $\boldsymbol{\epsilon}_{t_0,t}$, with $\mathcal{F}$ a recurrent NN, e.g. Mozaffar et al (2019). Despite some technical issues (fitting time series in raw form is dependent on increment size, Bonatti and Mohr (2022)), this approach does manage to completely avoid any constitutive assumptions (phenomenology). However, bringing such functional relations to a level of accuracy that might be acceptable for practical applications is likely to require significantly larger training datasets than what is typically required when adopting at the outset the minimal constitutive framework of metal plasticity - yield function and flow rule.

Table 3: NNs reported in Sections 4, 5 and 6. Notes: Φ_{A1} ≡ReLU; Training/fitting is performed in notebooks `nb*_fit*` and validation in notebooks `nb*_Valid.ipynb` .

Fit/Validation	Sect/Material	Hidden layers	Constraints	$\Delta\sigma$	ΔR	Δ_{xy}	min K_G
`nb00_fitHill`	4/AA6022-T4	(600,Φ_{A1})	*	0.0027	0.2198	0.0028	*
`nb00B_fitHill`	4/AA6022-T4	(60,Φ_{A1})|3(30,Φ_{A1})	*	0.0161	0.4044	0.0068	*
`nb01_fitHill`	4/AA6022-T4	(4,Φ_{S2})	$W_{2\|k} = 1$	0.0005	0.0006	7e-5	0.9628
`nb02_fitHill`	4/AA6022-T4	(4,Φ_{S2})	$W_{2\|k} = 1$	0.0023	0.0088	1e-4	0.9596
`nb07_fitPolyN`	5/AA6016-T4	3(20,Φ_{S2})	*	2.4e-4	0.0039	7e-5	-0.0008
`nb07B_fitPolyN`	5/AA6016-T4	3(10,Φ_{S2})	*	0.0021	0.0414	4.1e-4	-0.0499
`nb08_fitPolyN`	5/AA6016-T4	3(35,Φ_{S2})	*	0.0011	0.0398	3e-4	-0.1107
`nb10_fitPolyN`	5/AA6016-T4	4(10,Φ_{S2})	*	0.0017	0.3634	6.4e-4	-0.3992
`nb13_fitPolyN`	5/AA6016-T4	(20,Φ_{S8})|(20,Φ_{A2})	$W_{2\|jk}, W_{3\|j} \geq 0$	0.0022	0.3478	7e-4	0.1045
`nb14_fitPolyN`	5/AA6016-T4	(27,Φ_{S10})	$W_{2\|k} = 1$	0.0038	0.1109	6.4e-4	0.0900
`nb15_fitSHYqp`	6/AZ31B	(60,Φ_{A10})	$W_{2\|k} \geq 0$	0.0155	6.2518	0.0042	0.0848
`nb16_fitSHYqp`	6/AZ31B	(60,Φ_{A10})	$W_{2\|k} \geq 0$	0.0142	5.7443	0.0053	0.0816
`nb17_fitSHYqp`	6/AZ31B	(60,Φ_{A10})	$W_{2\|k} \geq 0$	0.0222	2.1020	0.0044	0.1025

where $\boldsymbol{a}_L = [a_{L|1}, ..., a_{L|C_L}]^T$ represents the (column) vector of nodes in layer $L \in \{1, ..., \Lambda\}$, and the $C_L \times C_{L-1}$ matrix $\boldsymbol{W}_L = [W_{L|ij}]$ aggregates all the node weights of a layer column-wise[17]. It is assumed that all units/nodes in layer L have the same activation function $\Phi_L : \mathbb{R} \longrightarrow \mathbb{R}$ and then $\overline{\boldsymbol{\Phi}}_{\boldsymbol{L}} : \mathbb{R}^{C_L} \longrightarrow \mathbb{R}^{C_L}$ is the vector function defined by $\overline{\boldsymbol{\Phi}}_L(z) = [\Phi_L(z_1), ..., \Phi_L(z_{C_L})]^T$.

For plane stress the input layer ($L = 0$) is:

$$\boldsymbol{a}_0 = \boldsymbol{\sigma} = [\sigma_x, \sigma_y, \sigma_{xy}]^T = [\sigma_1, \sigma_2, \sigma_3]^T \tag{31}$$

where the second equality is simply a more convenient notation (not to be confused with its more common usage in mechanical engineering where it represents principal stresses). In eq.(30), the output of the last layer $a_\Lambda \in \mathbb{R}_+$ is assumed homogeneous of degree $N \geq 2$ with respect to $\boldsymbol{\sigma}$, as in Section-3. The corresponding yield function is then:

$$g(\boldsymbol{\sigma}) = (a_\Lambda)^{1/N} \tag{32}$$

Its gradient is calculated by:

$$\frac{\partial g}{\partial \sigma_p} = \frac{1}{N} a_\Lambda^{\frac{1}{N}-1} \frac{\partial a_\Lambda}{\partial \sigma_p} = \frac{g}{N a_\Lambda} \frac{\partial a_\Lambda}{\partial \sigma_p} \tag{33}$$

for $p \in \{1, 2, 3\}$. We denote by $\{\boldsymbol{e}_i \,|\, i = 1, .., n\}$ the canonical base of $\mathbb{R}^n$. Then from eq.(30) we have

$$\frac{\partial \boldsymbol{a}_L}{\partial \sigma_p} = \frac{\partial}{\partial \sigma_p} \sum_{i=1}^{C_L} \Phi_L(z_{L|i}) \boldsymbol{e}_i = \sum_{i=1}^{C_L} \Phi'_L(z_{L|i}) \frac{\partial z_{L|i}}{\partial \sigma_p} \boldsymbol{e}_i$$

where, recall, $z_{L|i}$ denotes component i of the vector $\boldsymbol{z}_L \in \mathbb{R}^{C_L}$, and Φ' denotes the derivative of the scalar function Φ. By extension, we shall denote

$$\overline{\boldsymbol{\Phi}}'_L(z_L) = \sum_{i=1}^{C_L} \Phi'_L(z_{L|i}) \boldsymbol{e}_i = [\Phi'_L(z_{L|1}), ..., \Phi'_L(z_{L|C_L})]^T \tag{34}$$

[17]Eq (30.1) was written in Section-2 in the form $\boldsymbol{z}_L = \boldsymbol{W}_L \boldsymbol{a}_L$ where the weight matrix was defined row-wise (its rows correspond to nodes weights). The column-wise storage parallels the row-major storage of vectors in `C/C++` and is employed in virtually all NN software libraries.

Denoting with $\boldsymbol{x} \odot \boldsymbol{y} = \sum_i x_i y_i \boldsymbol{e}_i = [x_1 y_1, ..., x_n y_n]^T$ the component-wise (Hadamard) product of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, the sequence of calculations for $\partial a_\Lambda / \partial \sigma_i$ can now be written as:

$$\begin{aligned} \frac{\partial \boldsymbol{a}_L}{\partial \sigma_p} &= \overline{\boldsymbol{\Phi}}'_L(z_L) \odot \frac{\partial \boldsymbol{z}_L}{\partial \sigma_p} \\ \frac{\partial \boldsymbol{z}_L}{\partial \sigma_p} &= \frac{\partial \boldsymbol{a}_{L-1}^T}{\partial \sigma_p} \boldsymbol{W}_L \end{aligned} \tag{35}$$

Second order derivatives follow a similar pattern:

$$\frac{\partial^2 g}{\partial \sigma_q \partial \sigma_p} = \frac{g}{N a_\Lambda} \frac{\partial^2 a_\Lambda}{\partial \sigma_q \partial \sigma_p} - \frac{N-1}{g} \frac{\partial g}{\partial \sigma_p} \frac{\partial g}{\partial \sigma_q} \tag{36}$$

We have

$$\begin{aligned} \frac{\partial^2 a_L}{\partial \sigma_q \partial \sigma_p} &= \frac{\partial}{\partial \sigma_q} \sum_{i=1}^{C_L} \Phi'_L(z_{L|i}) \frac{\partial z_{L|i}}{\partial \sigma_p} \boldsymbol{e}_i \\ &= \sum_{i=1}^{C_L} \left[\Phi''_L(z_{L|i}) \frac{\partial z_{L|i}}{\partial \sigma_q} \frac{\partial z_{L|i}}{\partial \sigma_p} + \Phi'_L(z_{L|i}) \frac{\partial^2 z_{L|i}}{\partial \sigma_q \partial \sigma_p} \right] \boldsymbol{e}_i \end{aligned}$$

and the sequence for calculating $\partial^2 a_\Lambda / \partial \sigma_q \partial \sigma_p$ is

$$\begin{aligned} \frac{\partial^2 \boldsymbol{a}_L}{\partial \sigma_q \partial \sigma_p} &= \overline{\boldsymbol{\Phi}}''_L(z_L) \odot \frac{\partial \boldsymbol{z}_L}{\partial \sigma_q} \odot \frac{\partial \boldsymbol{z}_L}{\partial \sigma_p} + \overline{\boldsymbol{\Phi}}'_L(z_L) \odot \frac{\partial^2 \boldsymbol{z}_L}{\partial \sigma_q \partial \sigma_p} \\ \frac{\partial^2 \boldsymbol{z}_L}{\partial \sigma_q \partial \sigma_p} &= \frac{\partial^2 \boldsymbol{a}_{L-1}^T}{\partial \sigma_q \partial \sigma_p} \boldsymbol{W}_L \end{aligned} \tag{37}$$

Box.2: Evaluation algorithm for a_Λ, $\partial a_\Lambda/\partial a_\Lambda$ and $\partial^2 a_\Lambda/\partial\sigma_q\partial\sigma_p$

`Initialize:`

$$\boldsymbol{a}_0 = [\sigma_1, \sigma_2, \sigma_3]^T,\ \frac{\partial \boldsymbol{a}_0}{\partial \sigma_p} = \boldsymbol{e}_p,\ \frac{\partial^2 \boldsymbol{a}_0}{\partial \sigma_p \sigma_q} = \boldsymbol{0}$$

$$\boldsymbol{W}_L,\ L = 1, ..., \Lambda$$

`Iterate:`

`DO` $L = 1, \Lambda$

$$z_L = \boldsymbol{a}_{L-1}^T \boldsymbol{W}_L$$

$$\frac{\partial z_L}{\partial \sigma_p} = \frac{\partial \boldsymbol{a}_{L-1}^T}{\partial \sigma_p}\boldsymbol{W}_L,\ \frac{\partial^2 z_L}{\partial \sigma_q \partial \sigma_p} = \frac{\partial^2 \boldsymbol{a}_{L-1}^T}{\partial \sigma_q \partial \sigma_p}\boldsymbol{W}_L$$

$$\boldsymbol{a}_L = \overline{\boldsymbol{\Phi}}_L(z_L)$$

$$\frac{\partial \boldsymbol{a}_L}{\partial \sigma_p} = \overline{\boldsymbol{\Phi}}'_L(z_L) \odot \frac{\partial z_L}{\partial \sigma_p}$$

$$\frac{\partial^2 \boldsymbol{a}_L}{\partial \sigma_q \partial \sigma_p} = \overline{\boldsymbol{\Phi}}''_L(z_L) \odot \frac{\partial z_L}{\partial \sigma_q} \odot \frac{\partial z_L}{\partial \sigma_p} + \overline{\boldsymbol{\Phi}}'_L(z_L) \odot \frac{\partial^2 z_L}{\partial \sigma_q \partial \sigma_p}$$

`END DO`

The NN-specific part of the evaluation algorithm is summarized in Box-2. A few remarks are in order. First, each layer may have its own distinct activation function with the exception of the output layer, for which $\Phi_\Lambda(z) = z$, i.e., the identity function (for otherwise the output would be severely restricted in its modeling power (this can be easily seen on the example of Section-4, in the case of Hill'48 quadratic)).

Second, regarding an actual implementation. The algorithm in Box-2 applies to any densely connected architecture. The weight matrices $\boldsymbol{W}_L$ need not be constructed explicitly in code, the storage in a `Fortran`-style column vector of parameters being sufficient.

Third, when eq.(9) is used to obtain an exact orthotropic output, the definition in eq.(32) remains the same provided a_Λ is redefined as

$$a_\Lambda = \left(a_\Lambda^{(1)} + a_\Lambda^{(2)}\right)/2 \tag{38}$$

where $a_\Lambda^{(1)}$ is calculated with input $\boldsymbol{a}_0$ as in eq.(31), and $a_\Lambda^{(2)}$ with input $\boldsymbol{a}_0 = [\sigma_1, \sigma_2, -\sigma_3]^T$. In this case the initialization block in Box-2 becomes:

$$\boldsymbol{a}_0^{(1)} = [\sigma_1, \sigma_2, \sigma_3]^T,\ \frac{\partial \boldsymbol{a}_0^{(1)}}{\partial \sigma_p} = \boldsymbol{e}_p,\ \frac{\partial^2 \boldsymbol{a}_0^{(1)}}{\partial \sigma_p \sigma_q} = \boldsymbol{0}$$

$$\boldsymbol{a}_0^{(2)} = [\sigma_1, \sigma_2, -\sigma_3]^T,\ \frac{\partial \boldsymbol{a}_0^{(2)}}{\partial \sigma_p} = \boldsymbol{e}_p,\ p = 1, 2$$

$$\frac{\partial \boldsymbol{a}_0^{(2)}}{\partial \sigma_3} = -\boldsymbol{e}_3,\ \frac{\partial^2 \boldsymbol{a}_0^{(2)}}{\partial \sigma_p \sigma_q} = \boldsymbol{0}$$

$$\boldsymbol{W}_L,\ L = 1, ..., \Lambda$$

and the loop has to calculate both $a_\Lambda^{(1)}$ and $a_\Lambda^{(2)}$ and their gradients and hessians. The gradient and hessian formulas in eq.(33) and eq.(36) remain the same, provided a_Λ is as in eq.(38) and its derivatives are calculated by

$$\frac{\partial a_\Lambda}{\partial \sigma_p} = \frac{1}{2}\left(\frac{\partial a_\Lambda^{(1)}}{\partial \sigma_p} + \frac{\partial a_\Lambda^{(2)}}{\partial \sigma_p}\right)$$

$$\frac{\partial^2 a_\Lambda}{\partial \sigma_q \partial \sigma_p} = \frac{1}{2}\left(\frac{\partial^2 a_\Lambda^{(1)}}{\partial \sigma_q \partial \sigma_p} + \frac{\partial^2 a_\Lambda^{(2)}}{\partial \sigma_q \partial \sigma_p}\right)$$

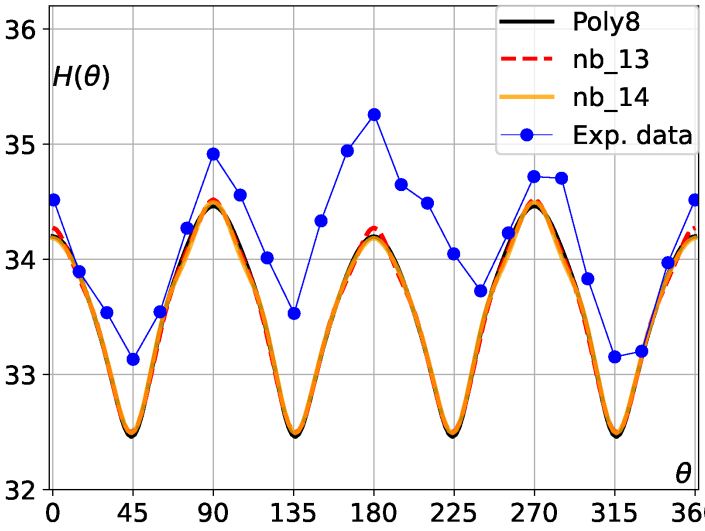

Figure 21: Profile predictions of `Poly8` and two NN-models, `nb13_fitPolyN` and `nb14_fitPolyN`.

