TECHNISCHE MECHANIK
an open access journal

journal homepage: www.ovgu.de/techmech

Tech. Mech., Vol. 43, Is. 2, (2023), 220–229

Received: 14.04.2023
Accepted: 05.06.2023
Available online: 05.06.2023

Autoregressive neural networks for predicting the behavior of viscoelastic materials

Lukas Maurer*, Fabian Duvigneau, Daniel Juhre

Otto von Guericke University, Institute of Mechanics, Universitätsplatz 2, 39108 Magdeburg, Germany

Abstract: In the present work, the capabilities of neural networks to describe viscoelastic material behavior are investigated. Using real one-dimensional test data from a tensile test, autoregressive neural networks were trained. The best networks were then used to calculate the stress and the stiffness in displacement- and force-driven simulations. The results were compared with both experimental data and simulation results of a classical material model.
The viscoelasticity discussed here plays a special role in the description of complex rubber materials, in addition to long-term effects, failure or heat-induced mechanisms. Classical material models simplify the real behavior, which is the reason for the occurrence of simulation errors. To overcome these limitations, this paper presents a different way of material modeling by describing the strain-stress correlation using a neural network. Previous stress states from the time history are used in the calculation to account for the path-dependent behavior of viscoelastic materials. Other effects, such as the influence of different temperatures, are not addressed in this work, but can be included with an appropriately large training data set.

Keywords: autoregressive neural networks, viscoelasticity, finite element method, rubber material, deep learning, material model

1 Introduction

The response of a material can usually be described by various material models. These constitutive equations give a fixed relationship between the stress and the deformation process of the material. The influence of e.g. temperature (Fu et al. (2020)) or humidity (Ishiyama and Higo (2002)) can be described by different more or less physically motivated models.
As the number of different environmental influences increases, so does the complexity of the chosen model. The adjustment of the different material parameters can then become a hard challenge, since the influences of the different phenomena overlap. For this reason, common simulations focus on the desired application domain and try to keep the material model as simple as possible. The choice of a specific material model then defines the general form of the material response function. When adjusting the material parameters, the error between test and simulation data is minimized. However, due to the defined model, constraints on the shape of the stress-strain curve may occur, simplifying the real behavior at a rather early stage of the simulation.

To overcome these problems, some data-driven approaches have been developed in recent years. For example, Kirchdoerfer and Ortiz (2017) have used experimental material data directly instead of fitting a material model. Using the nearest neighborhood, the stress to a given strain can be determined directly from a similar condition from the experiments. For strongly nonlinear phenomena such as materials with highly viscoelastic or plastic behavior, the response depends on the prior history, which can only be mapped by interpolating large amounts of data and using tangent space information (Ciftci and Hackl (2022)). One of the major problems with using data-driven methods is their poor extrapolation capability, but even classical material models are kind of limited to the domain to which they have been fitted, as will be shown in further sections.

For comparison between classical material models and neural networks, a Mooney-Rivlin material model with several Prony series was fitted to the same test data used for training an autoregressive neural network. The data were obtained from a one-dimensional tensile test with a rubber material. Unlike many other works in this field, real test data were used. In this paper, first the used model and the difference between classical feed-forward networks and so-called autoregressive neural networks are described. Afterwards, the tensile test is outlined, since a more complex one is needed to capture the viscoelasticity and thus the rate-dependent material behavior. Besides a critical comparison between the two material descriptions, one can read in this paper about the advantages of autoregressive neural networks and how they can also contribute to the development of classical material models, as they offer the possibility to weight the influence of different physical phenomena for a given material.

2 Viscoelastic material model

The material model used in this work for comparison with the neural network results is a two-parameter Mooney-Rivlin material. One or more Maxwell elements consisting of a linear spring E_i and a damper η_i can be added to account for viscoelasticity. These Prony series are used to describe the strain rate dependent behaviour of e.g. rubber materials. Several stress-strain curves and the rheological model are shown in Fig. 1. The Maxwell elements here are parallel to a nonlinear spring which describes the

* E-mail address: lukas.maurer@ovgu.de
doi: 10.24352/UB.OVGU-2023-058

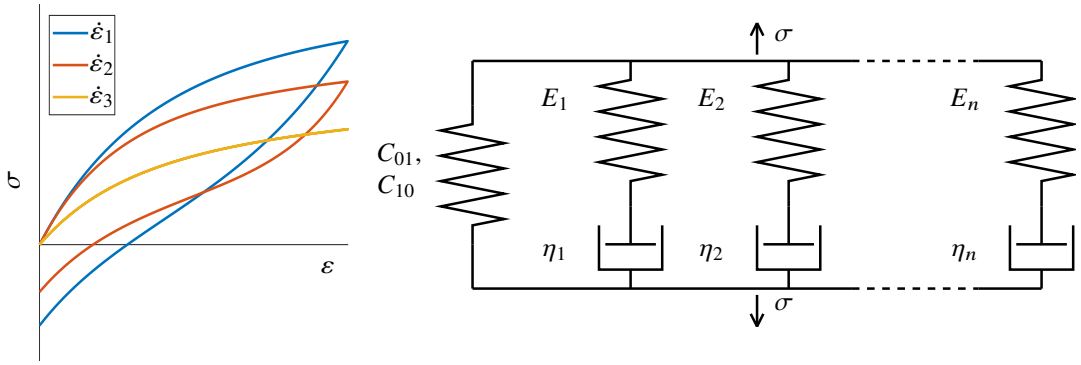

Fig. 1: Generalized Maxwell model (**left:** stress-strain curve with different strain rates, $\dot{\varepsilon}_1 > \dot{\varepsilon}_2 \gg \dot{\varepsilon}_3$; **right:** rheological model)

two-parameter Mooney-Rivlin material.

