Machine Learning for Computational Science and Engineering (2025) 1:31
https://doi.org/10.1007/s44379-025-00031-1

RESEARCH

A constitutive neural network for incompressible hyperelastic materials

Sanghee Lee[1] · Klaus-Jürgen Bathe[1]

Received: 7 July 2025 / Accepted: 23 July 2025

Abstract
We propose a B-spline-based constitutive neural network to model the mechanical behavior of incompressible isotropic materials. The theoretical foundation of this network is the Sussman-Bathe model which interpolates tension–compression test data points and recovers the strain energy function. Our neural network uses regression to self-optimize the knot configurations of the B-splines and to determine a twice differentiable curve of the material response that is closely aligned with the given data points. We address datasets displaying physically complicated behaviors. Through the patch test validation of the constitutive model and illustrative example solutions, we highlight the flexibility inherent in spline-based models and the automated approximation capabilities enabled by neural networks.

Keywords Finite element analysis · Material representations · Hyperelastic materials · Neural networks · Patch test · Further development of Sussman-Bathe model

1 Introduction

Finite element analysis is now widely used [1, 2], and constitutive modeling plays a crucial role in the field of analysis. A constitutive model is a mathematical representation of material behavior under various loading conditions, describing the relationship between stress and strain, and is used when simulating real-world materials. Thus, it is imperative to develop all-encompassing constitutive models for scientific research and the design of structures in engineering applications. Reliable numerical simulations using finite element analysis procedures require effective constitutive models for accurate response predictions.

To reach effective material representations, researchers have developed various constitutive models for different materials. The classically established constitutive models for isotropic rubber-like materials include the Neo-Hookean [3], Treloar [4], Mooney-Rivlin [5] and Ogden [6] models, but additional models are also valuable, like those presented by Arruda-Boyce [7], Yeoh [8], and Gent [9]. The underlying approach in these representations is to model the strain energy with assumed functions of the strain tensor. The stress–strain relation can then be calculated by differentiating the strain energy representation with respect to the strains. For example, the Neo-Hookean and Mooney-Rivlin models use the invariants derived from the strain tensor, defining the strain energy function as a polynomial of the invariants. Another example is the Ogden model with a power-law of the principal stretches defining the strain energy. The coefficients in these models are unknown and material dependent.

However, these functions predefined over the entire stress–strain domain may not be able to capture a complex behavior of the material, that, for example, includes peaks or wiggles. The Sussman-Bathe model [10] resolves this issue by introducing a spline interpolation approach, in which an inversion formula is used to recover the strain energy function from the interpolated stress–strain data. The model has shown good applicability in representing the stress–strain data of incompressible isotropic materials accurately and was extended to transversely isotropic materials by M. Latorre and F. J. Montáns [11].

On the other hand, the recent development of machine learning techniques indicates great potential in various fields (see, e.g. F. J. Montáns, E. Cueto and K. J. Bathe [12]). Many authors have applied a neural network to constitutive

Klaus-Jürgen Bathe
kjb@mit.edu

[1] Department of Mechanical Engineering and Center for Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA

modeling to predict material behavior. More than three decades ago, J. Ghaboussi, J. H. Garrett, and X. Wu [13] introduced the use of neural networks to predict the stress–strain relation of materials with a feed-forward neural network (FFNN). This early stage of neural network application was purely data-driven. Since then, FFNN have been widely adopted to predict material behaviors with attempts to incorporate prior knowledge of material theory into the neural network. For example, in ref. [14], a FFNN was applied to obtain the strain energy function of hyperelastic materials as a curve fitting problem, in ref. [15] a Constitutive Artificial Neural Network (CANN) was proposed to predict the stress–strain relation of materials using various distinct FFNN and in ref. [16], a parametrized NN is used to solve a hyperelasticity problem. Indeed, the use of neural networks in constitutive modeling has become advanced and diverse. P. Thakolkaran et al. [17] adopted an unsupervised learning technique with physics-motivated loss functions. L. Linden et al. [18] established a neural network model that satisfies the physical constraints of the material such as symmetry of the stress tensor, polyconvexity, material symmetry, and thermodynamic consistency. The last property is imposed through the invariants of the strain tensor as an input to the neural network.

Despite many efforts to incorporate physical constraints into the neural network models and their good performance in predicting the material behavior [19], most of the neural network models are still considered to be black-box models and consist of fully connected feed forward neural networks with hidden layers. Such concern regarding the interpretability of neural network models has been raised in many fields, and constitutive modeling is not an exception. K. Linka and E. Kuhl [20] proposed to use suitable expressions as activation functions in the neural network. In this way, the NN weights can choose the best strain energy function that fits the data. Not being fully connected and the selection of suitable activation functions makes the network quite flexible.

However, while this approach is effective to represent the classical models, it also inherits the limitations of these material representations. The predefined strain energy functions of the classical models as a basis of the model may not be able to capture a local peak or other complex behaviors in the stress–strain curve [10].

Given the advancements in finite element analysis solvers [21], such as adaptive mesh refinement and load–displacement-constraint methods [1, 21] for solutions, there is an increasing need for advanced constitutive models that emphasize the flexibility of the material model, rather than mathematical simplicity or mathematical conditions for computational stability such as polyconvexity. In this paper, inspired by the Sussman-Bathe model, we propose a B-spline-based constitutive neural network for use in modeling incompressible isotropic hyperelastic material behavior. While the Sussman-Bathe model, as usually used [10], effectively represents stress–strain data by interpolating given data points, it is susceptible to overfitting due to the nature of the spline interpolation. To overcome this limitation and make the model more general, we employ a B-spline based regression procedure within the neural network model.

B-spline-based NN architectures have recently been applied in constitutive modeling, see [22] and [23], to represent strain energy functions with additional corrections, but our study employs a B-spline-based network to directly represent the stress-strain response based on the Sussman-Bathe model. This direct approach is valuable for practical finite element analyses as we discuss below.

An effective constitutive model should be well-integrated with finite element procedures to produce accurate results. Several researchers have developed various NN-based constitutive models for hyperelastic materials and validated the models through finite element analysis [24]. For instance, I. Chung, S. Im, and M. Cho [25] developed a data-driven model for polymer materials where the dataset was generated through molecular dynamics simulations. The authors embedded their NN model in finite element analysis and analyzed a plate with a hole and a corner brace model. A. Mendizabal, P. Márquez- Neila, and S. Cotin [26] used three benchmark examples – a cantilever beam, an L-shape plate, and a model of a preoperative CT scan of a human liver – and demonstrated hyperelastic materials can be simulated in real-time. Similarly, in this study we employ finite element case studies to verify the performance of our proposed model.