The algorithm in Box-2 was implemented in a Abaqus-UMAT subroutine which was first tested/validated in the simulation of uniaxial tensile tests. Three models were compared, in three simulation batches, each batch consisting of seven simulations (of samples cut at angles 0^o, 15^o, ...,90^o from RD): `Poly8`, `nb13` and `nb14`. Typical mesh, deformed configuration and stress-strain curves are shown in Figs-31,32,33 in Appendix-C. The tested r-values (FE-simulation vs theoretical model) shown in Figs-34,35,36 in Appendix-C validate the correctness of the implementation. The cumulative run-time per simulation batch is reported in Table-4.

The algorithm in Box-2 was then applied to the simulation of cylindrical cup drawing. The geometry and simulation setting is the same as used in Soare and Diehl (2023) for the simulation of the `Poly8(TUAT)` model of AA6016T4, but here with a slightly denser mesh (see Fig-37 in Appendix-C) to allow for better run-times comparison (all simulation files - UMATs, input parameter files and python script for generating automatically the Abaqus simulation setup - are accessible in `partC` directory of the data and code repository referenced in the Data statement).

Three cup drawing simulations were performed. The first used the `Poly8` model of AA6016T4, to establish a comparison benchmark. The other two used the `nb13` and `nb14` models of the data sampled from the `Poly8` model (Section-5). The resulting cup height profiles are shown in Fig-21, where one can notice an almost complete overlap between the three simulations. This should be expected:

Figs-9,10 and Figs-18,19 in Appendix-C show how the two NN-models manage to almost clone (replicate) the `Poly8` model. In Table-4 are reported the simulation run-times. Also expected, based on its highly optimized implementation discussed in Soare and Diehl (2023): `Poly8` features the best run-time, followed closely by `nb14_fitPolyN`, which is essentially a FACET-model (Section-3) of degree 10 (and thus a `Poly10` model, which raises the question: Why not use a degree of 8 in `nb14_fitPolyN` to try to replicate the original `Poly8` model ? The answer is in the particular form of FACET - a sum of convex terms - whereas not every convex `Poly8` model accepts such a representation. Hence, in general, higher order FACET-models are required to replicate lower order `PolyN`-models.). Overall, the run-times of the two NN-models are not too far from that of `Poly8`, indicating that small-sized shallow networks are viable modeling and simulation tools. Code optimization on a case-specific basis is expected to further reduce the run-time featured by NN-models (for example, for networks with activation functions defined by integer exponents one can replace the quite expensive calls to the power function with a smaller number of direct multiplications).

Table 4: Run-times (in seconds) of uniaxial tensile testing and cup-drawing simulations.

	Wallclock	
Model	Uniaxial	Cup-Drawing
Poly8	4010	7971
`nb14_fitPolyN`	4834	8897
`nb13_fitPolyN`	5353	9871

References

Aretz, H., Barlat, F., 2004. General orthotropic yield functions based on linear stress deviator transformations. Proc. NUMIFORM 2004 Conference, edited by S. Ghosh, J.K. Lee and J.C. Castro, AIP Conf. Proc. 712, pp. 147-156.

Aretz, H., Barlat, F., 2013. New convex yield functions for orthotropic metal plasticity. Int. J. Non-Linear Mechanics, 51, 97-111.

Atkinson, K., Han, W., 2012. Spherical harmonics and approximations on the unit shpere: An introduction. Springer-Verlag.

Bonatti, C., Mohr, D., 2022. On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids. J. Mech and Physics of Solids, 158, 104697.

Calin, O., 2020. Deep Learning Architectures: A Mathematical Approach. Springer Publishing Company.

Chollet, F., 2021. Deep Learning with Python (2nd edition). Manning Publications.

Clevert, D-A, Unterthiner, T., Hochreiter, S., 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). ICLR (Poster) 2016. (arXiv:1511.07289)

Courant, R., Hilbert, D., 1953. Methods of mathematical physics. Interscience publishers, NY. Vol-I.

Esmaeili, A., Asadi, S., Larsson, F., Runesson, K., 2019. Construction of macroscale yield surfaces for ductile composites based on a virtual testing strategy. European J. of Mechanics -A/Solids, 77, 103786.

Fuhg, J.N., Bouklas, N., Jones, R.E., 2022. Learning hyperelastic anisotropy from data via a tensor basis neural network. J Mech Phys Solids, 16:105022.

Fuhg, J.N., Fau, A., Bouklas, N., Marino, M., 2023. Enhancing phenomenological yield functions with data: Challenges and opportunities. European J. of Mechanics - A/Solids, 99, 104925.

Géron, A., 2022. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd edition). O'Reilly Media, Inc.

Groemmer, H., 1996. Geometric applications of Fourier series and spherical harmonics. Cambridge University Press.

Habraken, A.M, Aksen, T.A., Alves, J.L., Amaral, R.L., Betaieb, E., Chandola, N., Corallo, L., Cruz, D.J., Duchêne, L., Engel, B., Esener, E., Firat, M., Sörensen, P.F., Lopez, J. G., Ghiabakloo, H., Kestens, A.I, Lian, J., Lingam, R., Liu, W., Ma, J., Menezes, L.F., Minh, T.N., Miranda, S.S., Neto, D.M., Pereira, A.F.G., Prates, P.A., Reuter, J., Baudard, B.R., Ulloa, C.R., Sener, B., Shen, F., Van Bael, A., Verleysen, P., Barlat, F., Cazacu, O., Kuwabara, T., Lopes, A., Oliveira, M.C., Santos, A.D, Vincze, G., 2022. Analysis of ESAFORM 2021 cup drawing benchmark of an Al alloy, critical factors for accuracy and efficiency of FE simulations. Int. J. of Material Forming, 15, 61.

Harsch, D., Heingärtner, J., Hortig, D., Hora, P., 2016. Virtual tryout planning in automotive industry based on simulation metamodels. IOP Conf. Series: Materials Science and Engineering 159 (2016) 012007.

Hartmaier, A., 2020. Data-Oriented Constitutive Modeling of Plasticity in Metals. Materials, 13, 1600.

Hill, R., 1948. A theory of the yielding and plastic flow of anisotropic metals. Proc. R. Soc. London Ser. A 193A 1033:281–297.

Hill, R., 1950. The mathematical theory of plasticity. Clarendon Press, Oxford, UK.

Hill, R., 1994. Classical plasticity: A retrospective view and a new proposal. J. Mech and Physics of Solids, 42, 1803-1816.

Hill, R., Hutchinson, J.W., 1992. Differential hardening in sheet metal under biaxial loading: A theoretical framework. J. Applied Mechanics, 59(2S), S1-S9.

James, G., Witten, D., Hastie, T., Tibshirani, R., Taylor, J., 2023. An introduction to statistical learning with applications in Python. Springer Nature Switzerland AG. Ch-2.

Kalina, K.A., Linden, L., Brummund, J., Kästner, M, 2023. FE^{ANN} : an efficient data-driven multiscale approach based on physics-constrained neural networks and automated data mining. Computational Mechanics, 71:827–851.

Klein, D.K., Fernández, M., Martin, R.J., Neff, P., Weeger, O. 2021. Poly-convex anisotropic hyperelasticity with neural networks. J. Mech. Phys. Solids, 159, 104703.

Kraska, M., Doig, M., Tikhomirov, D., Raabe, D., Roters, F., 2009. Virtual material testing for stamping simulations based on polycrystal plasticity Computational Materials Science, 46, 383-392.

Lou, X.Y., Li, M., Boger, R.K., Agnew, S.R., Wagoner, R.H., 2007. Hardening evolution of AZ31B Mg sheet. International J. of Plasticity, 23, 44-86.

Marsaglia, G., 1972. Choosing a point from the surface of a sphere. Ann. Math. Stat. 43, 645-646.

Mozafar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., Bessa, M.A., 2019. Deep learning predicts path-dependent plasticity. PNAS, 116, no. 52.

Muller, M. E, 1959. A note on a method for generating points uniformly on N-dimensional spheres. Comm. Assoc. Comput. Mach. 2, 19-20.

Nascimento, A., Roongta, S., Diehl, M., Beyerlein, I.J., 2023. A machine learning model to predict yield surfaces from crystal plasticity simulations. Int. J. Plasticity, 161, 103507.

Pereira, A.F.G., Ruivo, M.F., Oliveira, M.C., Fernandes, J.V., Prates, V.A, 2021. Numerical study of the square cup stamping process: a stochastic analysis. Esaform-2021 Proc., esaform21.2158.

Plunkett, B., Cazacu, O., Barlat, F., 2008. Orthotropic yield criteria for description of the anisotropy in tension and compression of sheet metals. Int. J. of Plasticity, 24, 847-866.

Raemy, C., Manopulo, N., Hora, P., 2017. On the modelling of plastic anisotropy, asymmetry and directional hardening of commercially pure titanium: A planar Fourier series based approach. Int. J. of Plasticity, 91, 182-204.

Rockafellar, R.T., 1970. Convex analysis. Princeton University Press, Princeton, N.J., U.S.A.

Roters, F., Diehl, M., Shanthraj, P., Eisenlohr, P., Reuber, C., Wong, S.L., Maiti, T., Ebrahimi, A., Hochrainer, T., Fabritius, H.-O., Nikolov, S., Friák, M., Fujita, N., Grilli, N., Janssens, K.G.F., Jia, N., Kok, P.J.J, Ma, D., Meier, F., Werner, E., Stricker, M., Weygand, D., Raabe, D., 2019. DAMASK - The Düsseldorf Advanced Material Simulation Kit for modeling multi-physics crystal plasticity, thermal, and damage phenomena from the single crystal up to the component scale Computational Materials Science, 158, 420-478.

Sedighiani, K., Diehl, M., Traka, K., Roters, F., Sietsma, J., Raabe, D., 2020. An efficient and robust approach to determine material parameters of crystal plasticity constitutive laws from macro-scale stress-strain curves. Int. J. Plasticity, 134, 102779.

Soare, S.C., Yoon, J.-W., Cazacu, O., 2008. On the use of homogeneous polynomials to develop anisotropic yield functions with applications to sheet forming. Int. J. of Plasticity, 24, 915-944.

Soare, S.C., Barlat, F., 2010. Convex polynomial yield functions. J. of the Mechanics and Physics of Solids, 58, 1804-1818.

Soare, S.C., Barlat, F., 2011. A study of the Yld2004 yield function and one extension in polynomial form: A new implementation algorithm, modeling range, and earing predictions for aluminum alloy sheets. European J. of Mechanics - A/Solids, 30, 807-819.

Soare, S.C., 2023. Bezier5YS and SHYqp: A general framework for generating data and for modeling symmetric and asymmetric orthotropic yield surfaces. European J. of Mechanics - A/Solids, 97, 104781.

Soare, S.C., Diehl, M., 2023. Calibration and fast evaluation algorithms for homogeneous orthotropic polynomial yield functions. Computational Mechanics, DOI: 10.1007/s00466-023-02408-6.

Strang, G., 2019. Linear Algebra and Learning from Data. Wellesley-Cambridge Press.

Thorpe, J.A., 1979. Elementary topics in differential geometry. Springer-Verlag, New York.

Tian, H., Brownell, B., Baral, M., Korkolis, Y.P., 2017. Earing in cup-drawing of anisotropic Al-6022-T4 sheets. Int. J. Material Forming, 10, 329–343.

Tong, W., Alharbi, M., 2017. Comparative evaluation of non-associated quadratic and associated quartic plasticity models for orthotropic sheet metals. Int. J. Solids and Structures, 128, 133-148.

Tong, W., 2018. On the certification of positive and convex Gotoh's fourth-order yield function. NUMISHEET-2018 Proceedings in J. Phys.: Conf. Ser. 1063 012093.

Van Houtte, P., Yerra, S.K., Van Bael, A., 2009. The Facet method: A hierarchical multilevel modelling scheme for anisotropic convex plastic potentials. Int. J. Plasticity, 25, 332-360.

Varjú, P.P., 2007. Approximation by homogeneous polynomials. Constructive Approximation, 26, 317-337.

Vegter, H., Boogaard, A.H., 2006. A plane stress yield function for anisotropic sheet material by interpolation of biaxial stress states. Int. J. of Plasticity, 22, 557-580.

Vlassis, N.N., Sun, W.C., 2021. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. Computer Methods in Applied Mechanics and Engineering, 377, 113695.

On the use of neural networks in the modeling of yield surfaces: Appendix-C

Stefan C. Soare

Contents

Some results from notebooks and simulations referenced in the main article are gathered here for the convenience of the reader as follows:

part-A: Section-4

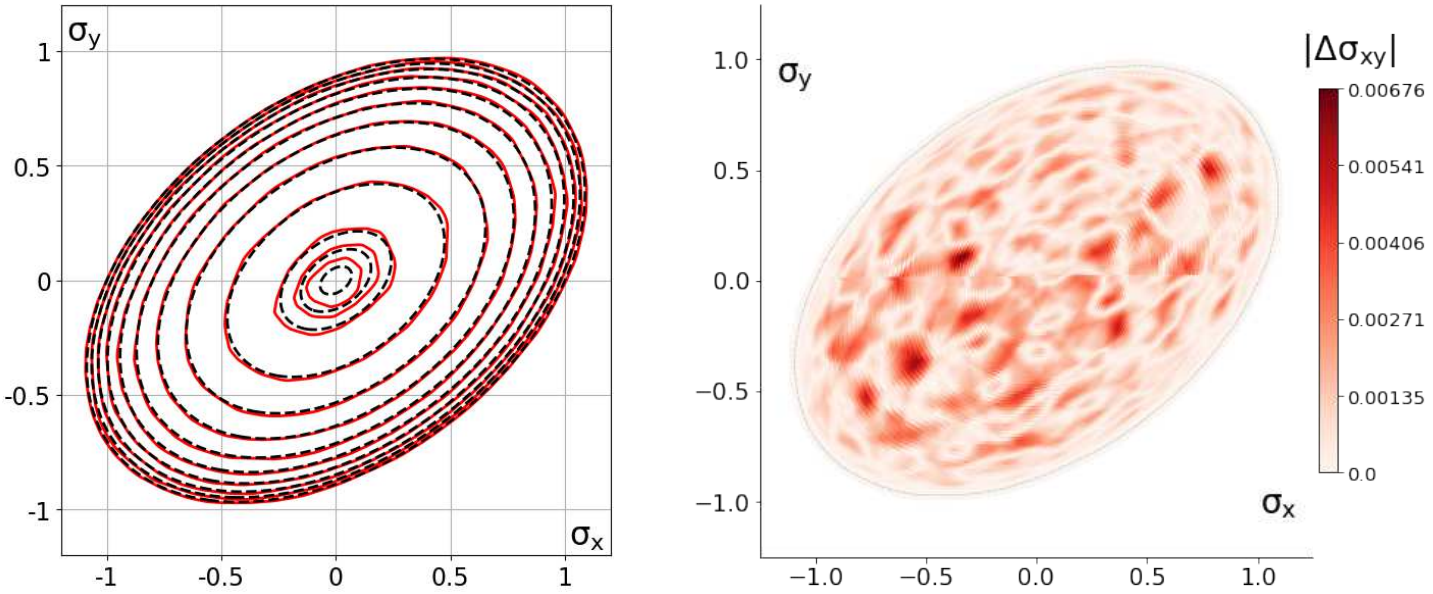

Figure.1: (nb00B) Left: $\sigma_{xy} = constant$ sections of a ReLU-based NN-model (red) of Hill'48 data (black). Right: Verification of the orthotropic symmetry of the NN-model.