The graph shows that the stress-strain curve $\sigma\,(\varepsilon,\,\dot{\varepsilon})$ depends on the strainrate $\dot{\varepsilon}$. Higher strain rates result in higher stresses $(\dot{\varepsilon}_1 > \dot{\varepsilon}_2 \gg \dot{\varepsilon}_3)$. In the quasi static case $(\dot{\varepsilon}_3)$, the loading and unloading curves are almost identical, so that no hysteresis occurs. In this case, the stress is only prescribed by the nonlinear spring, which represents the incompressible Mooney-Rivlin material with the strain-energy-density function

$$W(\boldsymbol{\varepsilon}) = C_{01}\,(I_2 - 3) + C_{10}\,(I_1 - 3)\,. \tag{1}$$

With the stretch ratio λ, which is defined as the quotient of the current length to the initial one, the principal invariants I_1 and I_2 can be reduced for the one-dimensional incompressible case to

$$\begin{aligned} I_1 &= \lambda^2 + \frac{2}{\lambda}, \\ I_2 &= 2\lambda + \frac{1}{\lambda^2}. \end{aligned} \tag{2}$$

Therefore, the strain-energy-density function can be written as

$$W(\lambda) = C_{01}\left(2\lambda + \frac{1}{\lambda^2} - 3\right) + C_{10}\left(\lambda^2 + \frac{2}{\lambda} - 3\right). \tag{3}$$

By deriving the strain-energy-density function according to the normal strain ε, the true stress

$$\sigma_0 = \frac{\partial W(\varepsilon)}{\partial \varepsilon} \equiv \frac{\partial W(\lambda)}{\partial \lambda} = C_{01}\left(2 - \frac{2}{\lambda^3}\right) + C_{10}\left(2\lambda - \frac{2}{\lambda^2}\right) \tag{4}$$

can be calculated. The stress from the Mooney-Rivlin formulation acts in addition to the stress contributions from the prony series

$$\sigma = \sigma_0 + \sum_{i=1}^{n} \sigma_i. \tag{5}$$

For each Maxwell element, the strain in the spring ε_1 and the strain in the damper ε_2 can be used to calculate the stress

$$\sigma_i = E_i \varepsilon_1 = \eta \dot{\varepsilon}_2. \tag{6}$$

The strain definition

$$\varepsilon = \varepsilon_1 + \varepsilon_2 \quad \rightarrow \quad \dot{\varepsilon}_2 = \dot{\varepsilon} - \dot{\varepsilon}_1 \tag{7}$$

gives the differential equation, which can be solved using an exponential approach

$$\begin{aligned} &\dot{\varepsilon}_1 + \frac{E}{\eta}\varepsilon_1 = \dot{\varepsilon} \\ &\dot{\varepsilon}_1 = \dot{\varepsilon} - \beta e^{-\beta(t-t_0)} \int \dot{\varepsilon} e^{\beta(t-t_0)}\,dt \quad \text{, with } \beta = \frac{E}{\eta}. \end{aligned} \tag{8}$$

The rate of the stress can therefore be calculated by deriving Eq. (6)

$$\dot{\sigma}_i = E_i \dot{\varepsilon}_1 = \eta \ddot{\varepsilon}_2. \tag{9}$$

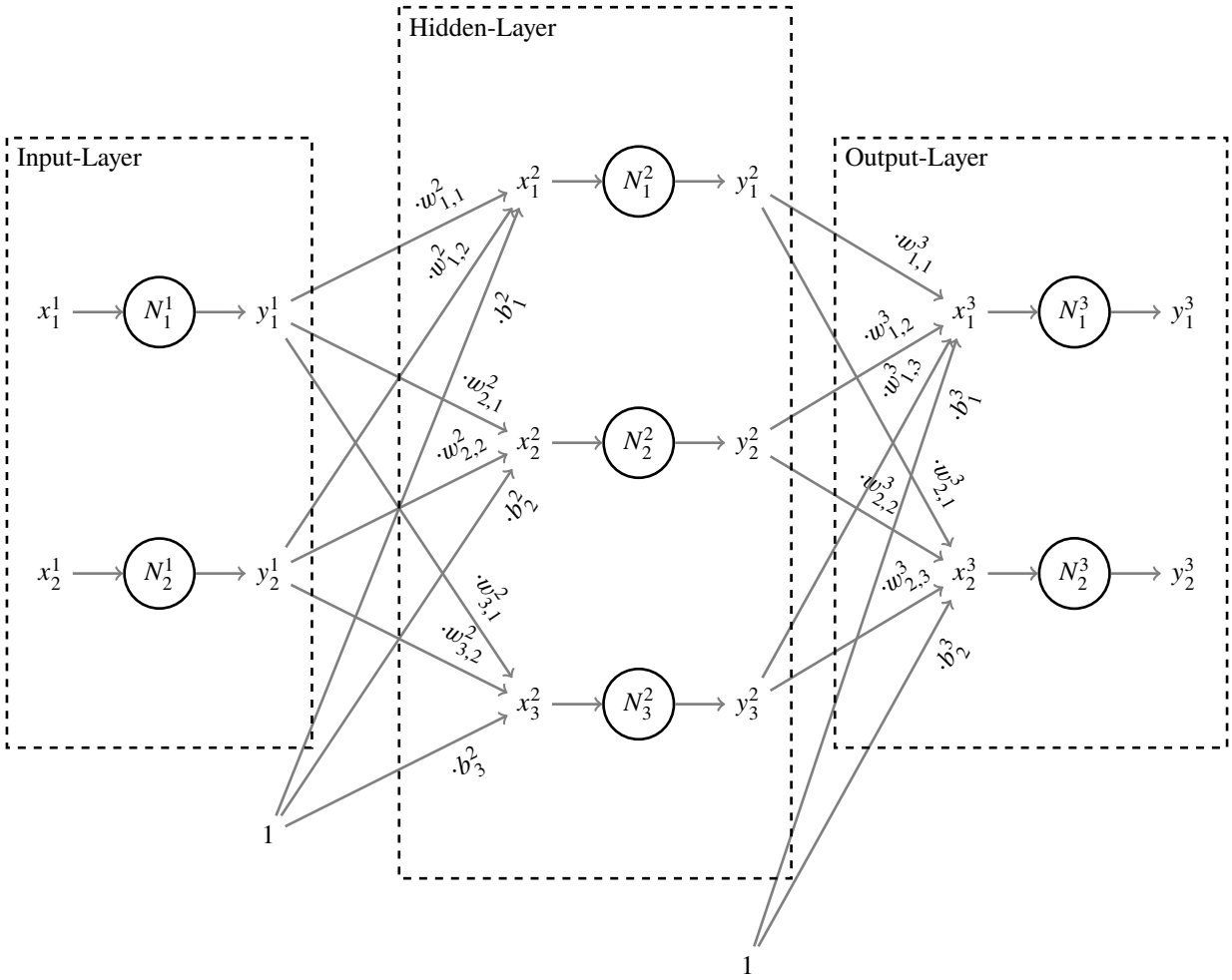

Fig. 2: Example of a feedforward neural network with two inputs, two outputs and a hidden layer with three hidden neurons

3 Neural networks

The presented material model is later fitted to the training data, and an autoregressive neural network is trained on the same data. In fact, fitting and training are very similar. However, the neural network has many more parameters, the so-called weights and biases.