An important validation of a finite element solution procedure is the patch test, which is a classical diagnostic used to assess the physical consistency of a finite element formulation. Hence the passing of the patch tests for the applicable stress and strain states is extremely important. And yet, the patch test validation seems to have rarely been addressed in machine learning-based finite element developments, but see the research of J. Jung, K. Yoon, and P. S. Lee [27]. In our achievements presented below, we incorporate the patch test validation to assess the validity of our material model for finite element analyses.

In the following sections, we first discuss the theoretical background of the Sussman-Bathe model as originally published and fundamental concepts of B-splines. We then introduce the neural network model that we use and its architecture. The training process and the objective function are described in detail. Finally, we present the results of illustrative examples and discuss the advantages and limitations of the proposed model. There are many different materials, not just pure rubbers, which exhibit locally varying trends in the stress–strain curve. Our model provides an example of a flexible and automated approach to constitutive modeling for such complex materials.

2 The Sussman-Bathe model and B-spline representation

While, classically, constitutive models use a predefined strain energy function for curve fitting, the Sussman-Bathe model uses stress–strain data directly to represent the material behavior through splines and establish an expression for the strain energy.

2.1 The Sussman-Bathe model

The strain energy representation of an incompressible isotropic hyperelastic material is in the Sussman-Bathe model given by

$$W = \sum_{i=1}^{3} w\left(e_i\right) \tag{1}$$

where $w(e_i)$ denotes the strain energy density as a function of the Hencky (logarithmic) strain e_i. Here, each e_i represents the principal logarithmic strain in the i th principal direction (i = 1,2,3). The Cauchy stress τ_i corresponding to e_i can be obtained by differentiating the strain energy function with respect to the Hencky strain e_i

$$\tau_i = \frac{\partial W}{\partial e_i} + p_H \tag{2}$$

where p_H is the hydrostatic pressure. We note that we will use the same function $w(e)$ for each principal strain direction.

In a uniaxial tensile-compressive test, for which we assume the physical data to be given, we have $e_1 = e, e_2 = e_3 = -\frac{1}{2}e$. Hence, at the current level of strain e, we obtain the following relation

$$\tau(e) = w'(e) - w'\left(-\frac{1}{2}e\right) \tag{3}$$

because $W|_e = w(e) + 2 \cdot w(-\frac{1}{2}e)$.

Once the stress–strain relation is established through a spline approximation, the strain energy function can be recovered by the inversion formula

$$w'(e) = \sum_{k=0}^{\infty} \tau\left(\left(-\frac{1}{2}\right)^k \cdot e\right) \tag{4}$$

Therefore, to represent the strain energy we need to describe in a reliable way the uniaxial stress–strain behavior of the material. In order to have an all-encompassing model for both tensile and compressive behaviors, the Sussman-Bathe model requires both tensile and compressive experimental data to represent the stress–strain curve accurately. More details about these derivations can be found in Appendix A.

In the original Sussman-Bathe model, the stress–strain data is interpolated by a spline function, and the strain energy function is obtained using Eq. (4). We introduce here a B-Spline technique replacing the interpolation of the stress–strain data. This approach helps to reduce the potential overfitting issue and to make the model more effective.

2.2 B-Spline representation

The B-spline approximation $S(x)$ of a function $f(x)$ is given by

$$f(x) \approx S(x) = \sum_{i=1}^{n} c_i B_{i,p}(x) \tag{5}$$

where $B_{i,p}(x)$ is the B-spline basis function of degree p, n is the number of basis functions, and c_i is the B-spline coefficient. Each basis function $B_{i,p}(x)$ is defined on a local interval, and its shape is determined by the configuration of a knot vector. We note that for cubic B-splines, five consecutive knots define one B-spline basis function and the function vanishes outside these knots.

One common approach for defining B-spline basis functions is through recursion. Let $\mathbf{t} = \{t_1, t_2, \dots, t_n\}$ be a nondecreasing knot vector. Then, the recursive formula for B-spline basis functions is given by

$$\begin{aligned} B_{i,0}(x) &= \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \\ B_{i,k}(x) &= a_{i,k}(x) \cdot B_{i,k-1}(x) + b_{i,k}(x) \cdot B_{i+1,k-1}(x) \text{ for } k = 1, 2, 3 \end{aligned} \tag{6}$$

where

$$a_{i,k}(x) = \begin{cases} \frac{x-t_i}{t_{i+k}-t_i} & \text{if } t_{i+k} > t_i \\ 0 & \text{if } t_{i+k} = t_i \end{cases}, \quad \text{and}$$

$$b_{i,k}(x) = \begin{cases} \frac{t_{i+k+1}-x}{t_{i+k+1}-t_{i+1}} & \text{if } t_{i+k+1} > t_{i+1} \\ 0 & \text{if } t_{i+k+1} = t_{i+1} \end{cases}$$

This recursive formula is known as the Cox-de Boor recursion formula. In this work, we use cubic B-splines ($p = 3$). To simplify the notation, we denote the cubic B-spline basis function as $B_i(x) := B_{i,3}(x)$ when $p = 3$. This basis function has nonzero support on the interval $[t_i, t_{i+4}]$. An example derivation of a B-spline function is given in Appendix B.

Examples of using B-spline basis functions are illustrated in Fig. 1. Six distinct knots are equally placed in the interval [0, 1] with three additional knots placed at each of the two boundaries. The basis functions $B_i(x)$ $(i = 1, 2, \cdots, 8)$ define nonzero values on the interval $[t_i, t_{i+4}]$ where t_i is the i-th knot. Two example spline functions $S_1(x)$ and $S_2(x)$, formed by a linear combination of $B_i(x)$ as in Eq. (5), are shown in Fig. 1, with the following values: For $S_1(x)$, we have