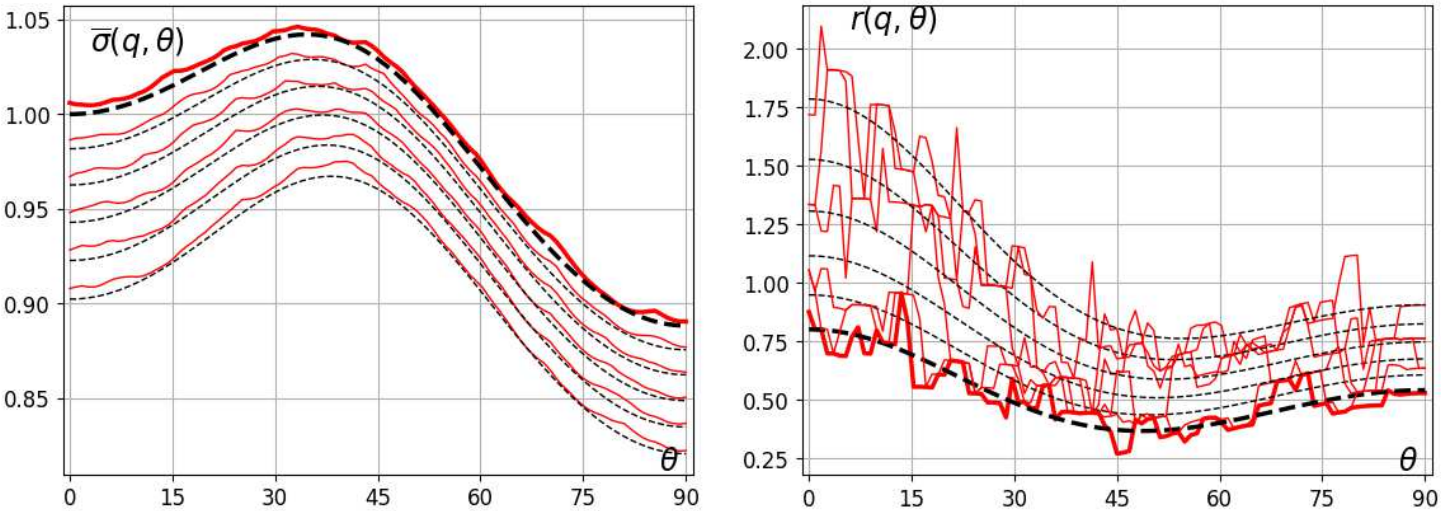

Figure.2: (nb00B) Predictions of the ReLU-based NN-model (red) of Hill'48 data (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

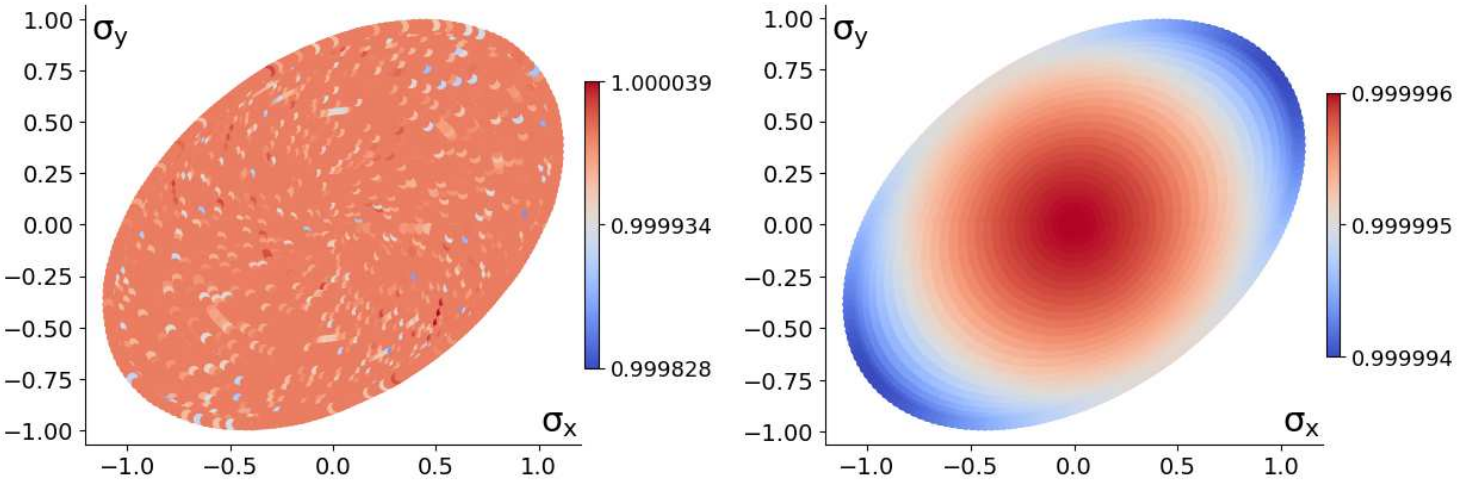

Figure.3: (nb00B) Verification of convexity via Jensen inequality, eq.(11): of the ReLU-based NN-model of Hill'48 data (Left) and of Hill'48 analytical model (Right). Values > 1 indicate deviations from convexity (see the discussion related to eq.(11) in the main article).

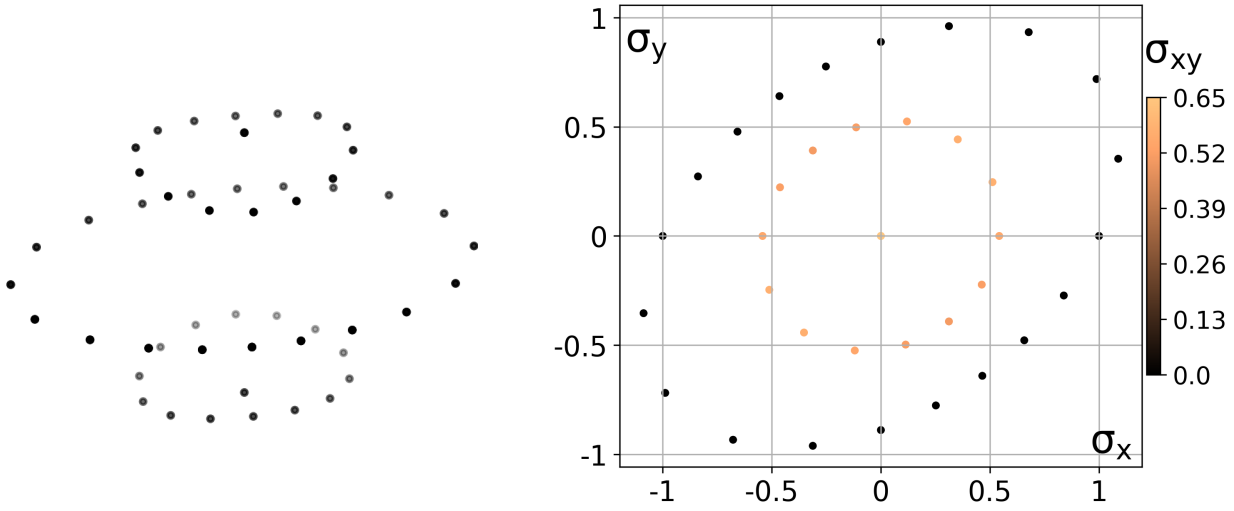

Figure.4: Left: Cloud of 72 samples extracted from a Hill'48 model of AA6022-T4 (Left) and its top view with corresponding numerical map (Right). Used in notebooks **nb02***.

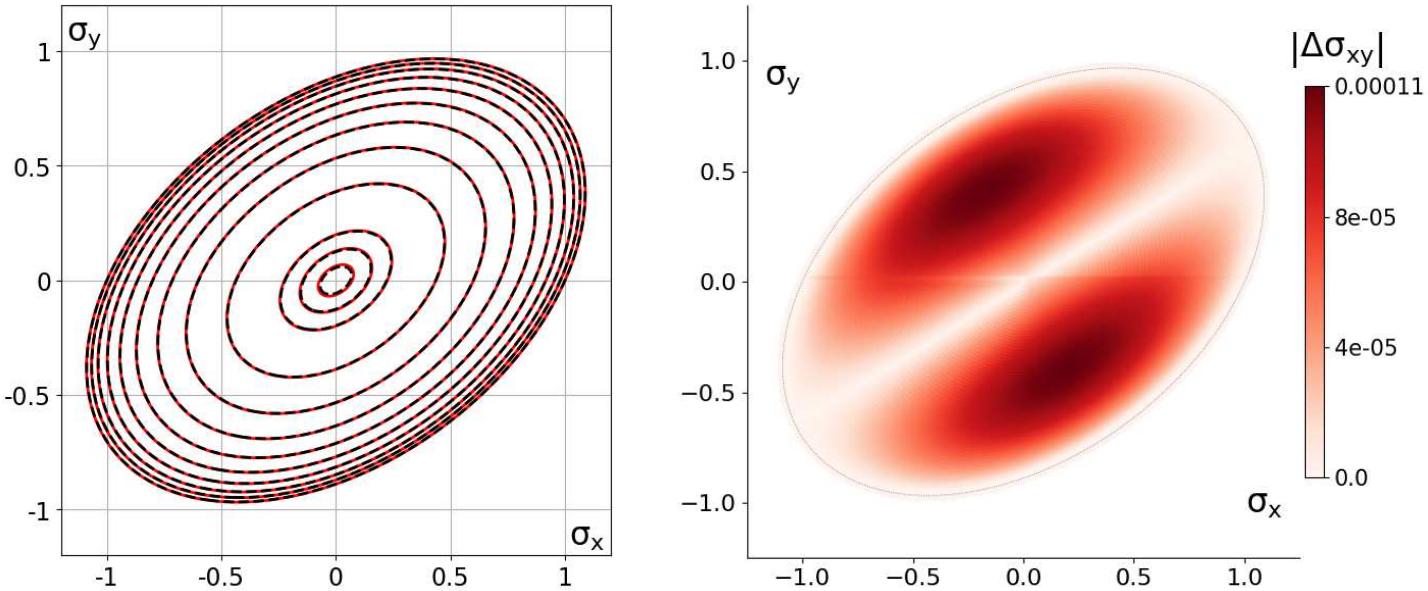

Figure.5: (nb02) Left: $\sigma_{xy} = constant$ sections of a ReLU-based NN-model (red) of Hill'48 data (black). Right: Verification of the orthotropic symmetry of the NN-model.

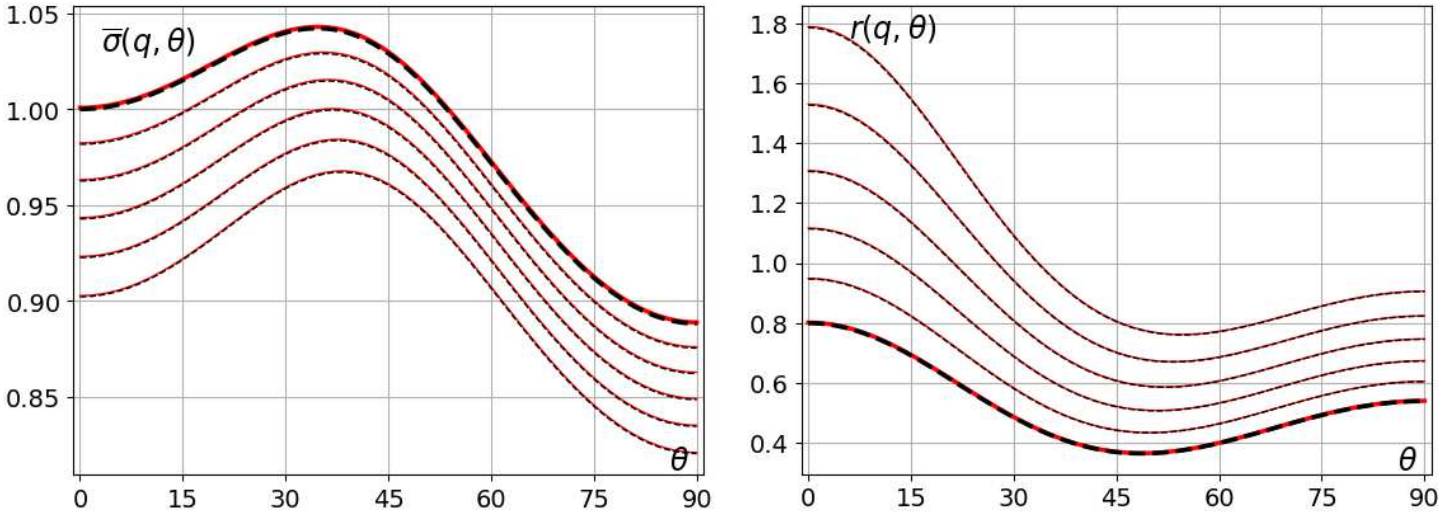

Figure.6: (nb02) Predictions of the ReLU-based NN-model (red) of Hill'48 data (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

The output of the NN-models in Section-4.2 is in fact the quadratic:

$$f(\boldsymbol{\sigma}) = a_1\sigma_x^2 + a_2\sigma_x\sigma_y + a_3\sigma_y^2 + a_4\sigma_{xy}^2 + a_5\sigma_x\sigma_{xy} + a_6\sigma_y\sigma_{xy}$$

For the NN-model in notebooks `nb02_*`, the corresponding parameters are as follow:

$$\begin{aligned}
a_1 &= 0.9985243696719408\\
a_2 &= -0.8886803835630417\\
a_3 &= 1.2656582449562848\\
a_4 &= 2.3961169943213463\\
a_5 &= 0.00046975910663604736\\
a_6 &= -0.00038280710577964783
\end{aligned}$$

Parameters a_5 and a_6 are close to zero, and hence the model is quasi-orthotropic, as illustrated also in Fig.5-Right.