Fig. 2 is a classical feedforward network with two inputs, two outputs and a hidden layer with three hidden neurons. The exponent indicates the layer number. In this example with only one hidden layer the last layer (the output layer) has the number three and therefore all neurons N_i^3, inputs x_i^3, outputs y_i^3, biases b_i^3 in this layer and all weights $w_{i,j}^3$ to this layer have the exponent three. The lower index indicates the specific neuron in the layer. Therfore x_3^2 is the input for the third neuron in the second layer. The weights have two numbers, as they build the connection between two layers. The weight $w_{i,j}^k$ connects the output of the neuron N_j^{k-1} with the input of neuron N_i^k.
The neurons N_i^k, shown as circles, are predefined functions that do not change during training. In the input and output layers these are usually linear functions that map the training data in the range between -1 and 1 and in the output layer in the range of the target set. Arguments of those functions are the neuron inputs x_i^k. Those are either the network inputs x_i^1 in the input-layer, or the neuron inputs inside the hidden- or output-layer. The function values are the neuron outputs y_i^k, in the output layer these variables are the outputs of the network.

The neuron inputs x_i^{k+1} are calculated using the weight matrix $\boldsymbol{W}$ and the neuron outputs of the previous layer y_j^k

$$\underbrace{\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ \dots \\ x_i^{k+1} \end{bmatrix}}_{\boldsymbol{x^{k+1}}} = \underbrace{\begin{bmatrix} w_{1,1}^{k+1} & w_{1,2}^{k+1} & \dots & w_{1,j}^{k+1} & b_1^{k+1} \\ w_{2,1}^{k+1} & w_{2,2}^{k+1} & \dots & w_{2,j}^{k+1} & b_2^{k+1} \\ \dots & \dots & \dots & \dots & \dots \\ w_{i,1}^{k+1} & w_{i,2}^{k+1} & \dots & w_{i,j}^{k+1} & b_j^{k+1} \end{bmatrix}}_{\boldsymbol{W^{k+1}}} \cdot \begin{bmatrix} y_1^k \\ y_2^k \\ \dots \\ y_j^k \\ 1 \end{bmatrix} \Rightarrow \boldsymbol{x^{k+1}} = \boldsymbol{W^{k+1}} \cdot \begin{bmatrix} \boldsymbol{x^k} \\ 1 \end{bmatrix}. \tag{10}$$

The output of the last layer $\boldsymbol{y}^K$ is the output of the neural network, which can thus be seen as a function of the inputs with the parameters $\boldsymbol{w}$ and $\boldsymbol{b}$.

$$\boldsymbol{y}^K = f\left(\boldsymbol{x}^1,\ \boldsymbol{w},\ \boldsymbol{b}\right) \tag{11}$$

The performance of a neural network can be described by the deviation between the network outputs $\boldsymbol{y}^K$ and the targets of the training set. The goal during training is to find a local minimum of this error function by adjusting the weights and biases. Almost all neural network training algorithms are based on the error backpropagation algorithm, where the influence of each parameter on this error is calculated - the gradient of the error function with respect to all weights and biases. For one of the simplest training algorithms, the gradient descent method, the parameters are changed in the negative direction of this gradient. For detailed information, see e.g. Goodfellow et al. (2016). In this work, the very common Levenberg-Marquardt algorithm was used (Marquardt (1963)).

Classical feedforward neural networks (Svozil et al. (1997)), such as the one shown in Fig. 2, may be well suited to represent hyperelastic material behaviour. In this case, each strain value is associated with only one specific stress value. For viscoelastic materials, or even more complex behaviour, the stress also depends on the history of the specimen, so these networks cannot be used in this form. Therefore, autoregressive neural networks are used in this work.

Autoregressive neural networks are a special type of feedforward neural networks. The output of the previous time step is used as one input for the next time step, allowing these networks to represent time series. Assuming a stress-strain curve, one obtains equally distant states in time with $t_0 = 0,\ t_1 = \Delta t,\ t_2 = 2\Delta t$. For example, the strain at time $t_5 = 5\Delta t$ is ε_{t_5} inducing the stress σ_{t_5} which, as a consequence of the path dependence of the viscoelastic material, is also influenced by all previous strain states $\varepsilon_{t_4},\ \varepsilon_{t_3},\ \varepsilon_{t_2},\ \ldots$. Since the number of inputs to a neural network is limited by the network structure, which can only be changed before training, the network can only take into account a limited number of previous time steps. To consider different strains and strain rates, the inputs of a possible autoregressive neural network could be $\boldsymbol{x}^1 = \begin{bmatrix}\sigma_{t_{n-3}} & \sigma_{t_{n-2}} & \sigma_{t_{n-1}} & \varepsilon_{t_{n-1}} & \varepsilon_{t_n}\end{bmatrix}^T$ to calculate the output $y^K = \sigma_{t_n}$. This exemplary network structure has five inputs. The last three stress values and the previous strain value are used to capture the history dependence and the current strain ε_{t_n} is used to determine the unknown stress state σ_{t_n}. For more information on this type of neural network, see e.g. Lin et al. (1996)

Since such networks describe a relationship between the current strain ε_{t_n} and the current stress σ_{t_n}, they can be directly integrated into a finite element environment. In non-linear simulations, the stiffness (matrix) changes during deformation and describes the relationship between the current strain and the current stress. This relationship can be determined by calculating the derivative of the network output σ_{t_n} according to the current strain ε_{t_n}. This gradient is then a function of the other inputs. However, this calculation can be very difficult, especially for large neural networks. A possible option would be to calculate the stiffness (matrix) using the finite difference method.

4 Experimental data

The two possible descriptions of the material from the previous sections were trained or fitted using real training data. For this purpose, a rubber material was used in a tensile test to obtain a one-dimensional stress-strain curve. The round test specimen has a diameter of 20 mm and a heigth of 15 mm. Two steel supports are glued to the specimen to hold it in place. The displacement and change in diameter were measured using an optical extensometer. All tests were performed displacement-driven, and the force was recorded with a load cell. The stress can thus be calculated from the force divided by the current cross-section of the specimen. Using the analytical solution from Chap. 2, the material parameters of the classical material model can be fitted using least squares optimization (Kraus et al. (2017)). This method is widely used, among other methods mentioned in Bradshaw and Brinson (1997). Unlike other projects where the test data contains various simple relaxation tests, e.g. Tzikang (2000), the experimental data is more chaotic, as more different data points are needed to train the neural network. Since the network does not know the basic shape of the response function given in the definition of the classical material model, a larger amount of data with many different strain rates is needed.