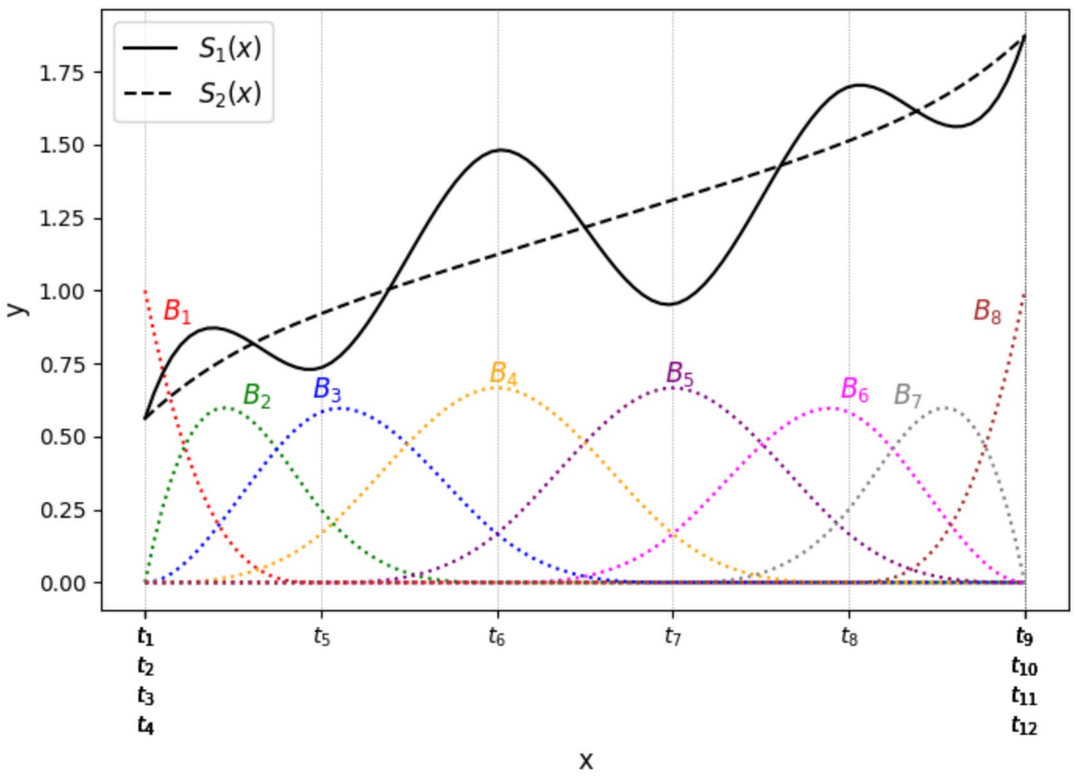


Fig. 1 Example of B-spline basis functions. Cubic B-splines are depicted with equally spaced knots, except at the boundaries. $S_1(x)$ and $S_2(x)$ are two examples of functions given by the basis functions

$$\begin{aligned} &c_1 = 0.5618, c_2 = 0.8790, c_3 = 0.7012, c_4 = 1.4153, \\ &c_5 = 1.0173, c_6 = 1.7313, c_7 = 1.5536, c_8 = 1.8708, \end{aligned} \tag{7}$$

and for $S_2(x)$, we have

$$\begin{aligned} &c_1 = 0.5618, c_2 = 0.7488, c_3 = 0.9358, c_4 = 1.1228, \\ &c_5 = 1.3098, c_6 = 1.4968, c_7 = 1.6838, c_8 = 1.8708. \end{aligned} \tag{8}$$

As represented in the figure, B-spline basis functions are smooth and have local support. We can obtain a good approximation of a smooth and locally varying function by placing the knots and finding the coefficients properly.

Using cubic B-splines is advantageous because the resulting function is C^2-continuous. This is a desirable property for the strain energy function of a hyperelastic material, as hyperelastic materials are expected to have a smoothly varying behavior in the stress–strain curve.

Finding an optimal number of knots and corresponding coefficients is a challenging task. If the number of knots is too large, the model may overfit the data, potentially leading to undesirable fluctuations and numerical instabilities. On the other hand, if the number of knots is too small, the model may not adequately capture the characteristics of the stress–strain curve. In addition, the knot placement is also not a straightforward task. If the knots are not placed properly, the model may not be able to capture a locally rapid change in the curve or may introduce unnecessary oscillations in the curve. In particular, knots should be placed more densely in the region where the target curve shows a rapid change in the slope as we encounter in the example of Section 4.1.2.

3 Neural network design

Our goal is to implement the B-spline-based constitutive model in a neural network for which we need to design the architecture. The network should take the Hencky strain as input and give the Cauchy stress as output. The input is passed to a function evaluator $\mathcal{B}$, which calculates each value of the B-spline basis function at the input point. The resulting values are multiplied by the B-spline coefficients obtained from a layer, and the sum of these values is the output of the network. The neural network architecture is illustrated in Fig. 2. The model mainly consists of two parts to be trained: the knot controller and a linear layer.