part-A: Section-5

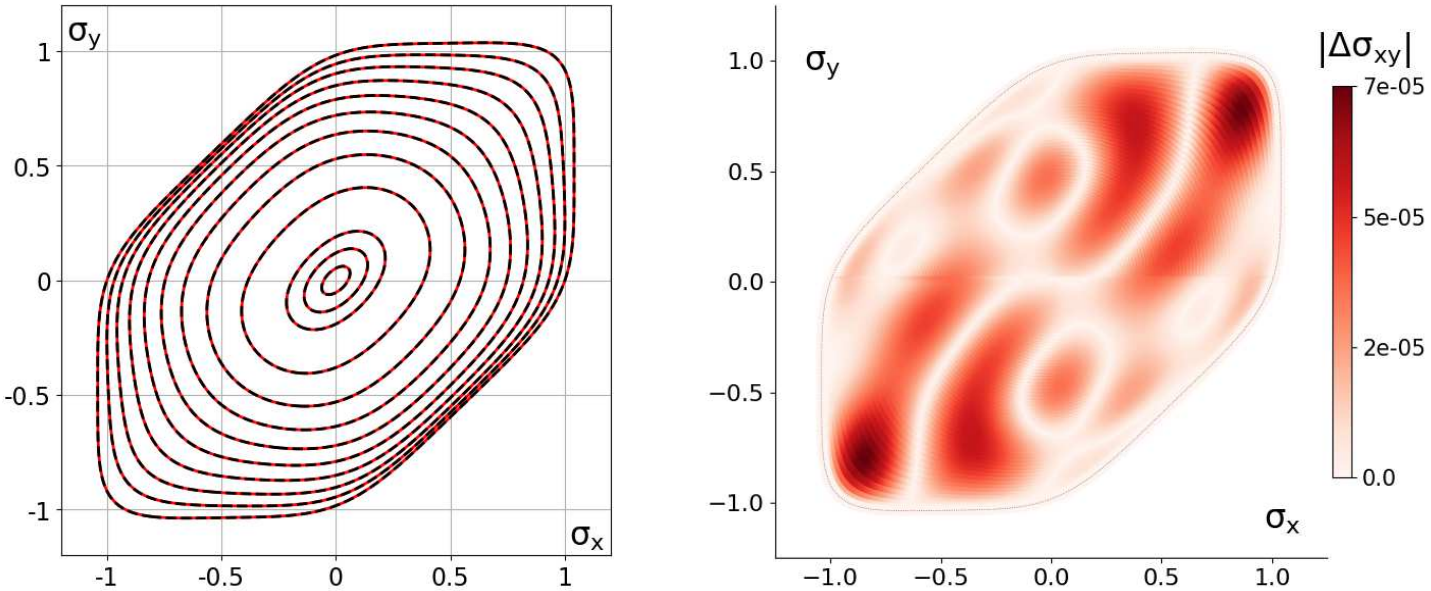

Figure.7: (nb07) Left: σ_{xy} = *constant* sections of the NN (red) and Poly8 (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

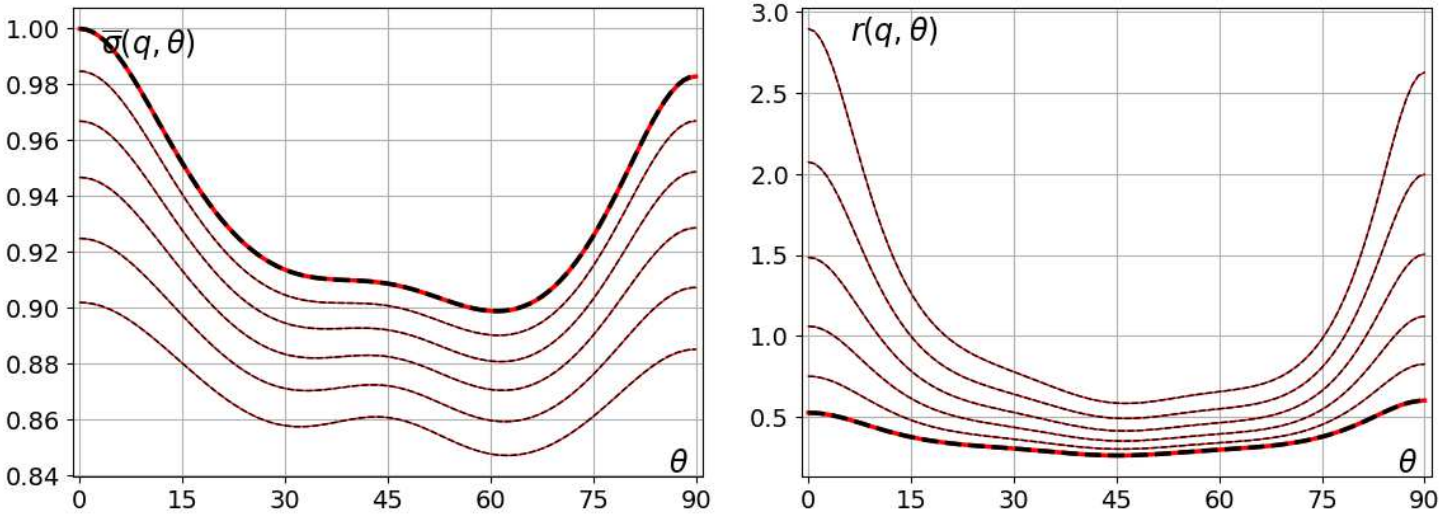

Figure.8: (nb07) Predictions of the NN-model (red) and Poly8 (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

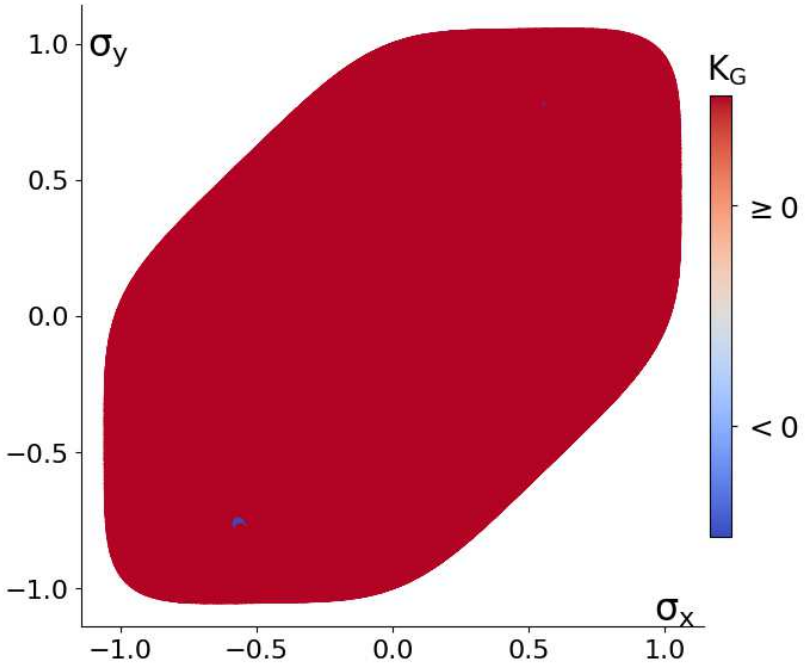

Figure.9: (nb07) Verification of the convexity of the NN-model. This model is almost convex: The two tiny blue patches (lower-left and upper-right corners) feature negative minimum Gauss curvatures $K_G \approx -0.0008$

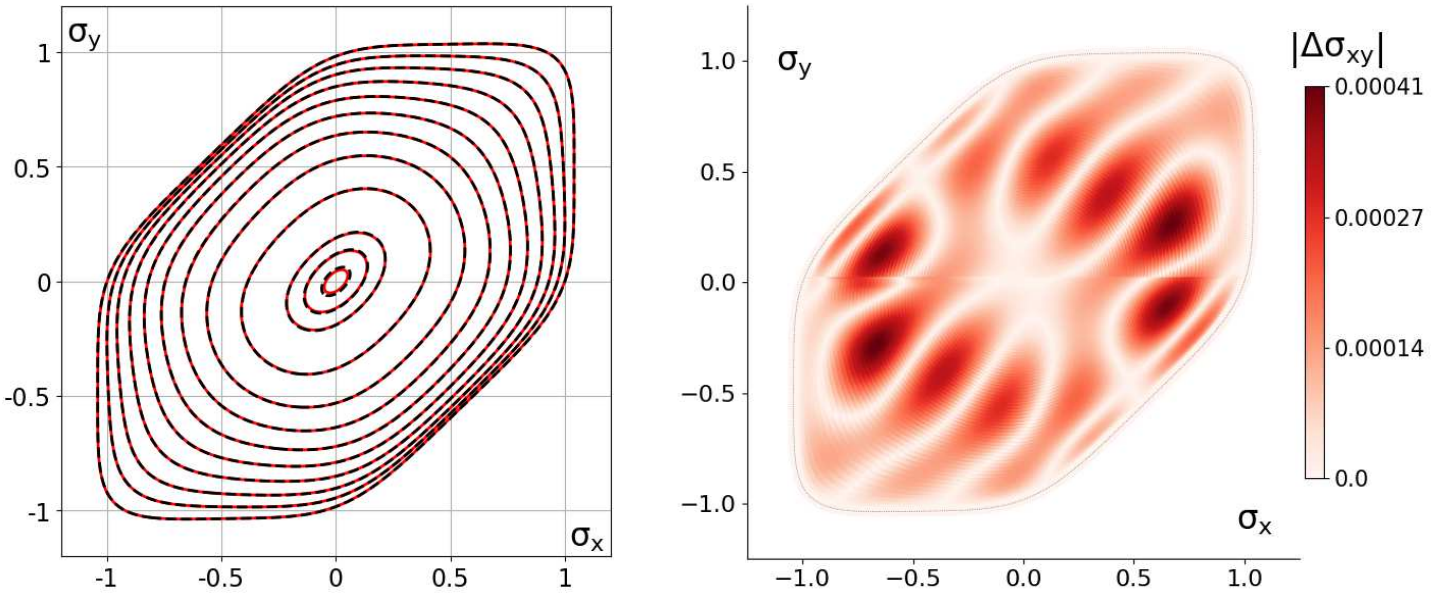

Figure.10: (nb07B) Left: $\sigma_{xy} = constant$ sections of the NN (red) and Poly8 (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

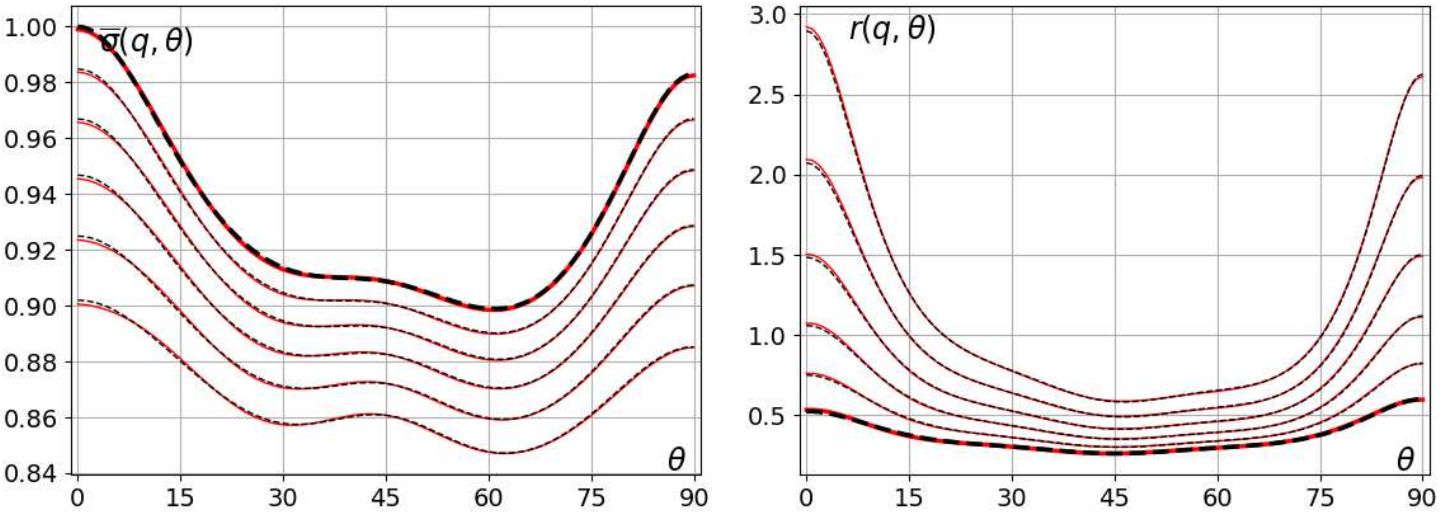

Figure.11: (nb07B) Predictions of the NN-model (red) and Poly8 (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

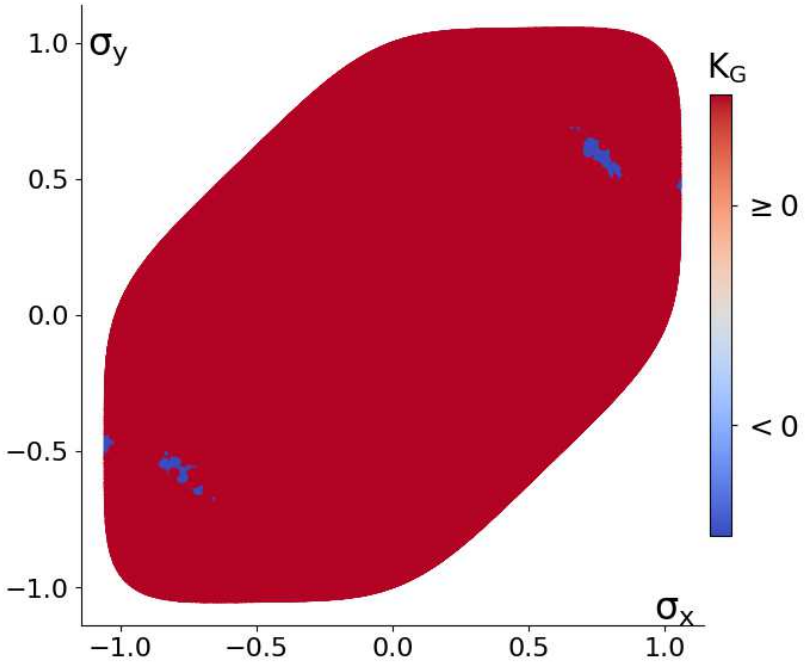

Figure.12: (nb07B) Verification of the convexity of the NN-model. The two blue patches (lower-left and upper-right corners) feature negative minimum Gauss curvatures $K_G \approx -0.0653$

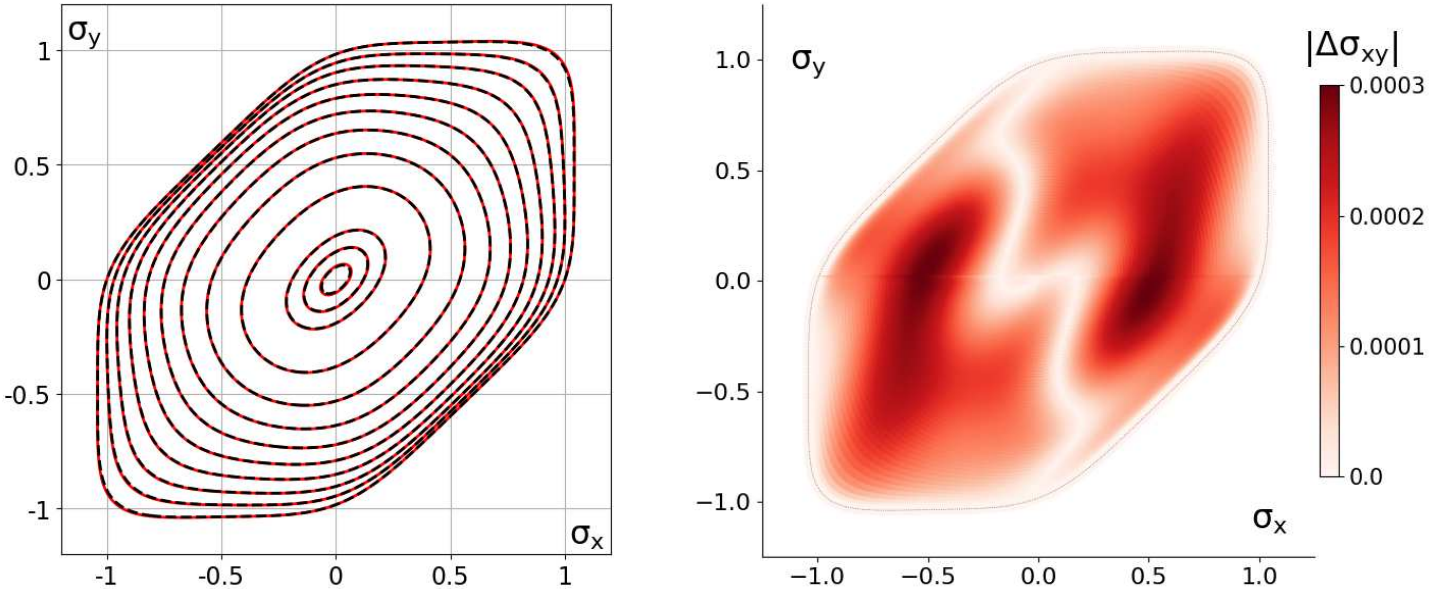

Figure.13: (nb08) Left: σ_{xy} = *constant* sections of the NN (red) and Poly8 (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