In order to obtain different strain rates in a single tensile test, the velocity of the displacement-driven experiment must change several times. Therefore, the tensile tests were performed in several displacement steps with equal time intervals. With this method it is possible for different strain rates to occur and it is also possible to have steps with almost only relaxation. A test consisting of 60 steps with a target displacement between 0 and 2 mm of the 15 mm long test specimen was therefore performed. The random distributetd displacement values results in strain rates between 0 and 13.3%. Each step has a time increment of 5 s, resulting in strain rates between 0 and 0.027 $\frac{1}{\mathrm{s}}$. This complex experiment is not necessary for fitting the classical material model, as the analytical solution only allows reasonable stress-strain curves. However, it is necessary for the neural network to obtain enough data points for training, as the number of parameters to be fitted is much higher, since there is no equivalent to the physically motivated function of the material model.

The complete training data contains a total of 7750 strain-stress pairs. The data and the experimental setup are shown in Fig. 3.

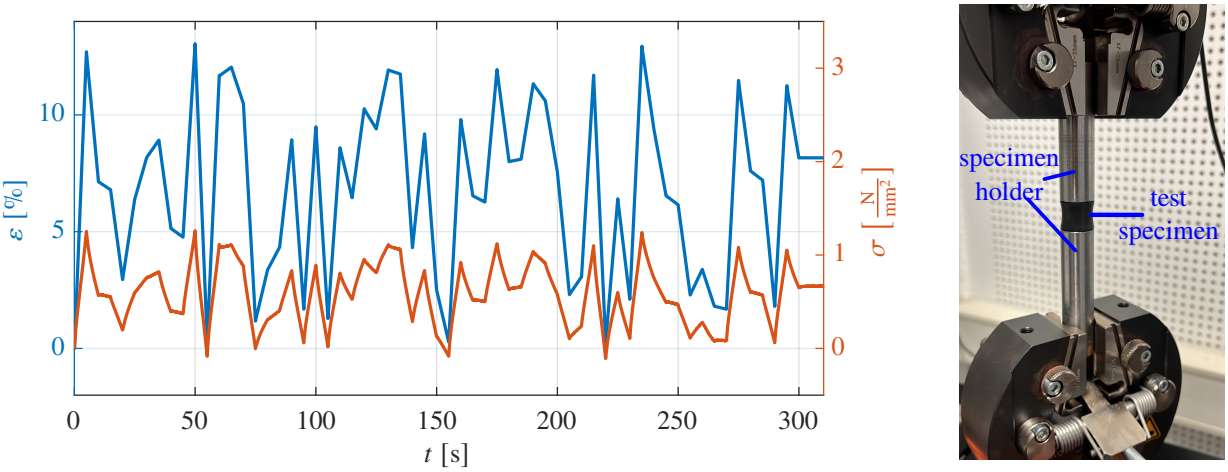

Fig. 3: Experimental tensile test (**left:** test data with maximum 2 mm elongation; **right:** experimental setup)

5 Fitting the material models

In order to compare the accuracy of the neural network approach, the visco-elastic material model presented in Chap. 2 was first fitted with a different number of Prony series. As already explained, the material parameters were calculated by minimizing the mean squared error (MSE) of the analytical stress contribution compared to the experimental results, therefore it cannot be guaranteed that the global minimum and thus the best material parameters were obtained. However, varying the initial values of the material parameters for the optimization process did not change the fitted material parameters. As expected, the accuracy increases with the number of Maxwell elements, but reaches a lower limit at three Prony series with a total of eight material parameters, as shown in Tab. 1.

The objective for the neural network approach is therefore to have an error of less than $4.22 \cdot 10^{-4}$. The use of the error backpropagation algorithm to train the neural network is also based on minimizing the squared error, so a good comparison can be made. Since the initialisation of the weights and biases is random, the same network structure will end up in different states for each training, so the same network structure will need to be trained several times to exploit the potential of a particular structure. The choice of input variables is another influence factor affecting the performance. As mentioned in Chapter 3, the input variables of the autoregressive neural network are the current strain ε_{t_n} and the previous states of stress and strain to cover the dependence on the history. To clarify the required inputs, the procedure described in the following chapter can be used.

5.1 Clarify inputs

Defining the structure of a neural network is always a challenge; in addition to the number of input layers and the number of neurons they contain, for autoregressive neural networks the determination of the necessary prior states, the so-called input delays, must also be clarified. In addition to the classical trial-and-error approach illustrated in Ruiz et al. (2016) for autoregressive neural networks, there are two ideas that can help. The first idea is based on the development of a performance criteria by means of a decision tree. A sufficiently large network structure is used and the number of delays is varied, starting with only one delay for each variable, in our case the current strain ε_{t_n} and the stress of the previous step $\sigma_{t_{n-1}}$. Multiple training of this network structure gives a starting point from which the MSE should decrease. Then the performance is calculated with one more strain state and afterwards exclusively with one more stress state. The network that performs better is used as the basis for the next comparison. The iteration proceeds in the direction of the optimal distribution of the input values. With two additional strain values and the last two stress values, the best results were obtained with this method for the given example studied in the paper at hand. The input to the neural network should therefore be ε_{t_n}, $\varepsilon_{t_{n-1}}$, $\varepsilon_{t_{n-2}}$, $\sigma_{t_{n-1}}$, $\sigma_{t_{n-2}}$. Unreasonably large networks may lead to overfitting and poor generalization, and would also increase subsequent computation time (Ying (2019)).

The first method, which involves analyzing each iteration step and training each network multiple times, is relatively time-consuming and only suitable for a small number of different input variables. In the second method, a small network with only one neuron in the first hidden layer is trained to decide which input variables to use. Obviously, this will perform poorly and cannot be used to

Tab. 1: Accuracy of the analytical viscoelastic material model with respect to the number of Prony series

Prony series	Number of material parameters	MSE
0	2	$2.24 \cdot 10^{-3}$
1	4	$5.36 \cdot 10^{-3}$
2	6	$4.23 \cdot 10^{-4}$
3	8	$4.22 \cdot 10^{-4}$

make further predictions. However, a clear indication of the influence of the different input variables can be obtained from this trained network (Goh (1995)). Referring to Fig. 2, each input is represented linearly in the range between −1 and 1 by the input neurons N_i^1.
Over the entire training set, the distribution of the outputs of the neurons of the input layer y_i^1 is approximately the same, since, for example, the input x_1^1 of the first training point becomes the second input of the next training point. The distribution of stress is almost the same as the distribution of strain, which is not necessarily the case for different variables. Under this condition, the weight between the input and the first hidden layer (with only one neuron) can be used to estimate the influence of the different variables and their delay. If the inputs are not similar distributed, this can lead to wrong assumptions. For example, one of the inputs has many data points in the range between −5 and 5, but also two data points −50 and 50. With the mapping function, these inputs would be mapped in the range of −1 (−50) and 1 (50), so the main data points would be in the range between −0.1 and 0.1. That the main part of the data influences the input of the next neuron as if the outlier values were not present, the weight must be much higher (factor 10) than the weight for a normally distributed neuron. This can lead to a wrong interpretation of the weights.