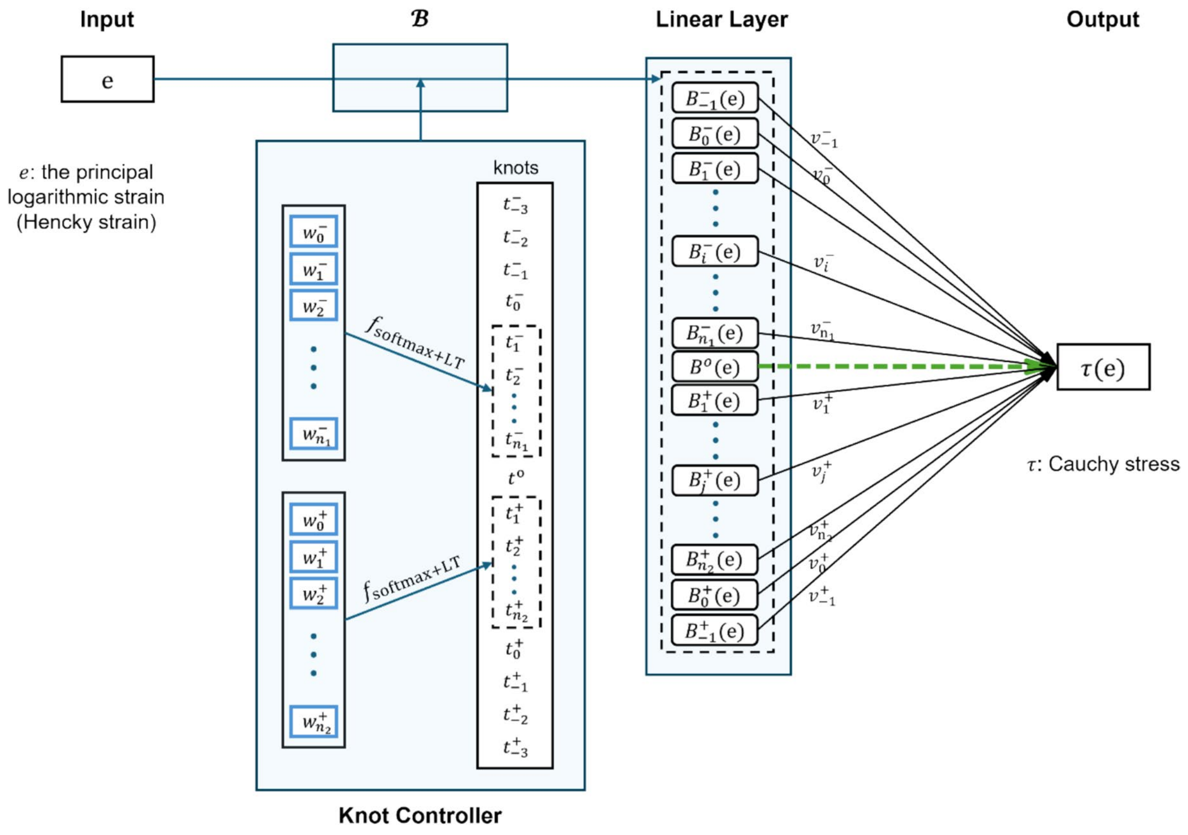

Input
e
e: the principal logarithmic strain (Hencky strain)
$\mathcal{B}$
Knot Controller
knots
w_0^-, w_1^-, w_2^-, $w_{n_1}^-$
w_0^+, w_1^+, w_2^+, $w_{n_2}^+$
$f_{\text{softmax+LT}}$
t_{-3}^-, t_{-2}^-, t_{-1}^-, t_0^-, t_1^-, t_2^-, $t_{n_1}^-$, t^o, t_1^+, t_2^+, $t_{n_2}^+$, t_0^+, t_{-1}^+, t_{-2}^+, t_{-3}^+
Linear Layer
$B_{-1}^-(e)$, $B_0^-(e)$, $B_1^-(e)$, $B_i^-(e)$, $B_{n_1}^-(e)$, $B^o(e)$, $B_1^+(e)$, $B_j^+(e)$, $B_{n_2}^+(e)$, $B_0^+(e)$, $B_{-1}^+(e)$
v_{-1}^-, v_0^-, v_i^-, $v_{n_1}^-$, v_1^+, v_j^+, $v_{n_2}^+$, v_0^+, v_{-1}^+
Output
$\tau(e)$
τ: Cauchy stress

Fig. 2 Architecture of the B-Spline-based constitutive neural network. The input is the Hencky strain, and the output is the Cauchy stress. $\mathcal{B}$ is the function evaluator that calculates the B-spline basis function values at the input points. The green dashed line represents the coefficient at zero strain that is not learnable and must be determined by Eq. (17). The learnable parameters are $w_i^{\{+,-\}}$ and $v_i^{\{+,-\}}$, and they are described in detail in Section 3.1

3.1 Neural network architecture

We begin by outlining the requirements for knots and the structure of the knot configuration, assuming that we use basis functions of degree 3. Then, each layer of the network is introduced and explained in detail.

3.1.1 Knot requirements

Knots are the points that divide the one-dimensional strain domain (the total interval) into subintervals, within which piecewise polynomial functions are defined. The good positioning of the knots is very important because the positions decide the shape of the B-spline basis functions. Without loss of generality, we place the knots within the range of the input data in an increasing order. The knots at the boundaries are fixed at the first and last data points, and our interest is to determine effective positions of the internal knots.

There are a few necessary conditions that the knots should satisfy.

Firstly, one knot must be placed at the origin. The undeformed state of the material corresponds to the origin of the stress–strain curve, with compression before and tension after the origin. At the origin, the stress and strain are zero. By placing a knot at the origin and adjusting the B-spline coefficients accordingly (see Eq. (17)), we ensure that the stress–strain curve passes through the origin.

Secondly, the knots at each end of the interval must have multiplicity four. We have a nonzero stress at the boundaries; thus, we need a basis function that does not vanish at these interval ends. This can be achieved by placing four knots at the same position at each boundary.

Thirdly, the internal knots should all be distinct. From Eq. (6), we can deduce that any repetition of internal knots leads to reduced smoothness, possibly manifested as a sharp spike in the B-spline basis functions. Thus, we do not allow any repetition of knots except for the knots at the boundaries.

In short, the knots must satisfy the following three conditions:

1/ One knot should be placed at the origin.
2/ Four knots must be placed at each end of the complete interval.
3/ Knots should be placed in a strictly increasing order by position (except at the boundaries).

3.1.2 Knot controller

The number of internal knots is prescribed to the knot controller, hence needs to be defined by the user of the constitutive model. The controller then determines the placement of the internal knots. To ensure that the above knot conditions are satisfied, we separately set two positive integers for the number of internal knots – one for the negative domain and the other for the positive domain.

Let n_1, n_2 be the given number of internal knots (thus excluding the origin and the left or right boundary, whichever applicable) for the negative and positive domains, respectively, obtained from the number of data points, see Section 3.2.2. We need learnable parameters to determine the placement of the internal knots. Given the number of internal knots, the unknowns are the positions of the knots, or equivalently, the distances between the knots. Hence, $n_1 + 1$ and $n_2 + 1$ should be the total number of learnable parameters in the knot controller, say $\{w_0^-, w_1^-, \dots, w_{n_1}^-\}$ and $\{w_0^+, w_1^+, \dots, w_{n_2}^+\}$. We apply a softmax function to these weight vectors to obtain two vectors with all positive components that (for each vector) sum to one

$$softmax(w_i) = \frac{e^{w_i}}{\sum_{j=0}^{n} e^{w_j}}. \tag{9}$$

We note that any vector $\boldsymbol{\mu} = \{\mu_0, \mu_1, \cdots, \mu_n\}$, where $\sum_{i=0}^{n} \mu_i = 1$ and $\mu_i > 0$ for all i, can partition an interval (x_l, x_r) into n subintervals, by mapping each element proportionally to the length of the interval length using a linear transformation. This concept is mathematically expressed as follows:

$$x_i = x_l + (x_r - x_l) \cdot \sum_{j=0}^{i-1} \mu_j, \quad i = 1, 2, \dots, n \tag{10}$$

and we call x_i the partition points. We use this property to determine the positions of the knots. Since we have a vector with its elements summing to one (using the softmax function), the knot positions are obtained directly from the partition points.

Let $\bar{e}_{min}$ and $\bar{e}_{nmin}$ denote the smallest and second (next) smallest values in the strain data. By taking $x_l = \bar{e}_{nmin} < 0$ and $x_r = 0$, the negative knots are determined using

$$t_{-3}^- = t_{-2}^- = t_{-1}^- = t_0^- = \bar{e}_{min},$$
$$t_i^- = \bar{e}_{nmin} + (0 - \bar{e}_{nmin}) \cdot \sum_{j=0}^{i-1} softmax(w_j^-) \quad \text{for } i = 1,2,\dots,n_1 \tag{11}$$

We use the second smallest value as the lower bound of the negative domain, rather than the smallest value, to prevent the first knot from being placed between the two smallest data points as no useful information can be learned in that region.