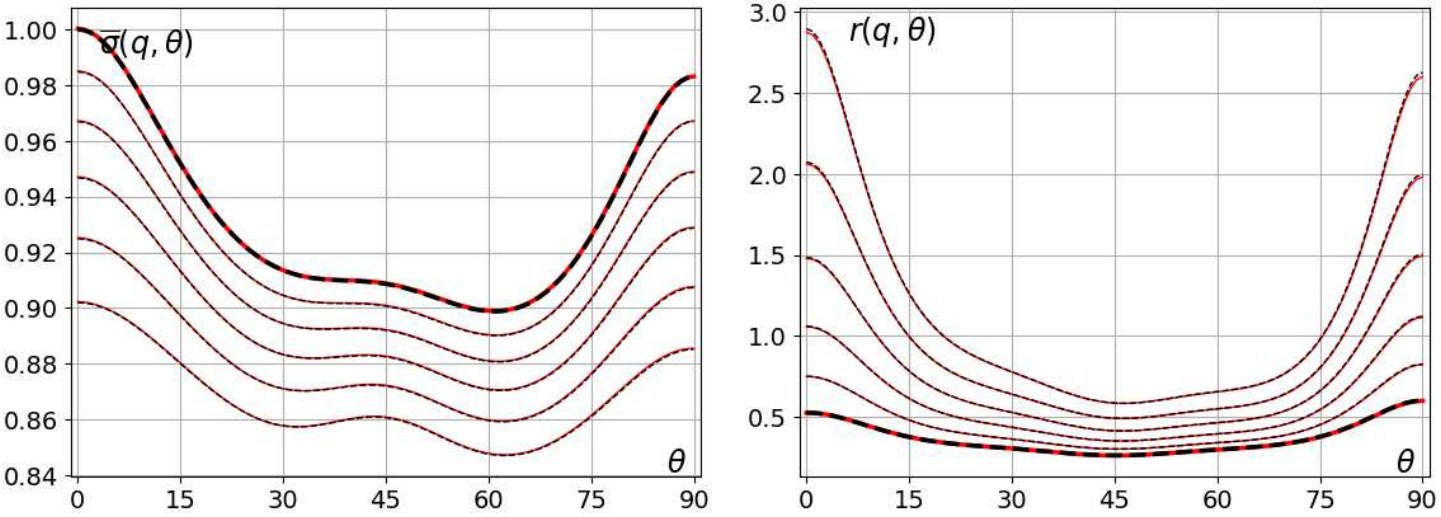

Figure.14: (nb08) Predictions of the NN-model (red) and Poly8 (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

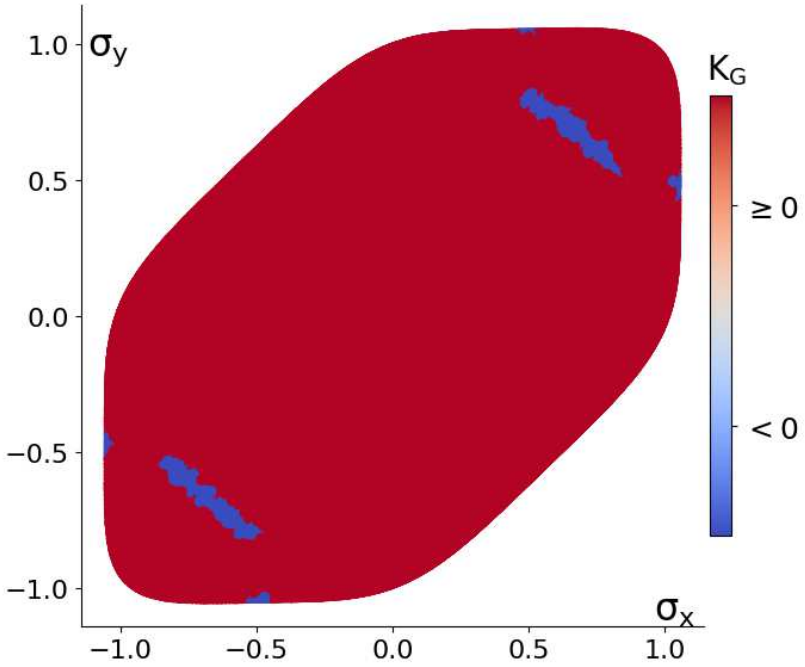

Figure.15: (nb08) Verification of the convexity of the NN-model. The two blue patches (lower-left and upper-right corners) feature negative minimum Gauss curvatures $K_G \approx -0.145$

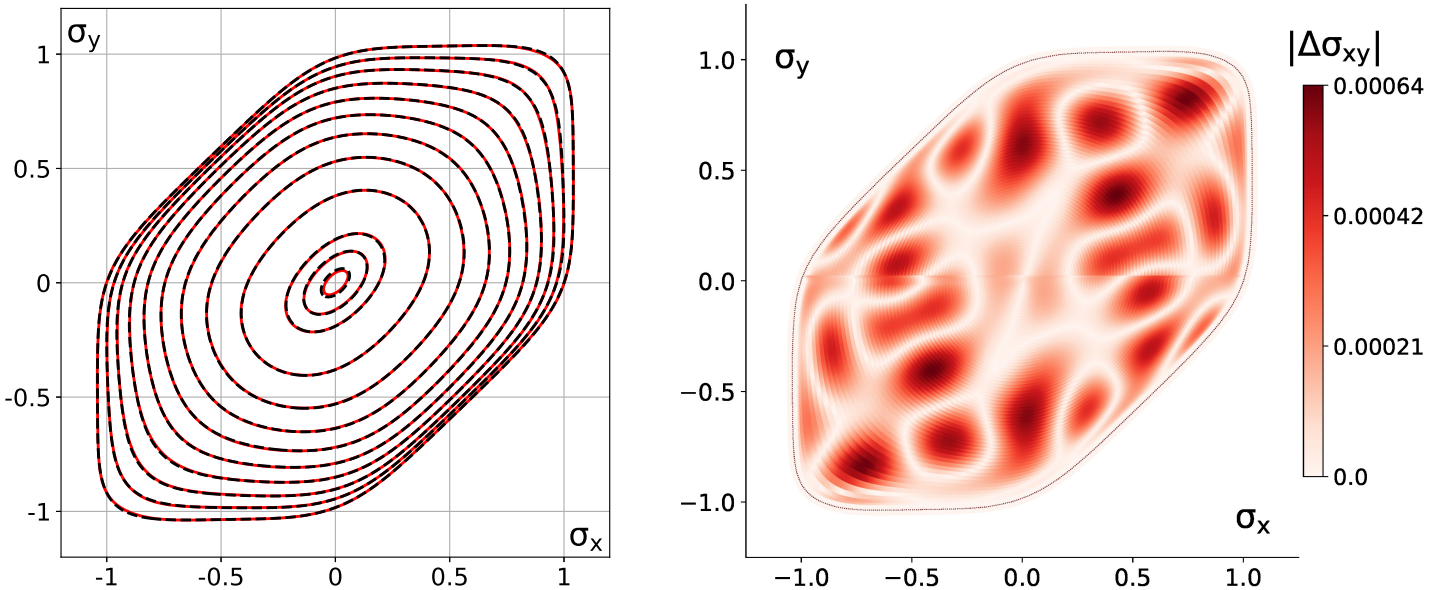

Figure.16: (nb10) Left: σ_{xy} = *constant* sections of the NN (red) and Poly8 (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

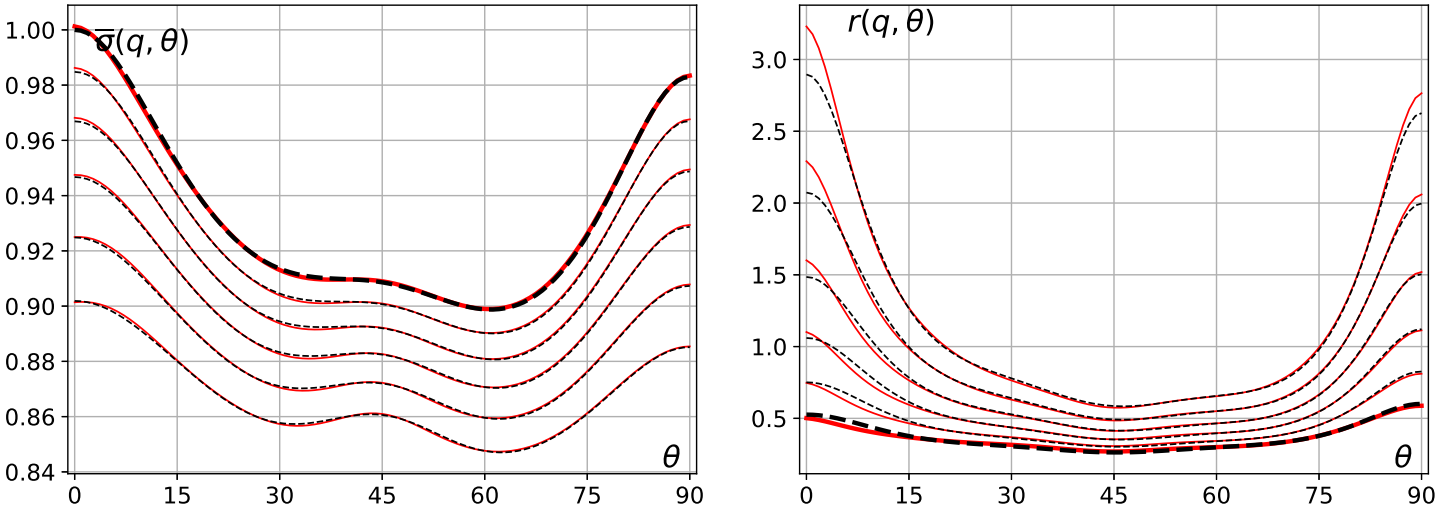

Figure.17: (nb10) Predictions of the NN-model (red) and Poly8 (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

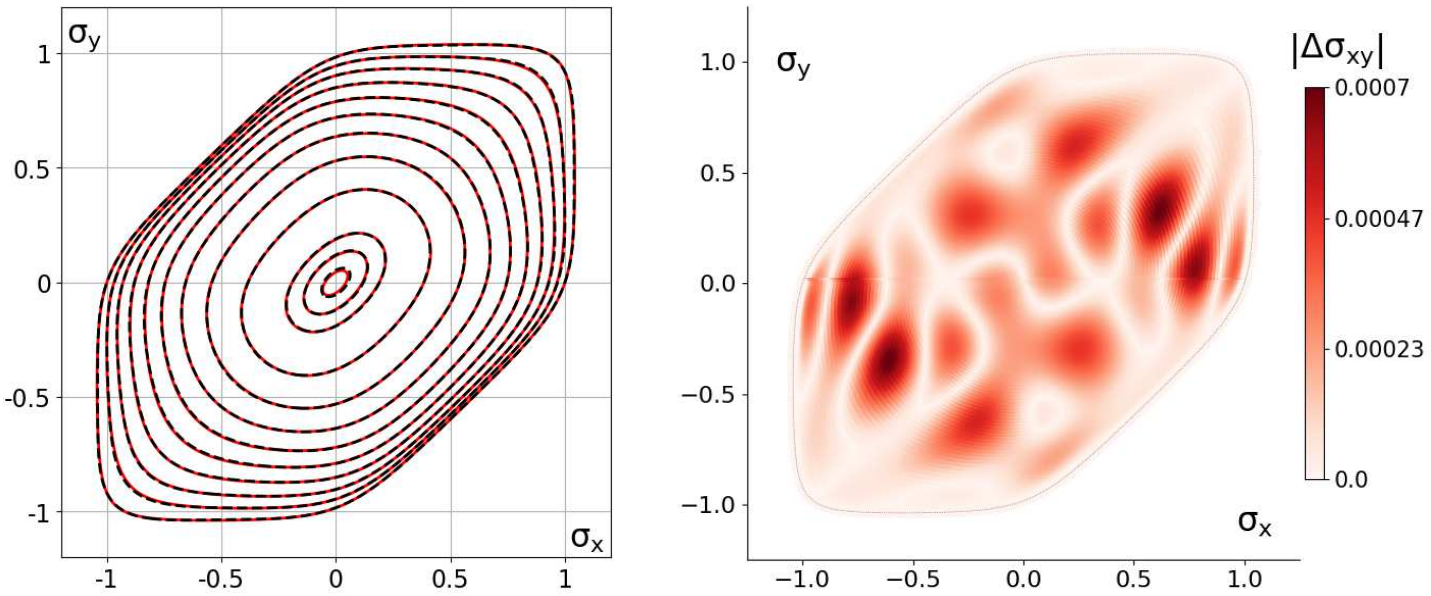

Figure.18: (nb13) Left: $\sigma_{xy} = constant$ sections of the NN (red) and Poly8 (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

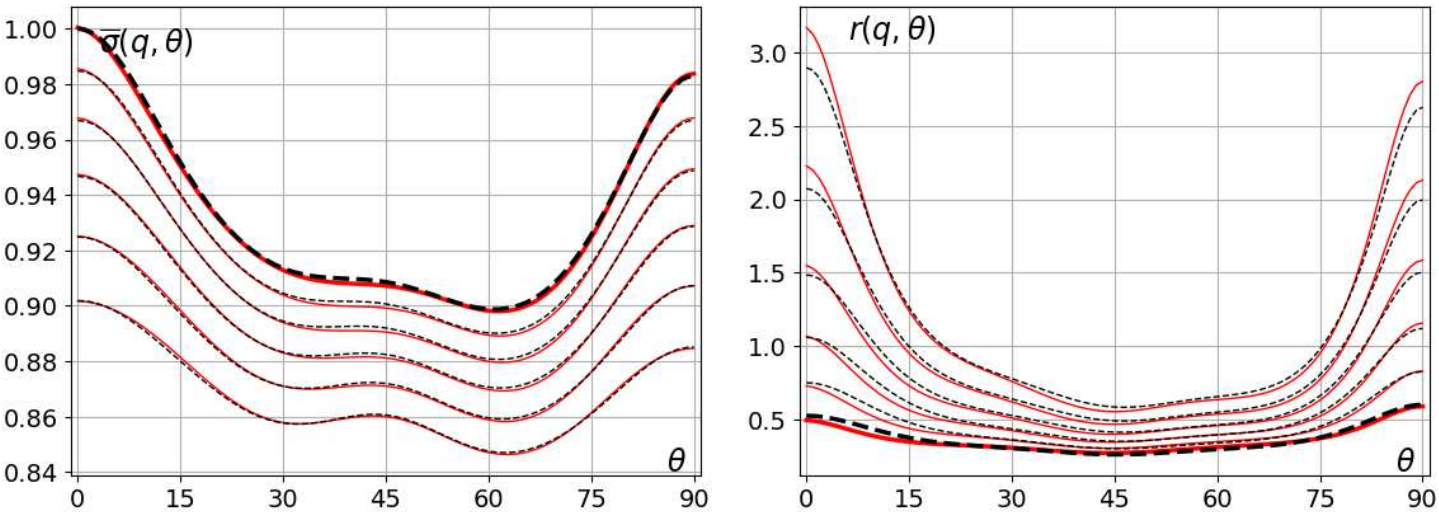

Figure.19: (nb13) Predictions of the NN-model (red) and Poly8 (black): Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