However, this is not the case with our data set. The current and the last four strain inputs and the last five stress inputs were used as inputs for the training. The normalized weights are shown in Eq. (12) for different time step sizes.

$$\boldsymbol{x}^{\mathbf{1}} = \begin{bmatrix} \varepsilon_{t_n} \\ \varepsilon_{t_{n-1}} \\ \varepsilon_{t_{n-2}} \\ \varepsilon_{t_{n-3}} \\ \varepsilon_{t_{n-4}} \\ \sigma_{t_{n-1}} \\ \sigma_{t_{n-2}} \\ \sigma_{t_{n-3}} \\ \sigma_{t_{n-4}} \\ \sigma_{t_{n-5}} \end{bmatrix} \Rightarrow \underset{(\,\Delta t = 0.04\ \mathrm{s})}{\boldsymbol{w}^{\mathbf{2}}} = \begin{bmatrix} -0.930 \\ 1.000 \\ 0.240 \\ 0.087 \\ -0.398 \\ -0.603 \\ -0.229 \\ -0.023 \\ 0.179 \\ 0.056 \end{bmatrix}, \quad \underset{(\,\Delta t = 0.10\ \mathrm{s})}{\boldsymbol{w}^{\mathbf{2}}} = \begin{bmatrix} -0.682 \\ 1.000 \\ -0.141 \\ -0.118 \\ -0.062 \\ -0.703 \\ 0.076 \\ 0.088 \\ 0.049 \\ -0.007 \end{bmatrix}, \quad \underset{(\,\Delta t = 0.20\ \mathrm{s})}{\boldsymbol{w}^{\mathbf{2}}} = \begin{bmatrix} -0.590 \\ 1.000 \\ -0.367 \\ -0.057 \\ 0.009 \\ -0.721 \\ 0.262 \\ 0.028 \\ -0.009 \\ 0.003 \end{bmatrix} \tag{12}$$

As one can see, the values marked in red are the largest absolute values within the vector. The network has therefore analyzed them as the most important for calculating the output. As expected, the most recent variables are the most important, in particular the inputs ε_{t_n}, $\varepsilon_{t_{n-1}}$ and $\sigma_{t_{n-1}}$. The fourth highest value depends on the size of the time step. For larger time steps, where the network was trained with a coarser sampling rate of the experimental data, the next highest inputs are $\varepsilon_{t_{n-2}}$ and $\sigma_{t_{n-2}}$. This would, of course, be the most intuitive order of the inputs. Looking at the first weight vector, trained with the full dataset, the five most interesting inputs have changed. Here, $\varepsilon_{t_{n-4}}$ have a much higher influence. One explanation for this could be that the network is much better at accepting long term effects with these inputs than it is with another input that is very close to the others. The total number of used inputs depends on the complexity of the used network. A larger network would have the ability to leverage the influence of an input that only has an absolute weight of 0.1 or less to achieve better network performance. All network structures trained in this work did not provide any further benefit with additional inputs below an absolute weight of 0.2. Training networks with a coarser sampling rate only served to explain the scaled weights method; in the remainder of this paper, we will focus on networks trained with the full dataset ($\Delta t = 0.04$s). Therefore, ε_{t_n}, $\varepsilon_{t_{n-1}}$, $\varepsilon_{t_{n-2}}$, $\varepsilon_{t_{n-4}}$, $\sigma_{t_{n-1}}$ and $\sigma_{t_{n-2}}$ are the best input correlations for a network of this time step size with six inputs.

This approach was confirmed by training a larger network with more than one neuron in the first hidden layer. With the same number of input variables, the network with the described inputs performed better than the network with the inputs without a missing time step.

5.2 Training of the neural network

Different network structures with two or three hidden layers and up to five hidden neurons in each layer were trained. In the hidden neurons the usual tanh-sigmoid transfer functions were used (Wlodzislaw and Jankowski (2001)). For the training of an autoregressive network, two different types of performance criteria can be obtained. The training performance is computed during training using the error between the goals of the training set and the output of the network. This error is calculated for each data point, forming the mean square error.
During training, this performance criterion is used to adjust the network parameters. However, for the later application, a simulation where the results of the previous step are used for the next step, this criterion should not be used alone to evaluate the applicability of the network. For example, an error could occur that is relatively small, but the network is so sensitive that the output of the next time step causes the simulation to crash, or the error accumulates over the next steps and leads to incorrect results for the entire simulation, even if the performance criterion is very small in relation to the training data set.

The second performance is calculated over the whole data set. Time steps are calculated one after the other. This means that small errors can occur and the network has to prove its stability over time. The second performance is therefore much more interesting. It can also be used for comparison with the generalised Maxwell model.