Similarly, let $\bar{e}_{max}$ and $\bar{e}_{nmax}$ denote the largest and second largest values in the strain data. With $x_l = 0$ and $x_r = \bar{e}_{nmax} > 0$, by the same reasoning, the positive knots are determined by

$$t_0^+ = t_{-1}^+ = t_{-2}^+ = t_{-3}^+ = \bar{e}_{max}$$
$$t_i^+ = 0 + (\bar{e}_{nmax} - 0) \cdot \sum_{j=0}^{i-1} softmax(w_j^+) \quad \text{for } i = 1,2,\dots,n_2 \tag{12}$$

As noticed, the weights w_i^- and w_i^+ here basically shift the positions of the knots. The final ordered set of knots then becomes Eq. (13).

$$\left\{\bar{e}_{min} = t_{-3}^- = \cdots = t_0^- < t_1^- < \cdots < t_{n_1}^- < t^o = 0 < t_1^+ < \cdots < t_{n_2}^+ < t_0^+ = \cdots = t_{-3}^+ = \bar{e}_{max}\right\} \tag{13}$$

3.1.3 Linear layer

From the knot controller, we have collected $(n_1 + n_2 + 9)$ knots in total. In a cubic spline setting, we recall five consecutive knots determine one B-spline basis function. Thus, we need to find $n_1 + n_2 + 5$ coefficients. We label the learnable parameters as follows.

$$v_{-1}^-,\ v_0^-,\ v_1^-,\ \dots,\ v_{n_1}^-,\ v^o,\ v_1^+,\ \dots,\ v_{n_2}^+,\ v_0^+,\ v_{-1}^+ \tag{14}$$

where each $v_i^\cdot$ is associated with the B-spline basis function $B_i^\cdot(e)$, which is defined by five consecutive knots centered at t_i.

We renumber the weights in Eq. (14) collectively as v_i in sequential order, $i = 1, \dots, n_1 + n_2 + 5$, and do the B-spline basis functions $B_i(e)$ in the same way. Using this notation, we can write the output of the network $\mathbf{NN}(e)$ as

$$\mathbf{NN}(e) = \sum_{i=1}^{n_1+n_2+5} v_i B_i(e) \tag{15}$$

This expression follows the structure of the spline representation in Eq. (5) by identifying the learnable weights v_i with the spline coefficients c_i. Thus, the output in Eq. (15) approximates the Cauchy stress $\tau(e)$ at the input strain e.

As noted in Section 3.1.1., the network must satisfy the following constraint

$$\sum_{i=1}^{n_1+n_2+5} v_i B_i(e=0) = 0 \tag{16}$$

which reflects the stress-free state at zero strain. Because each basis function has compact support over five knots and vanishes at the endpoints, $B_i(e=0)$ only has nonzero values when whose second, third, and fourth knots are at the origin. Therefore, Eq. (16) can be rewritten as

$$v^-_{n_1} B^-_{n_1}(0) + v^o B^o(0) + v^+_1 B^+_1(0) = 0 \tag{17}$$

We can solve Eq. (17) and find the value of the coefficient v^o that satisfies the zero-passing condition. As a result, the liner layer has $(n_1+2)+(n_2+2)$ learnable parameters.

3.2 Training procedure

The training process involves determining the optimal number and values for the weights $\{w_i\}$ in the knot controller and $\{v_i\}$ in the linear layer. The number of knots may be reduced in the solution process through a clustering algorithm in dynamic layer updates. The values of weights are optimized through standard gradient-based optimization.

A schematic diagram of the overall process is given in Fig. 3, and we discuss the details of the solution procedure in the next sections.

3.2.1 Data preparation

Our neural network model requires the logarithmic strain as input and the Cauchy stress as output. Therefore, each pair of uniaxial test data must be checked and

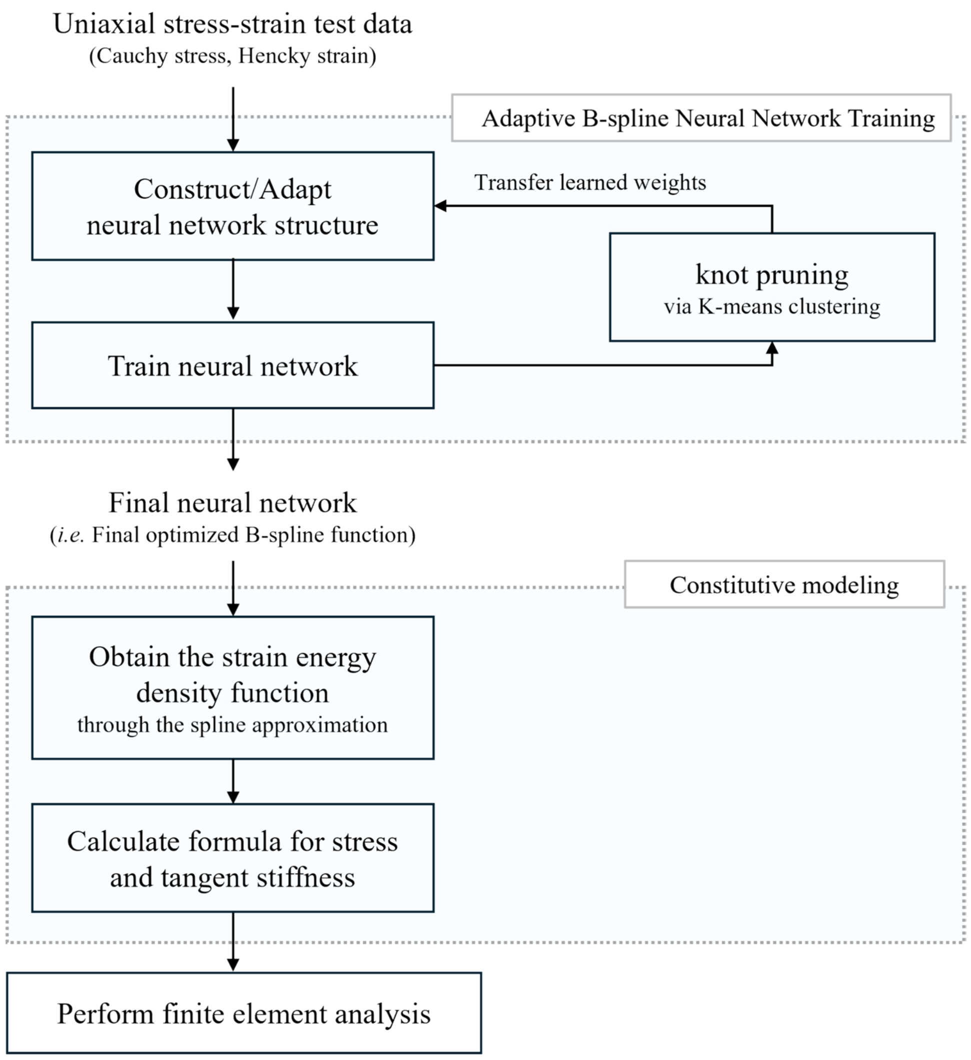

Uniaxial stress-strain test data
(Cauchy stress, Hencky strain)
Adaptive B-spline Neural Network Training
Construct/Adapt neural network structure
Transfer learned weights
knot pruning via K-means clustering
Train neural network
Final neural network
(i.e. Final optimized B-spline function)
Constitutive modeling
Obtain the strain energy density function through the spline approximation
Calculate formula for stress and tangent stiffness
Perform finite element analysis

Fig. 3 Overall process of using the B-Spline-based constitutive neural network – from input data to performing finite element analysis

possibly preprocessed to correspond to logarithmic strain and Cauchy stress data, and the data should be sorted in ascending order of the strain values. That is, for each pair of data (e_i, τ_i), we should have $e_i < e_{i+1}$ for all i. As indicated in Eqs. (11) and (12), the two smallest and two largest values in the strain data must be specified to determine the domain of the knots.