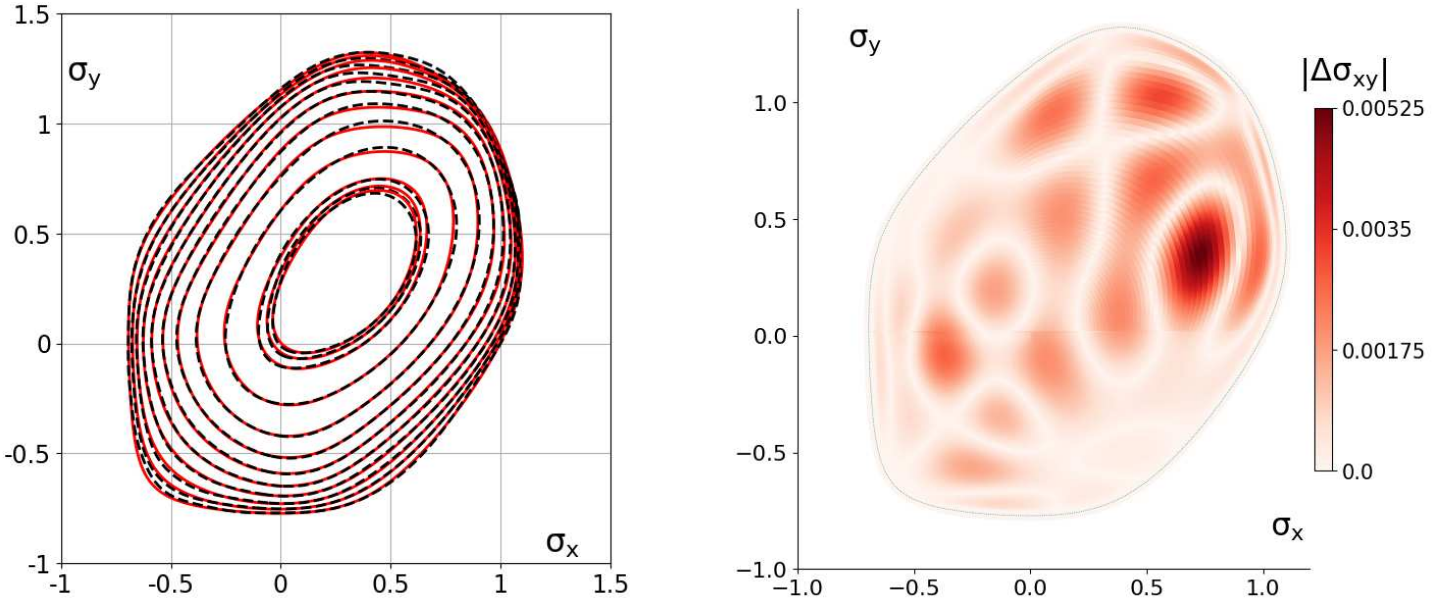

Figure.20: (nb16) Left: $\sigma_{xy} = constant$ sections of the NN (red) and SHYqp (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

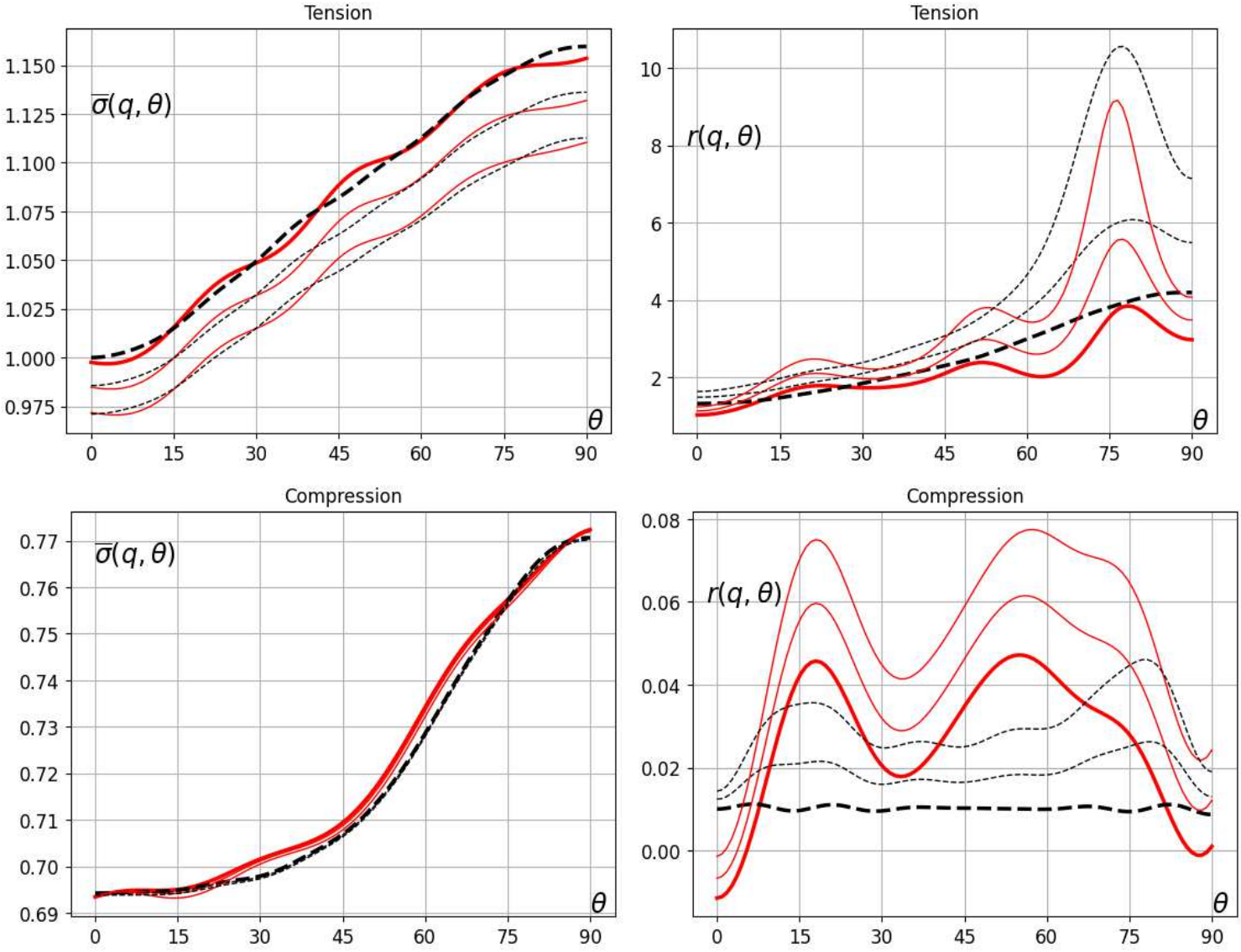

Figure.21: (nb16) Predictions of the NN-model (red) and SHYqp (black) for compression tests: Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

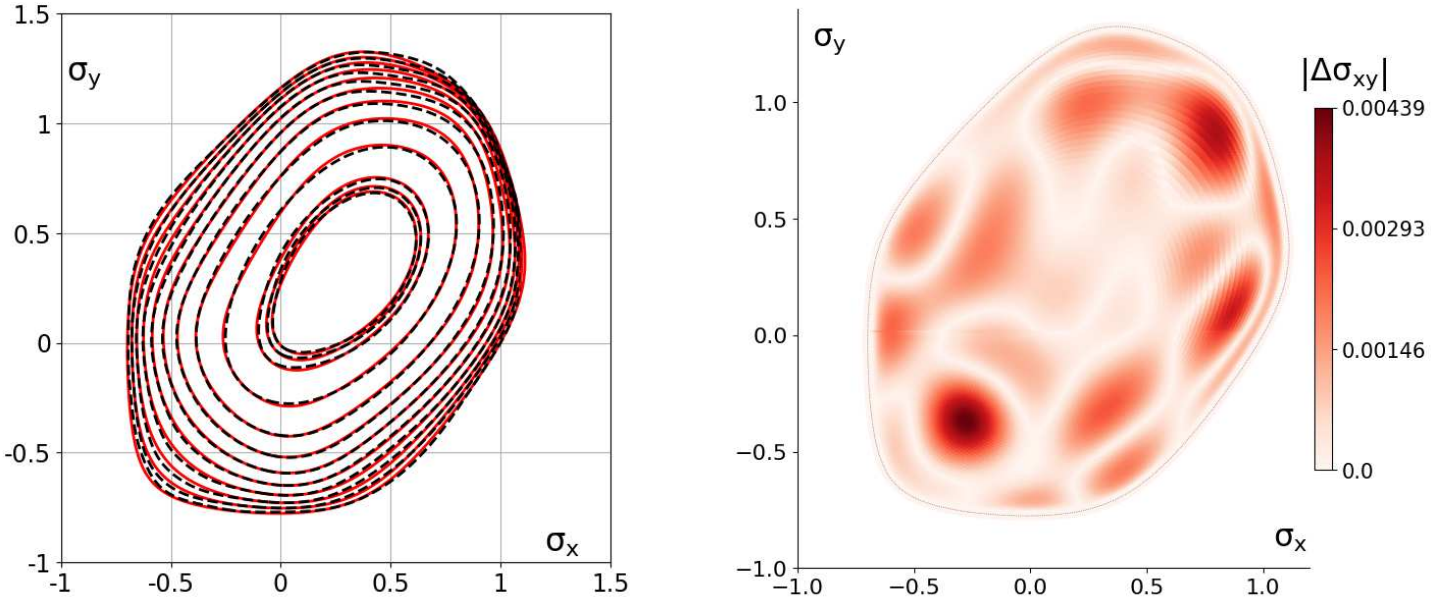

Figure.22: (nb17) Left: $\sigma_{xy} = constant$ sections of the NN (red) and SHYqp (black) models. Right: Verification of the orthotropic symmetry of the NN-model.

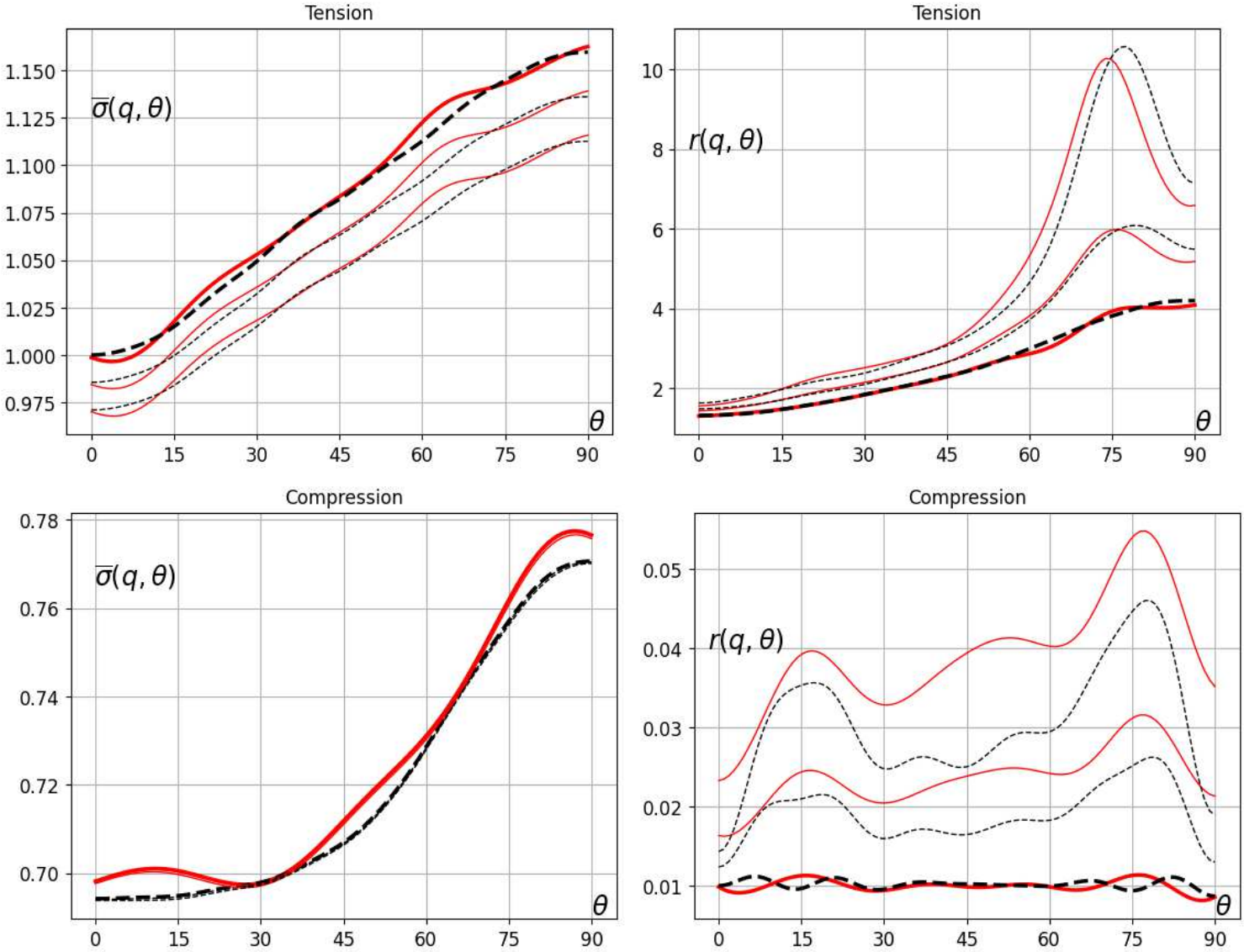

Figure.23: (nb17) Predictions of the NN-model (red) and SHYqp (black) for compression tests: Directional stresses (Left) and r-values (Right). Thick lines correspond to uniaxial ($q = 0$).