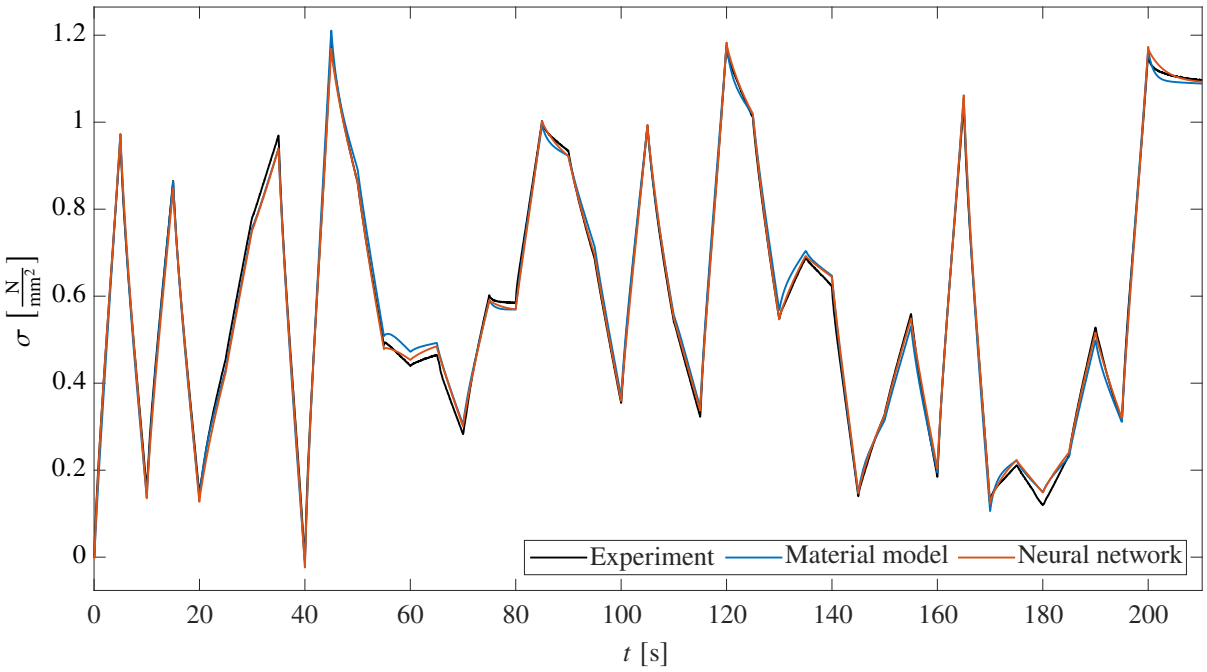

Fig. 4: Comparison of the simulation results of the material model and the neural network regarding the interpolation set

The best results were obtained with a small network with two hidden layers and three neurons in each layer. In total the network has 42 network parameters (weights and biases) and achieved a MSE of $2.64 \cdot 10^{-4}$. Hence, it has a better performance on the training data set than the fitted Generalised Maxwell model. The training was performed in a few seconds, which is much faster and more convenient than the parameter identification of the classical model. Here, the analytical solution has to be programmed and the calculation involves many more previous time steps, which is why the optimization takes longer (at least under the assumption that the same data set is used for fitting and for training).

In the following, the two material descriptions are used to perform several displacement-driven simulations and compare the results with those of experimental tests. In Chap. 6.4, a force-driven simulation is presented to explain the steps involved in implementing the network formulation in a finite element environment.

6 Comparison

In the previous section it was shown that the autoregressive network can describe the material behaviour as well as the classical material model. The advantages are a faster parameter identification and a better fit to the experimental data. Since both material models can describe the real material from the experiment quite well, the next step is to test the stability of the neural network in different test sets.

For the displacement-driven test, the neural network can be used directly without the need for an iterative integration scheme, just as for the classical material model. The strain vector is determined, so it is possible to calculate the stress at any given point for the material model, since the stress-strain formulation depends only on the history of the strain curve. The neural network has to calculate the stress-strain curve step by step because the current calculation of stress requires previous stress states as input for the autoregressive formulation.

The initial inputs can be set to zero. The first input vector for the autoregressive neural network is therefore

$$\boldsymbol{x}_{t_1}^{\mathbf{1}} = \begin{bmatrix} \varepsilon_{t_1} \\ \varepsilon_{t_0} \\ \varepsilon_{t_{-4}} \\ \sigma_{t_{-1}} \\ \sigma_{t_{-4}} \end{bmatrix} = \begin{bmatrix} \varepsilon_{t_1} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \Rightarrow \quad \sigma_{t_1} = \boldsymbol{y}_{t_1} = f\left(\boldsymbol{x}_{t_1}^{\mathbf{1}},\, \boldsymbol{w},\, \boldsymbol{b}\right). \tag{13}$$

6.1 Interpolation set

To evaluate the generalizability of the two material descriptions, a second experiment was performed with the same restrictions as for the training set. Here, different strain rates occur that lie in between the strain rates from the training set. As shown in Fig. 4, both models can reproduce the displacement driven experiment quite well. With the training set, the neural network's MSE of $2.13 \cdot 10^{-4}$ is slightly better than the material model's MSE of $3.61 \cdot 10^{-4}$.

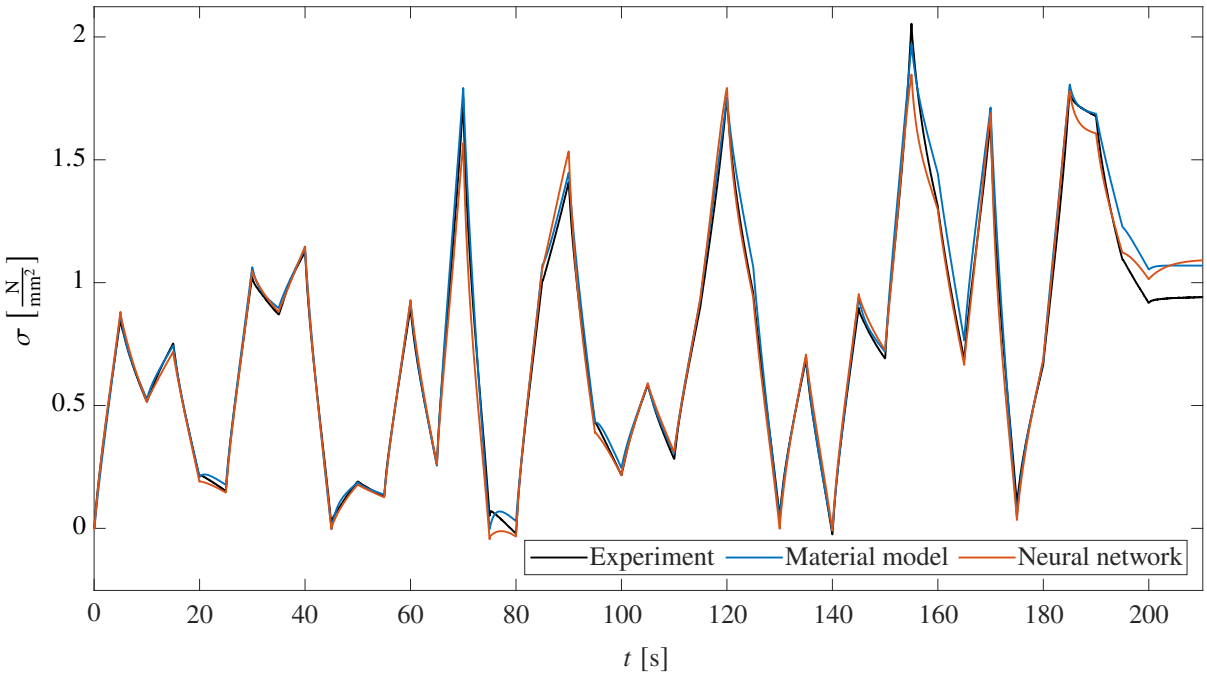

Fig. 5: Comparison of the simulation results of the material model and the neural network regarding the extrapolation set

Both values are smaller than the performance criteria for the training set, which is due to the distribution of strains and strain rates. A set with more critical values results in a higher MSE. The fact that the neural network again performs better on a complete new set than the material model suggests that within the range of the training set, the neural network can describe the behavior of the material better than the classical material model.