3.2.2 Initialization

To provide a stable starting point for training, we want to assign the initial number of knots based on data and place them uniformly. The number of internal knots in each of the negative and positive domains is set as $n_1 = \min(30,\ n_{\text{neg}} + 1)$ and $n_2 = \min(30,\ n_{\text{pos}} + 1)$, where n_{neg} and n_{pos} denote the number of data points, respectively, within each of the two strain domains, see Section 3.1.2. These formulas were chosen to have an effective scheme with the "30" as an upper bound to also prevent too fine a representation. The uniform placement of knots is prepared through zero initialization of the weights $\{w_i\}$ in the knot control layer.

The weights $\{v_i\}$ are initialized reflecting the global trend of stress–strain relationships, which generally exhibits a monotonically increasing behavior in typical loading conditions. Hence, we initialize the weights to evenly divide the stress range.

3.2.3 Loss function

The purpose of the training process is to minimize the difference between the predicted stress and the actual stress. We use the mean squared error (MSE) as the loss function

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\overline{\tau}_i - \mathbf{NN}\left(\overline{e}_i\right)\right)^2 \tag{18}$$

where N is the total number of data points, $\overline{\tau}_i$ is the actual stress at the i-th strain data point, and $\mathbf{NN}\left(\overline{e}_i\right)$ is the predicted stress at that data point. No normalization is applied to the stress values when computing the error.

3.2.4 Dynamic layer reconfiguration

Once the positions of the knots have been trained, the solution procedure should reduce the number of knots if they are deemed too close to each other to avoid overfitting. To achieve the reduction in knots, the distances between neighboring knots are calculated, and the distances are clustered into several groups using the K-means algorithm [28, 29]. When two neighboring knots are identified as very close through the clustering process, the two knots are merged into a single knot by averaging their positions. After iterating over all pairs of selected points, the layers and the entire structure of the network are reconfigured accordingly.

We found choosing the number of clusters as four is a reasonable choice, which provided a good balance between over- and under-simplification. Using too few clusters (e.g., $k \leq 3$) may result in excessive knot removal, potentially degrading the expressiveness. Conversely, choosing too many clusters (e.g., $k \geq 5$) may focus only on small-distance outliers, resulting in a negligible reduction. Of course, the number of clusters used can be adjusted based on the specific dataset and the desired level of detail in the stress–strain curve representation. However, $k = 4$ consistently produced stable and meaningful simplifications in our experiments. Empirically, repeating this dynamic layer update process twice tends to produce a good result.

3.2.5 Training setup

We used the deep learning framework PyTorch [31] for the neural network implementation. In our NN model, we minimize the MSE loss function through ADAM (Adaptive Moment Estimation) [30] in a full-batch setting. The parameter updates were achieved iteratively through the automatic differentiation engine of PyTorch.

The model was trained in three stages with progressively refined knot configurations and decreasing learning rates. In the first stage, the model was trained for 100 epochs using the ADAM optimizer, with a learning rate of 0.2 for the knot controller and 0.1 for the linear layer. Separate learning rates were assigned based on empirical observations to balance the speed of knot adjustment and coefficient optimization.

The model was then retrained for additional 100 and 150 epochs in the second and third stages, respectively, with reduced learning rates of 0.1 and 0.05 for both layers. After each stage, the network was reconstructed by updating the layer configuration as described in Section 3.2.4. We found that this progressive reduction stabilizes the optimization process by avoiding overshooting and improves convergence.

Additional observations regarding the convergence of the training process and the strategy to stop the process are discussed in Appendix C. We should note that there is no separate testing performed for accuracy evaluation as in the usual use of NN, that is, the solution procedure uses all given data for training. Once the NN scheme converges to represent the given data, the constitutive representation has been established.

4 Illustrative examples

Our objective is to present here example analyses to illustrate the performance of our proposed model.

In the analyses, we use for comparison the extended Mooney-Rivlin model, one of the most widely-used models of hyperelastic materials. The standard Mooney-Rivlin model is a two-term polynomial model with respect to the invariants of the deformation tensor, and an extended Mooney-Rivlin model has additional higher order terms to capture the higher nonlinearity of the stress–strain curve more accurately. The strain energy function of the extended Mooney-Rivlin model is given by

$$\begin{aligned} W(I_1, I_2) = {} & C_1 \cdot (I_1 - 3) + C_2 \cdot (I_2 - 3) \\ & + C_3 \cdot (I_1 - 3)^2 + C_4 \cdot (I_1 - 3) \cdot (I_2 - 3) + C_5 \cdot (I_2 - 3)^2 \\ & + C_6 \cdot (I_1 - 3)^3 + C_7 \cdot (I_1 - 3)^2 \cdot (I_2 - 3) + C_8 \cdot (I_1 - 3) \cdot (I_2 - 3)^2 + C_9 \cdot (I_2 - 3)^3 \end{aligned} \tag{19}$$

where I_1 and I_2 are the first and second invariants of the deformation tensor, respectively. The coefficients C_i are the material parameters to be determined by fitting the model to the given data.

We used ADINA for our finite element analyses [21]. The program provides a user-coded material subroutine that is called at every solution step and every Gauss integration point of the finite elements in the mesh. Given the strain values calculated in ADINA in each iterative step of analysis, the corresponding stresses and tangent stiffness values need to be evaluated at each Gauss point through the user-defined subroutine. In the case of using the Mooney-Rivlin model, the stresses and tangent stiffness values are directly obtained by differentiation using Eq. (19) [1].

In our use of the NN we proceed as follows. We first train the NN as described above to represent the given stress–strain data set. The material response is then given in closed form through the set of piecewise cubic polynomials established in Section 3. We next use this material model representation and hard code it as the representation of the material response for each principal Hencky strain direction.