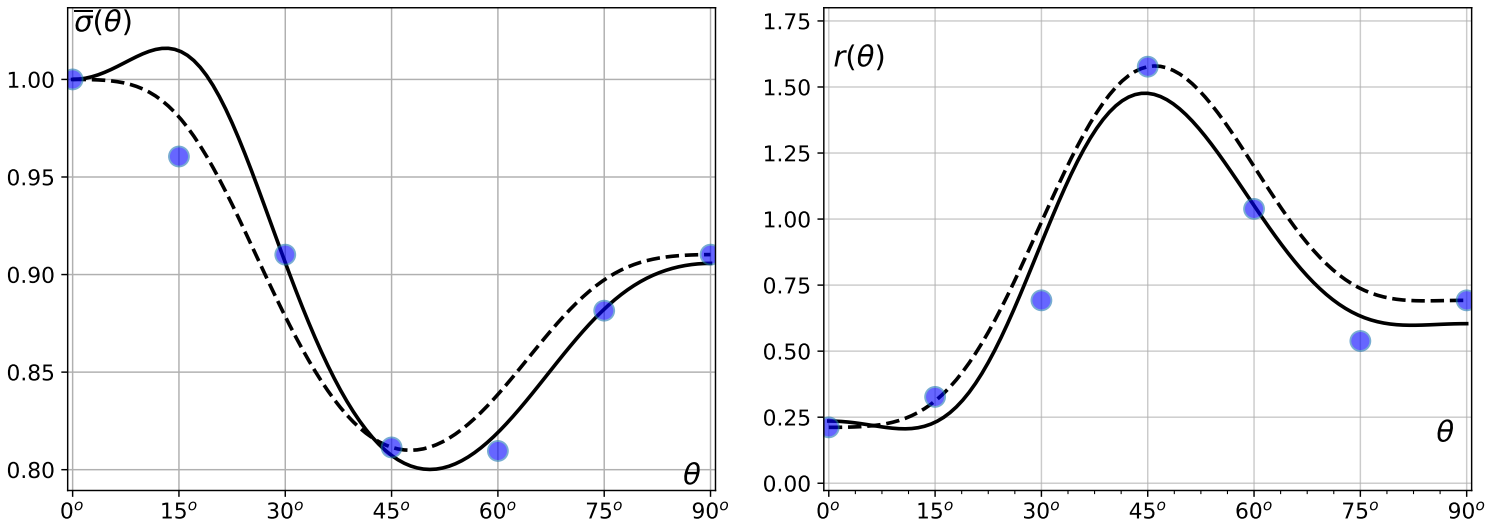

Figure.24: Directional yield stress (Left) and r-values (Right) of the Poly4 model of AA2090-T3 featured by solution **ID#4** in Table-2 of article shown with solid line (See Section-7.3 of the article for explanations on the nature of these solutions). Comparison with the customized Poly4 model row ID#3 (dashed). The blue markers show the experimental data.

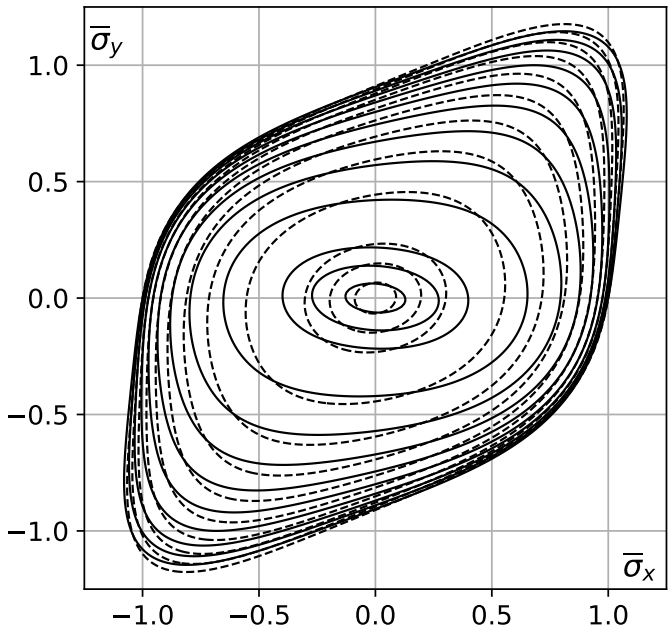

Figure.25: $\sigma_{xy} = const$ sections through the yield surfaces of the two Poly4 models shown in Fig.24: **ID#4**-solid, ID#3-dashed.

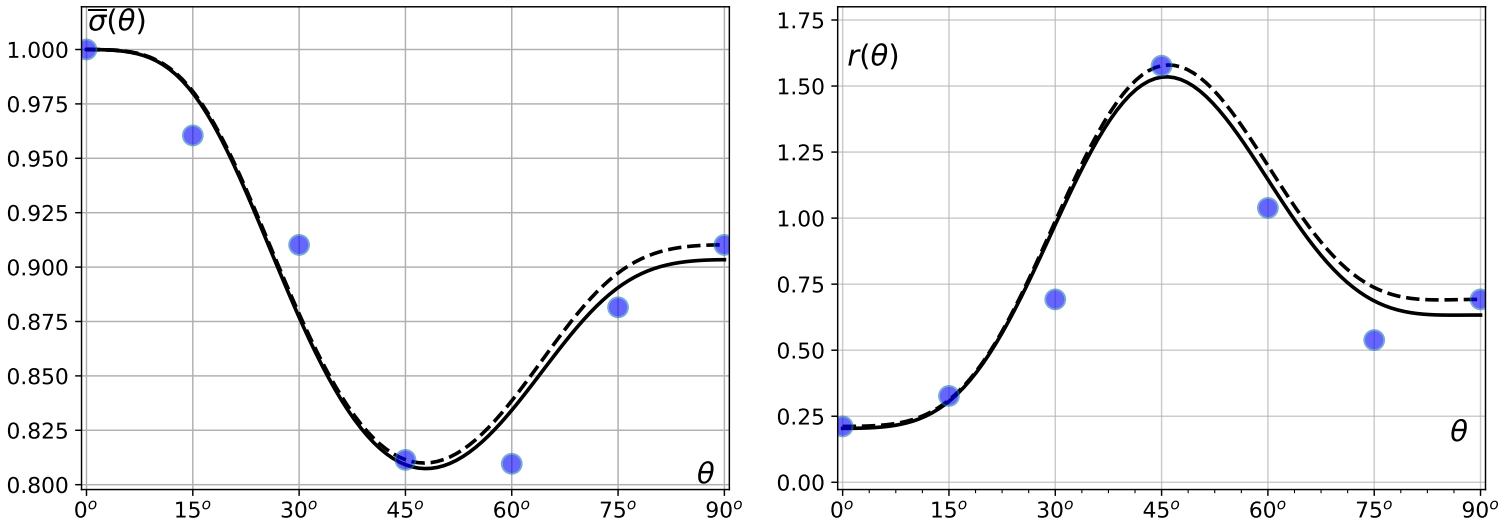

Figure.26: Directional yield stress (Left) and r-values (Right) of the Poly4 model of AA2090-T3 featured by solution **ID#5** in Table-2 of article shown with solid line (See Section-7.3 of the article for explanations on the nature of these solutions). Comparison with the customized Poly4 model row ID#3 (dashed). The blue markers show the experimental data.

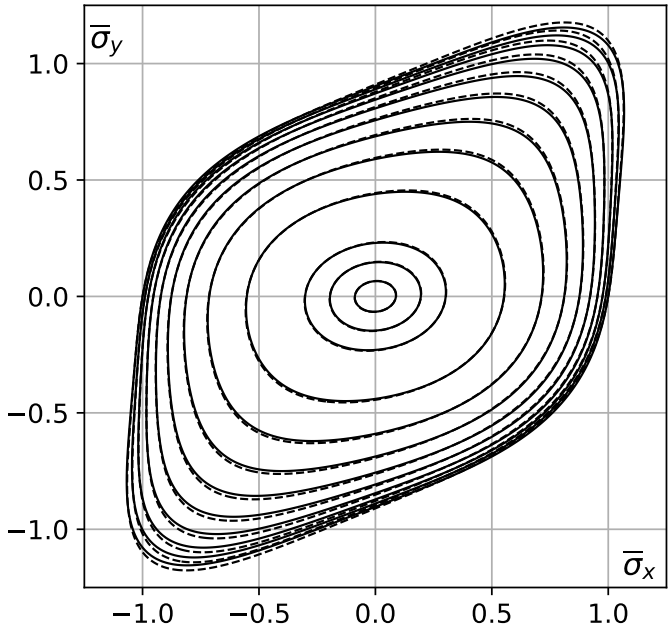

Figure.27: $\sigma_{xy} = const$ sections through the yield surfaces of the two Poly4 models shown in Fig.26: **ID#5**-solid, ID#3-dashed.

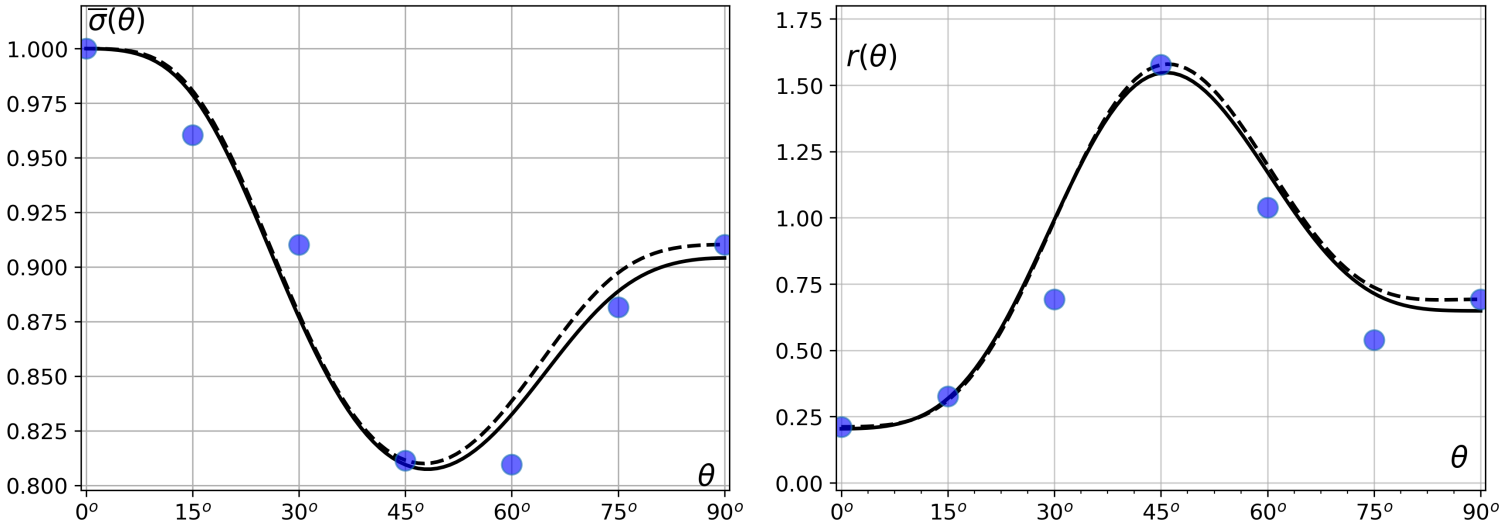

Figure.28: Directional yield stress (Left) and r-values (Right) of the Poly4 model of AA2090-T3 featured by solution **ID#6** in Table-2 of article shown with solid line (See Section-7.3 of the article for explanations on the nature of these solutions). Comparison with the customized Poly4 model row ID#3 (dashed). The blue markers show the experimental data.

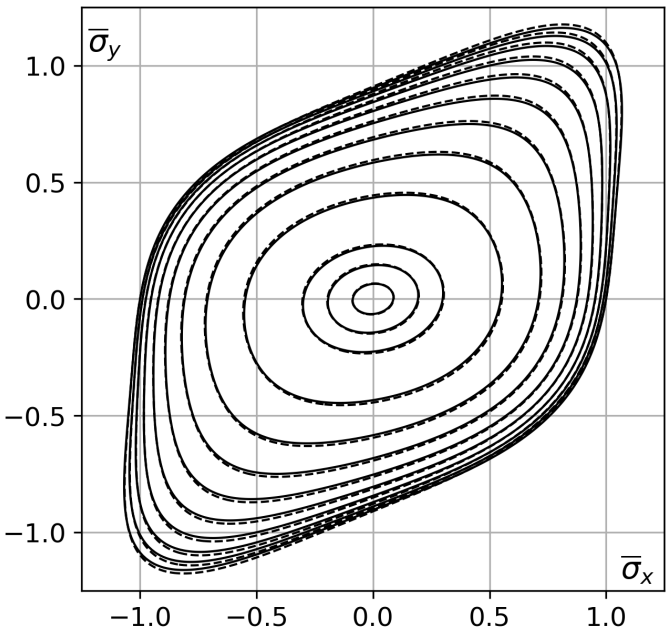

Figure.29: $\sigma_{xy} = const$ sections through the yield surfaces of the two Poly4 models shown in Fig.26: **ID#6**-solid, ID#3-dashed.

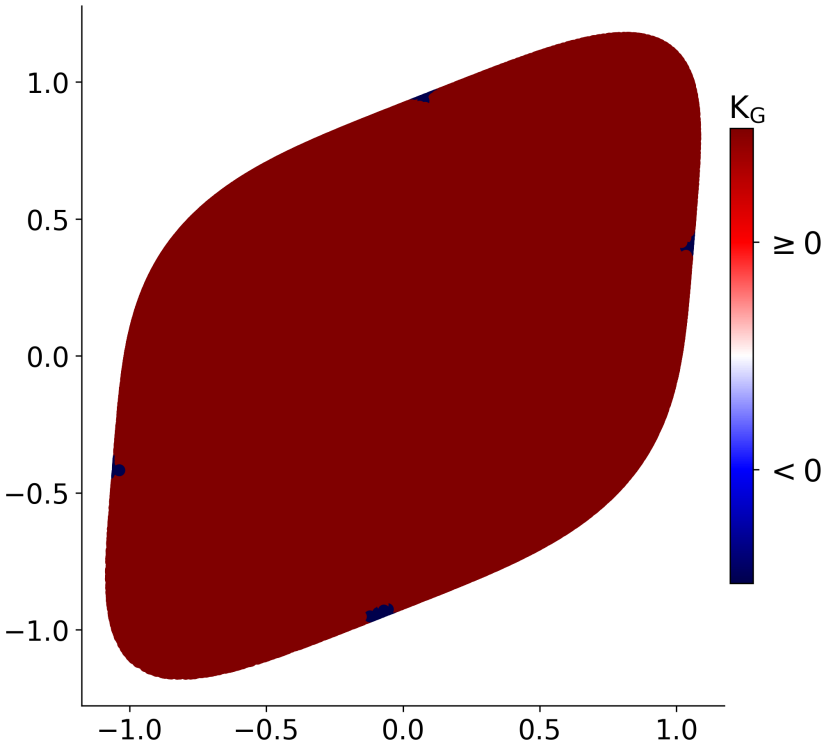

Figure.30: Verification of the convexity of the Poly4 model **ID#6** shown above in Figs-28 and 29. There are four blue patches near the middle of the straight edges of the outer contour that feature a negative minimum Gauss curvature $K_G \approx -0.0256$.
.

The NN-models `nb13` and `nb14` were subject to the same test methodology employed by Soare and Diehl(2023). Two types of FE-simulations were performed: uniaxial tensile tests on samples cut at angles

$$\theta \in \{0^o, 15^o, 30^o, 45^o, 60^o, 75^o, 90^o\}$$

from the rolling direction (RD), and cylindrical deep drawing. The purpose of the simulation of the uniaxial tests is dual: first to test/validate the correct implementation of the NN-model in the UMAT-subroutine; second, to compare the total simulation run-tines (with Poly8).

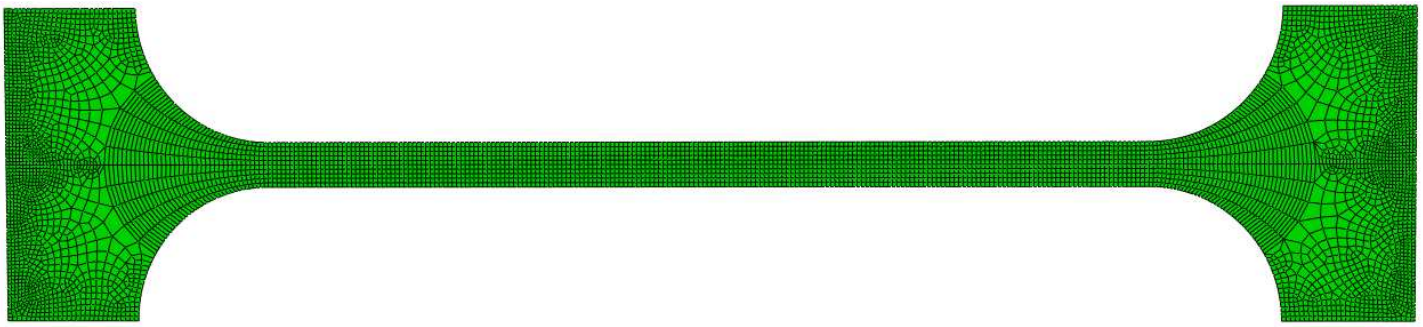

Figure.31: Mesh used on the uniaxial test geometry: 5712 S4R reduced integration elements, with 5 integration section points.