A well-known disadvantage of neural networks is their poor extrapolation capabilities (Haley and Soloway (1992)). To investigate this for the autoregressive neural network material formulation, a third experiment was performed.

6.2 Extrapolation set

For the extrapolation set, the same routine as for the creation of the training and interpolation set was performed for a larger range of displacements. Here, the displacements are within the normal range up to 2 mm or 13.3 % strain for the first 10 loading steps. After that, 30 loading steps were performed with displacements up to 3 mm or 20 % strain. As can be seen in Fig. 5, both material descriptions have major problems at the highest stress peaks and also at the end where a very small strain rate meets a high strain.

The MSE for the neural network of $3.43 \cdot 10^{-3}$ is slightly worse than the MSE of the classical material model of $3.53 \cdot 10^{-3}$. The bigger problem here is that both material models have an error ten times larger than the training or interpolation set, which cannot be due to the larger stress values alone.

As expected, the autoregressive neural network can only be useful within the range of the training data. It is interesting to note that the classical material model also has major problems in extrapolation and cannot provide reasonable results outside the training or fitting range.

All tests and training were performed under cyclic loading and unloading. To investigate the ability to account for long-term effects, a creep test and a relaxation test were performed.

6.3 Relaxation test

For the relaxation test a constant strain of 10 % was applied to the specimen within 5 s. The experimental data and simulation results are shown in Fig. 6.

Since the stress values are in the range of the training set, both material descriptions capture the peak stress value quite well. As can be seen, the neural network has a much shorter relaxation time. Since the training set does not contain such data points, the network does not have the ability to learn these effects. Regarding the structure of the network, only the last two stress values affect the next stress value, as the strain values remain the same. This describes the steady state where the stress values of the material model keep decreasing. The relaxation function is part of the material model, and the neural network had to learn it through a cyclic loading and unloading training set.

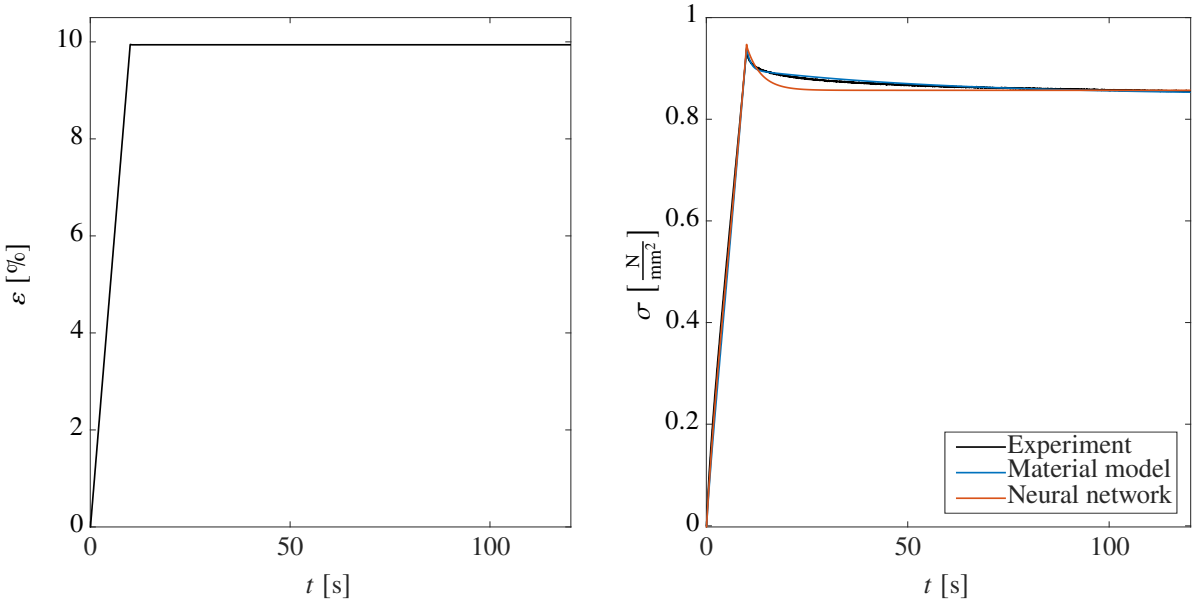

Fig. 6: Comparison of the simulation results of the material model and the neural network for the relaxation test

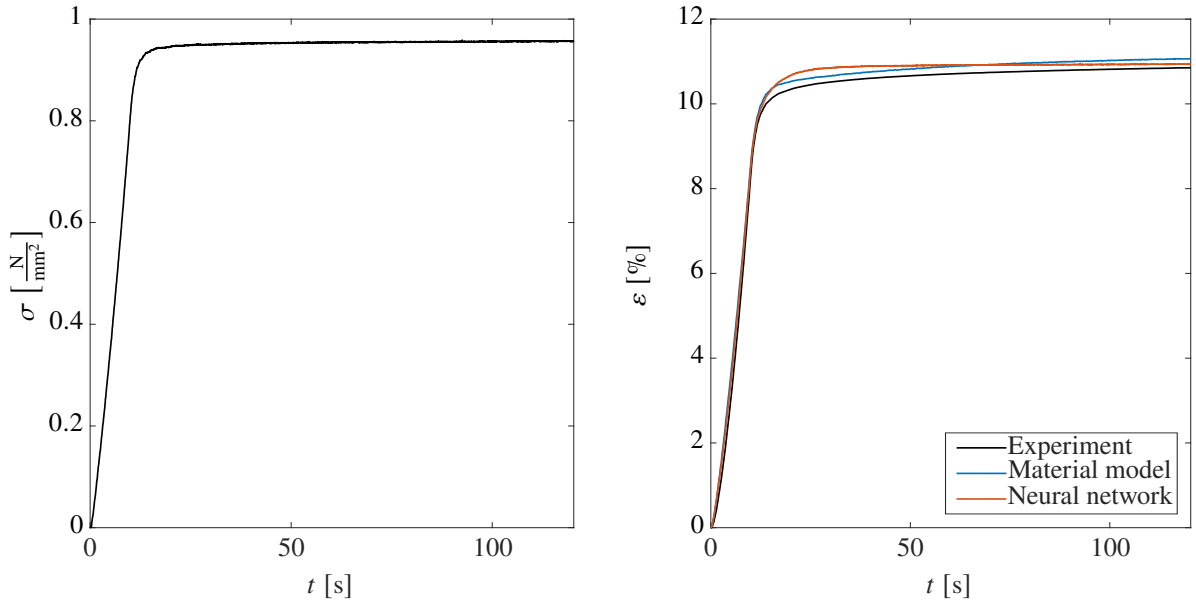

Fig. 7: Comparison of the simulation results of the material model and the neural network for the creep test