Using this code for the stress–strain representation, we can directly calculate, at any strain value given by ADINA, the principal stresses and, by differentiation, the tangent stiffness values. In conventional settings of NN-based constitutive models, the stress and tangent stiffness tensor are computed by using the network through forward passes and computing derivatives through automatic differentiation at each Gauss point for *every* Newton iteration. In contrast, our model is fully interpretable and admits a closed form representation in terms of the piecewise cubic polynomials used. The subroutine for these calculations to obtain the stresses and tangent stiffness values was written in Fortran and integrated into the ADINA user-coded material model framework.

Therefore, in essence, the set of piecewise cubic polynomials established in our NN replaces the classical closed-form expressions, like the Mooney-Rivlin model in Eq. (19), to model a hyperelastic material in a finite element analysis.

We present in Section 4.1 how two sets of material data are represented by our model. Thereafter we provide two examples of finite element analyses using our constitutive model in ADINA, see Sections 4.2 and 4.3.

4.1 Material representations

In this section, we focus on the material representations reached using our model and the Mooney-Rivlin model in order to illustrate the capability of our NN constitutive model.

We consider generalized Mooney-Rivlin models using three different polynomial orders, denoted as MR_2, MR_5, and MR_9. Here, MR_n refers to the Mooney-Rivlin model with the first n terms in Eq. (19). For example, MR_2 is the standard Mooney-Rivlin model that retains only the first two terms. While other procedures can be employed, we use a least-squares approach to find the Mooney-Rivlin coefficients throughout this study. Adopting the least-squares method allows for a fair comparison, as it gives the solution that minimizes the mean squared error (MSE) (Eq. (18)) by design. The scipy.optimize module in Python was used for the least-squares fitting.

Figure 4 shows the uniaxial response data to be represented in the constitutive models.

The results obtained for each dataset in Fig. 4 are presented in the following sections.

4.1.1 Typical hyperelastic material—Treloar's rubber

As shown in Fig. 4, the benchmark dataset published by Treloar of vulcanized rubber at 20 °C includes both uniaxial tension and compression, see ref. [4]. We digitized the data given by Treloar in ref. [4] to obtain Fig. 4. Using these data, we reach the results using the Mooney-Rivlin models and the B-Spline-based constitutive neural network shown in Fig. 5. We see that excellent representations of the original data are obtained.

4.1.2 Hyperelastic dataset with tensile data of an artery

To further demonstrate the flexibility of our model, we considered a dataset of an artery.

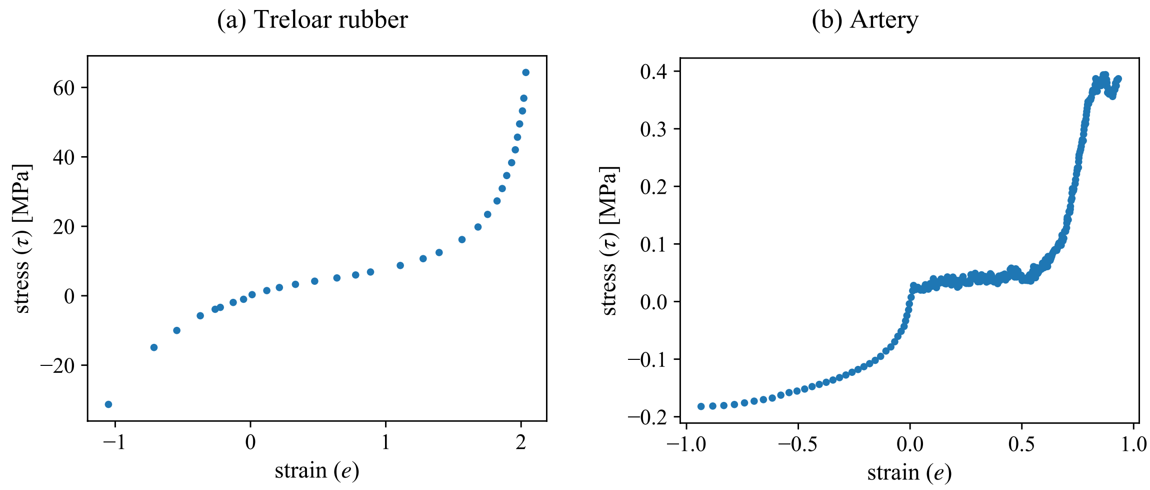


Fig. 4 Uniaxial stress–strain test datasets

Since only the tensile data was measured experimentally [32], we supplemented it with a simple but reasonable rubber-like dataset to represent the compressive region, see Figs. 4 and 6. This dataset is more difficult to represent due to the simple behavior in compression and complicated behavior in tension with the presence of localized fluctuations.

The results obtained are shown in Fig. 6. The imbalance of the data distribution impairs the performance of the Mooney-Rivlin models because the MR_5 and MR_9 models focus primarily on capturing the behavior of the tensile data, where the data points exhibit high density. The attempts to fit the tensile data leads to a rather inaccurate representation in the compressive region. However, our approach achieves better results throughout the stress–strain data.

4.2 Patch tests

The patch test is a standard test to verify an element formulation or a solution scheme as in dynamics, [1, 2, 33]. Considering the formulation of elements, satisfying the patch test ensures that the elements reproduce the uniform stress fields imposed by constant boundary tractions with minimal displacement boundary conditions and converge in complex simulations of structures. As suggested in ref. [1], we use a square patch of elements with a uniform loading in the x -direction, y-direction and corresponding to simple shear. The distorted mesh of four-node elements used in the test is shown in Fig. 7. The patch test is said to be passed if the stress values in the elements in the deformed configuration correspond to the exact analytical solution.

Our objective here is not to test the formulation of the elements but instead whether the material representation in our constitutive model and the implementation are correct. The elements in ADINA are formulated to satisfy the patch test and hence we can proceed using that program to test our NN-based constitutive model. We use the rubber data of Fig. 4(a) and the results obtained using our NN model are presented in Figs. 8 to 11. Figures 8 and 9 show the deformed meshes for a force of 12.50 N applied to each of the two points of loading. The calculated stress values are equal to the analytical value.

Usually, the patch test is only performed in infinitesimally small strain conditions because focus is on the element formulation. However, to test our constitutive NN model and its implementation it is necessary to perform the patch test with larger stretches such as we used. The stress–strain responses are plotted in Fig. 10 which shows that excellent results are obtained, including using the standard Mooney-Rivlin model in ADINA with the parameter set referred to as MR_9. The result of the shear patch test is given in Fig. 11. In this case we considered only small deformations because imposing shearing tractions for large deformations would result in complex stress distributions within the elements.

Having performed the above tests, and obtained good results, we can consider our NN constitutive model and its implementation to give accurate and reliable representations of incompressible hyperelastic material response.