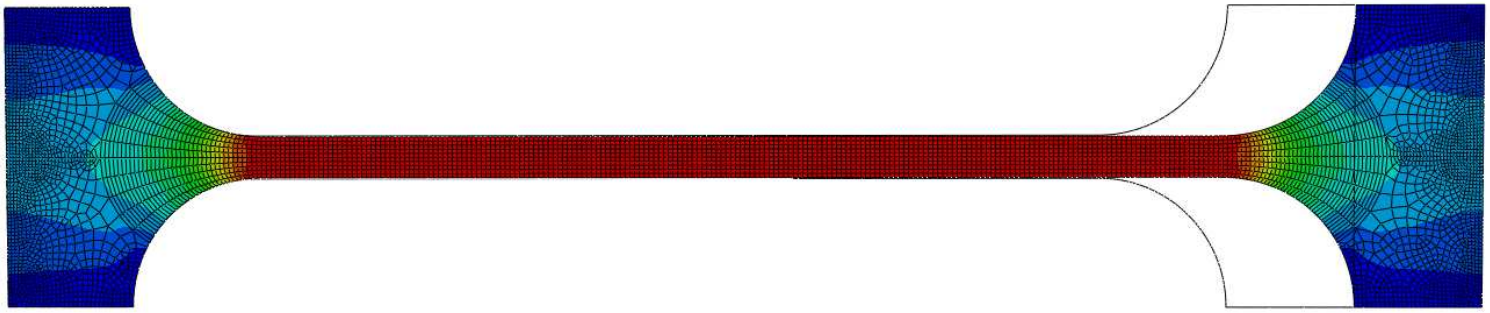

Figure.32: Typical (final) deformed configuration of a uniaxial test simulation (superimposed on the initial, un-deformed configuration).

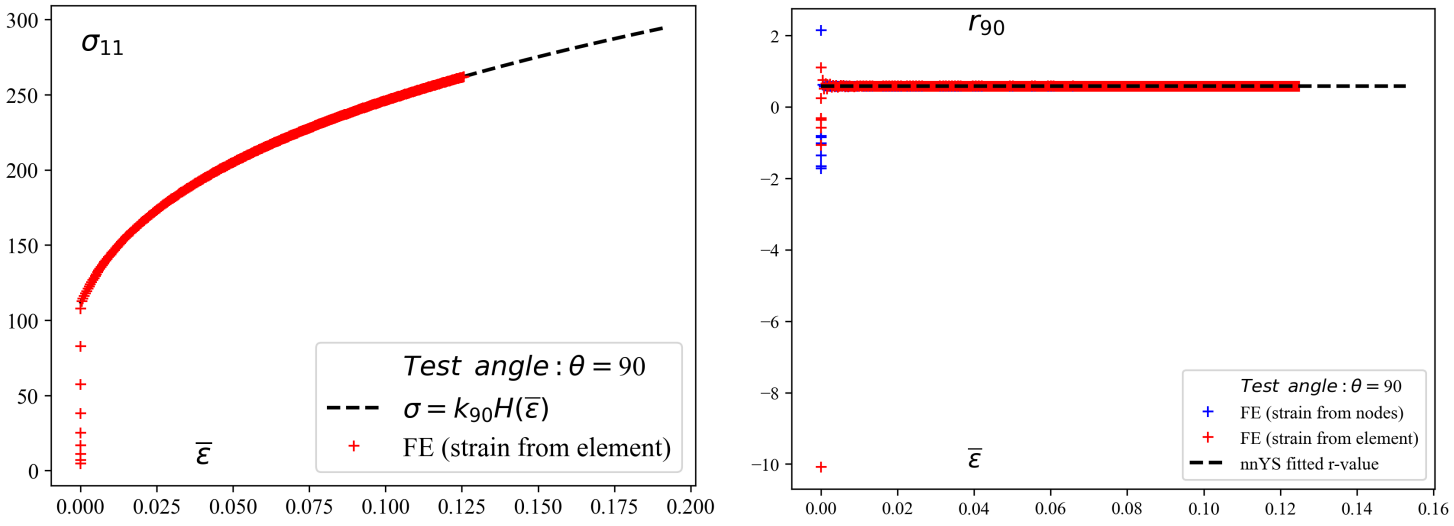

Figure.33: Typical stress strain curve (left) and r-value (right) of the test (here for $\theta = 90^o$).

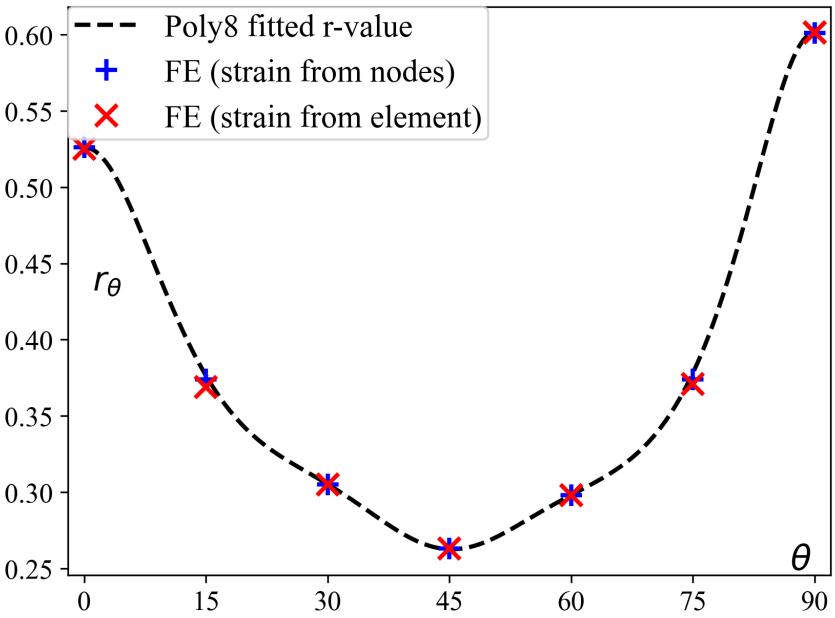

Figure.34: Poly8 r-values for AA6016T4: as fitted by the theoretical model (dashed); as predicted by finite element simulation with strains calculated from four reference nodes ('+' markers) and strains extracted from a reference element ('×' markers), see Fig-3 of the cited reference for details.

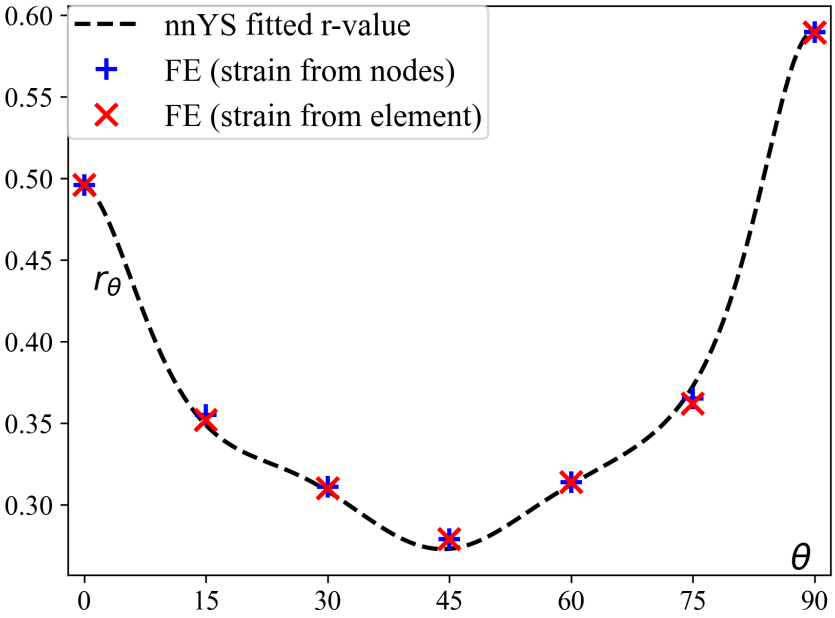

Figure.35: nb13 r-values for AA6016T4: as fitted by the theoretical model (dashed); as predicted by finite element simulation with strains calculated from four reference nodes ('+' markers) and strains extracted from a reference element ('×' markers), see Fig-3 of the cited reference for details.

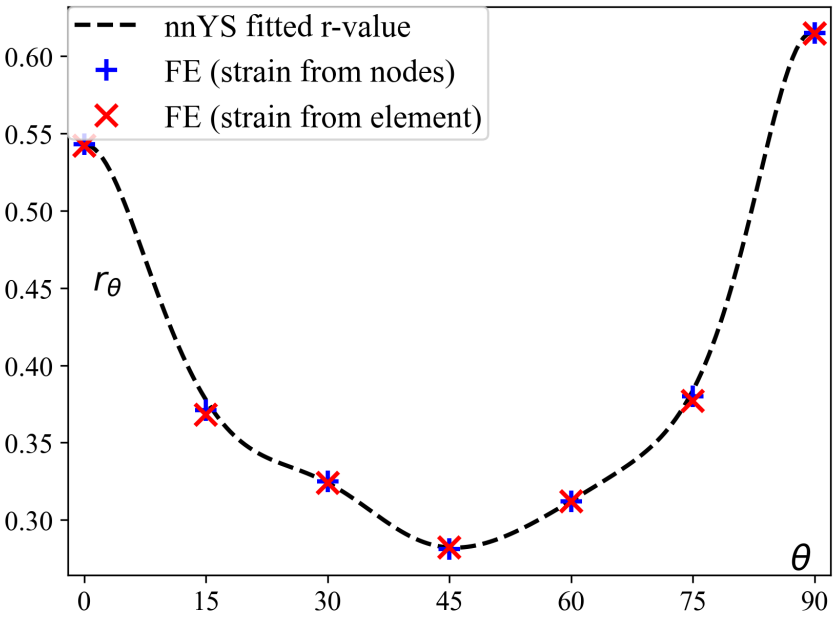

Figure.36: nb14 r-values for AA6016T4: as fitted by the theoretical model (dashed); as predicted by finite element simulation with strains calculated from four reference nodes ('+' markers) and strains extracted from a reference element ('×' markers), see Fig-3 of the cited reference for details.

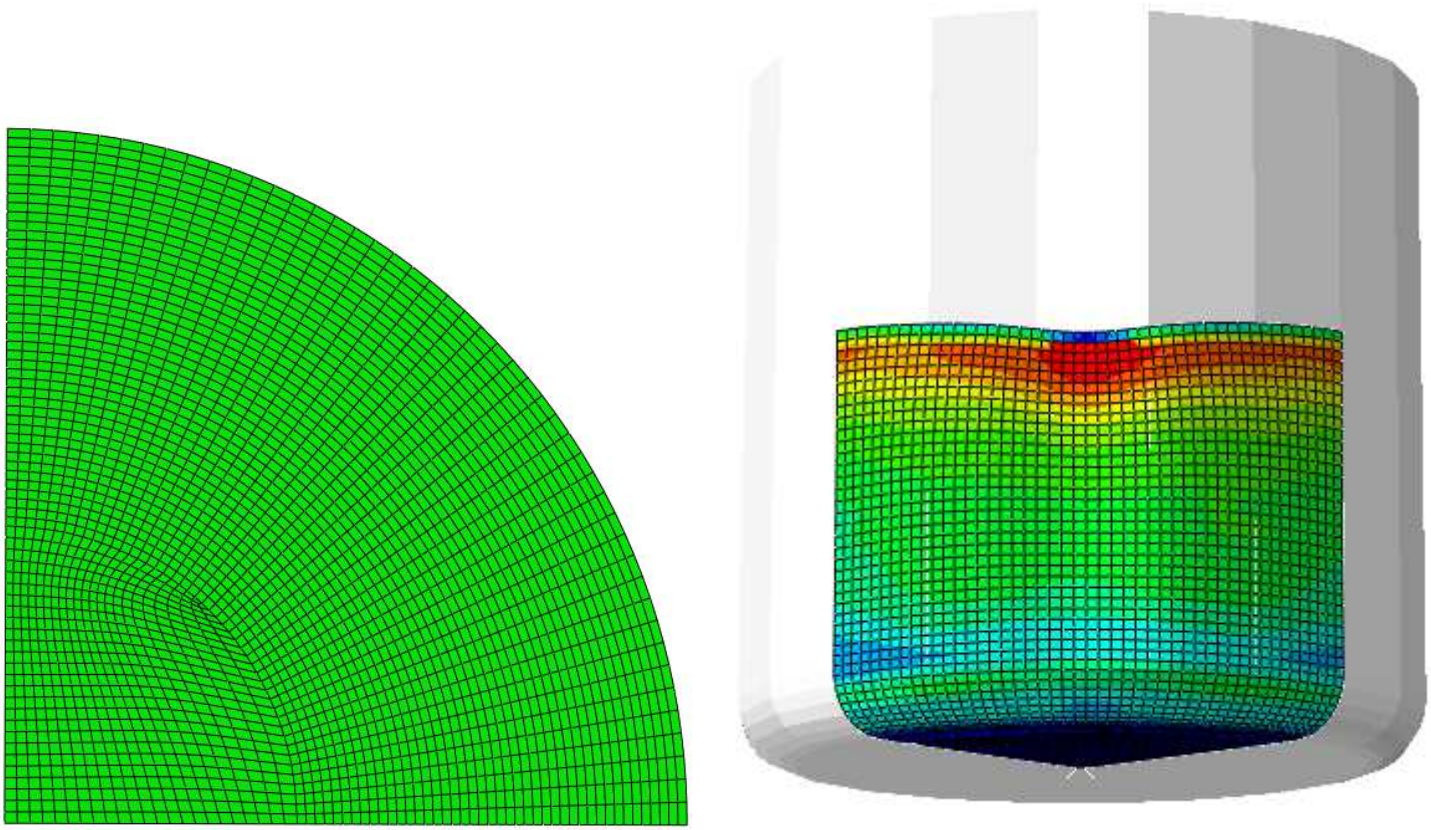

Figure.37: Left: Mesh used on the quarter blank for the three cup-drawing simulations reported in Appendix-B of the main article: 2618 S4R elements, with 11 integration section-points. **Right:** Typical (final deformed configuration (showing the simulated quarter-cup and the punch)
.

Simulation parameters
(data reported in Habraken et al(2022))

- Mechanical parameters
  - Isotropic elasticity: $E = 70$ GPa, $\nu = 0.33$
  - Swift hardening law: $\sigma\left(\overline{\epsilon}\right) = A\left(B + \overline{\epsilon}\right)^{C}$
    $A = 478.2$ MPa, $B = 0.0068$, $C = 0.2895$
  - Friction (between blank and tools) coefficient:
    $\mu = 0.09$
- Geometry
  (See Fig-6 in Soare and Diehl(2023))
  - Blank thickness: 1 mm
  - Blank diameter: 107.5 mm
  - Die opening diameter: 62.4 mm
  - Die shoulder radius: 10 mm
  - Punch diameter: 60 mm
  - Punch nose radius: 5 mm