6.4 Creep test

For the creep test, both models must be used in an iterative integration scheme. At each time step, the derivative of the stress-strain equation is used to calculate the additional strain required to reach equilibrium for the material equation. For the classical material model this is quite straightforward. The second order partial derivative of the strain-energy density function (Eq. (1)) with respect to strain forms the stiffness matrix. For the neural network this is also possible but the calculation becomes more time consuming. The nested transfer functions lead to a very long formulation of the tangent stiffness matrix and the previous state variables have to be substituted several times. Therefore, a finite difference method was used for the calculation in this work. This can also be a possible way for the implementation of a neural network in a finite element method as well. The result of the creep test is shown in Fig. 7.

As with the relaxation test, the results look good, although the same problems occur. The neural network stops relatively quickly in the steady state, while the material model describes the relaxation better because it is part of the material formulation.

7 Summary and Outlook

As shown in the last section, the autoregressive neural network can describe a complex material almost as well as the classical material. Since neural networks are generally poor at extrapolation, the training set must cover the range of subsequent use. This is also true for fitting classical material models, e.g. Prony series for long-term effects would require test data at a much lower frequency, which cannot be covered by the experimental scheme presented here. An advantage of the classical model is that it tends

to be more stable and can produce physically meaningful results over a wide extrapolating range. Outside the training domain, the network would behave very differently from the material model, as these structures generally have poor extrapolation capabilities.

A major advantage of neural networks is the ability to create a model that fits the experimental data better than classical material models. With sufficient coverage of the input range and a well-tested network structure to avoid overfitting (Ying (2019)), this can increase the accuracy of a simulation for a unknown material. Autoregressive networks have the advantage of taking into account the history of the material, as they use previous states of the variables in their calculation. The network formulation can also be used to find the most important variables, as shown in Chap. 5.1. This weighting of inputs for small networks can be used to decide whether different parameters, such as temperature, affect the response of the material or whether other variables are more important. This information can also help in selecting the best material model or in developing new models.

In the method presented here, each neural network has a defined step size, which can lead to problems in the finite element simulation, as the time step size could be smaller than that of the network. For such cases, several networks could be trained with different step sizes. Another approach presented by Jung and Ghaboussi (2006) was to use the strain rate as an additional input. With this idea, it is possible to train the network with different time step sizes.

For more complex materials, further investigation is required, for example, to account for the effect of temperature on the behaviour of rubber materials. For a real application in finite element simulation, the whole concept has to be transformed into a three-dimensional space. This is another challenge as the experiments become much more complex as the strains and stresses have to be measured in multiple directions. Another interesting part is to increase the efficiency of the tangent stiffness matrix, which is more time-consuming with the finite difference method than with the direct formulations known from classical material models.

The possibility of describing an unknown material using a data-driven approach is the major result that warrants further research. This idea has been presented in this paper for the one-dimensional case with convincing results.

References

Roger Bradshaw and L.C. Brinson. A sign control method for fitting and interconverting material functions for linearly viscoelastic solids. *Mechanics of Time-Dependent Materials*, 1:85–108, 01 1997. doi: 10.1023/A:1009772018066.

Kerem Ciftci and Klaus Hackl. Model-free data-driven simulation of inelastic materials using structured data sets, tangent space information and transition rules. *Computational Mechanics*, 70, 08 2022. doi: 10.1007/s00466-022-02174-x.

Xintao Fu, Zepeng Wang, Lianxiang Ma, Zhaoxuan Zou, Qingling Zhang, and Xinxin Guan. Temperature-dependence of rubber hyperelasticity based on the eight-chain model. *Polymers*, 12(4), 2020. ISSN 2073-4360. URL https://www.mdpi.com/2073-4360/12/4/932.

Anthony Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9:143–151, 12 1995. doi: 10.1016/0954-1810(94)00011-S.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

P.J. Haley and Don Soloway. Extrapolation limitations of multilayer feedforward neural networks. pages 25 – 30 vol.4, 07 1992. ISBN 0-7803-0559-0. doi: 10.1109/IJCNN.1992.227294.

C. Ishiyama and Y. Higo. Effects of humidity on young's modulus in poly(methyl methacrylate). *Journal of Polymer Science Part B: Polymer Physics*, 40:460 – 465, 03 2002. doi: 10.1002/polb.10107.

Sungmoon Jung and Jamshid Ghaboussi. Neural network constitutive model for rate-dependent materials. *Computers & Structures*, 84:955–963, 06 2006. doi: 10.1016/j.compstruc.2006.02.015.

Trenton Kirchdoerfer and Michael Ortiz. Data driven computing with noisy material data sets. *Computer Methods in Applied Mechanics and Engineering*, 326, 02 2017. doi: 10.1016/j.cma.2017.07.039.

Michael Kraus, Miriam Schuster, Johannes Kuntsche, Geralt Siebert, and Jens Schneider. Parameter identification methods for visco- and hyperelastic material models. *Glass Structures & Engineering*, 2, 10 2017. doi: 10.1007/s40940-017-0042-9.

Tsungnan Lin, William Horne, Peter Tino, and C. Giles. Learning long-term dependencies in narx recurrent neural networks. *Neural Networks, IEEE Transactions on*, 7:1329 – 1338, 12 1996. doi: 10.1109/72.548162.

Donald Marquardt. An algorithm for least square estimation of non-linear parameters. *SIAM Journal on Applied Mathematics*, 11: 431–441, 06 1963. doi: 10.1137/0111030.

Luis Ruiz, Manuel Cuéllar, Miguel Calvo-Flores, and Maria del Carmen Pegalajar Jiménez. An application of non-linear autoregressive neural networks to predict energy consumption in public buildings. *Energies*, 9:684, 08 2016. doi: 10.3390/en9090684.

Daniel Svozil, Vladimir Kvasnicka, and Jiří Pospíchal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39:43–62, 11 1997. doi: 10.1016/S0169-7439(97)00061-0.

Chen Tzikang. Determining a prony series for a viscoelastic material from time varying strain data. 06 2000.

Duch Wlodzislaw and Norbert Jankowski. Transfer functions: hidden possibilities for better neural networks. pages 81–94, 01 2001.

Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022, 02 2019. doi: 10.1088/1742-6596/1168/2/022022.