4.3 Analysis of a plate with a hole

To further assess the capability of our NN model, we perform a finite element simulation of a plate with a hole, which is a common benchmark problem to test finite element methods. In this simulation, 2D plane stress conditions are assumed and using symmetry only one-quarter of the plate is modeled. The material of the plate is that of the artery shown in Fig. 4(b), and the initial thickness of the plate is 1.0. The plate is subjected to a uniform tensile load in the y

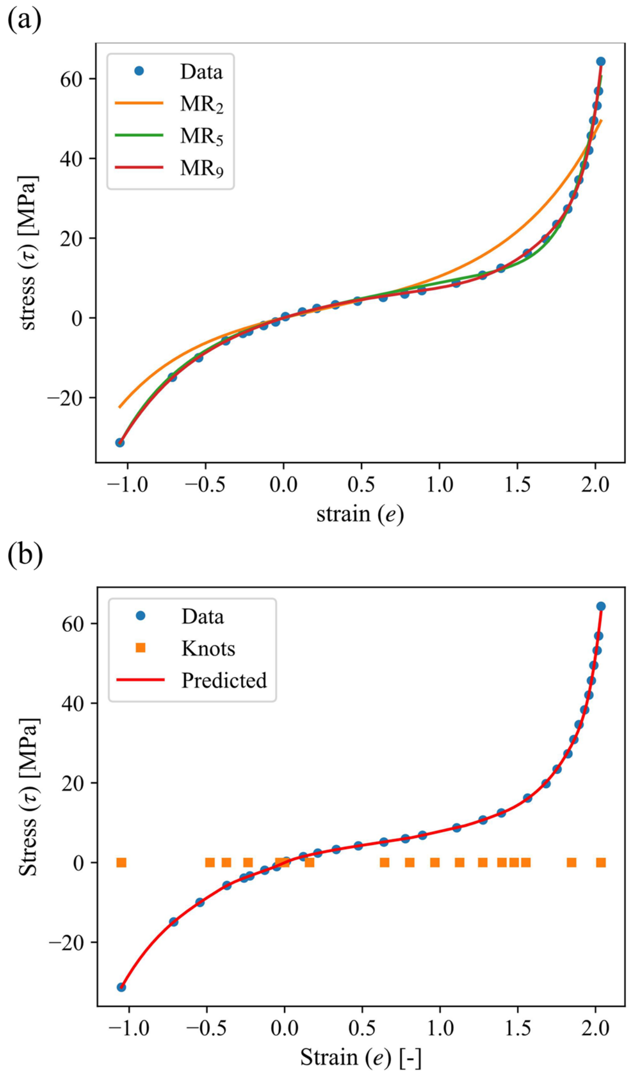


Fig. 5 The results using the Mooney-Rivlin model and the B-Spline-based constitutive neural network on Treloar's vulcanized rubber data. (**a**) Using the Mooney-Rivlin models, and (**b**) Using our B-Spline-based constitutive neural network

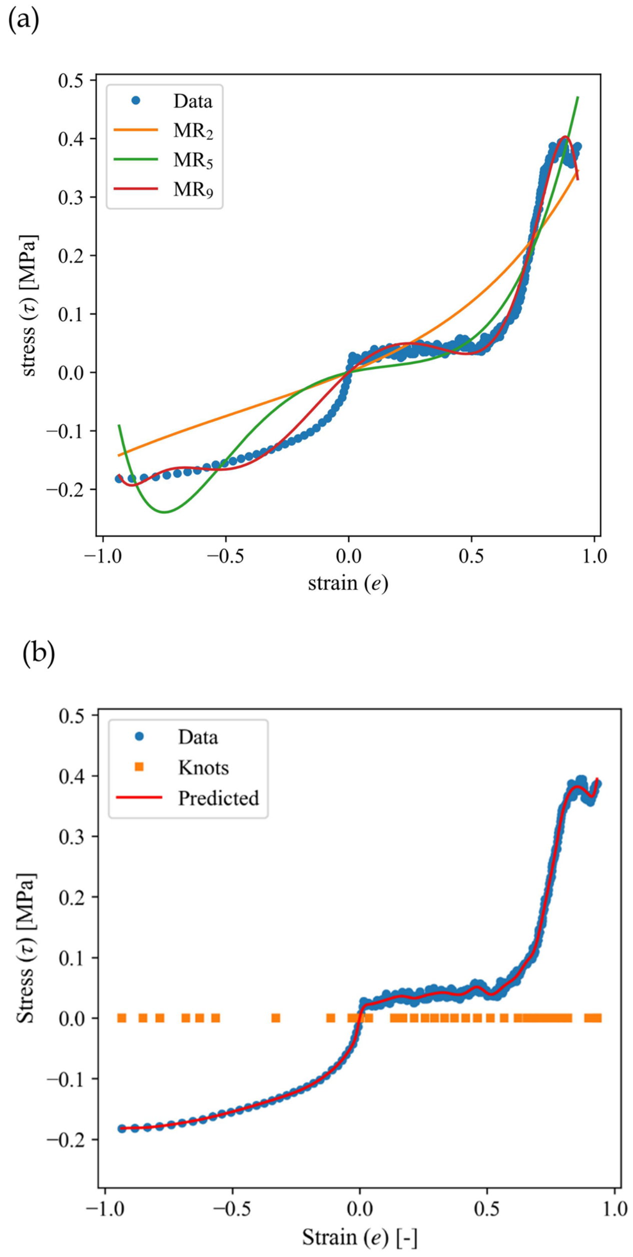


Fig. 6 The results using the Mooney-Rivlin model and the B-Spline-based constitutive neural network for the dataset with a peak and wiggles. (a) Using the Mooney-Rivlin models, and (b) Using our B-Spline-based constitutive neural network

-direction through the imposed displacements at the ends of the plate as depicted in Fig. 12.

Our objective is to compare the performance of our NN model with a Mooney-Rivlin model in this finite element simulation. We used the Mooney-Rivlin coefficients obtained from the fitting in Fig. 6. From an engineering perspective, the MR_2 model might be the best choice considering the three Mooney-Rivlin models. The MR_5 model shows an inadequate representation of the behavior in compression, while the MR_9 model being more complex is closer to the experimental data. However, the slope of the stress–strain curve of the MR_9 model becomes negative at the tensile end which may cause the extrapolation to yield a non-physical behavior. Thus, also for simplicity, we used the MR_2 model in this simulation.

To trace the nonlinear response with ADINA, we employed for the solution of the nonlinear finite element equations the Newton–Raphson method with the line search strategy, the energy tolerance ETOL = 0.0001, force tolerance FTOL = 0.001, and the maximal number of iterations MAXITER = 100. Also, we employed the automatic time-stepping scheme to have stability and convergence during the solution process. In general, the Newton–Raphson iterations converged within 20 iterations per load step. The automatic

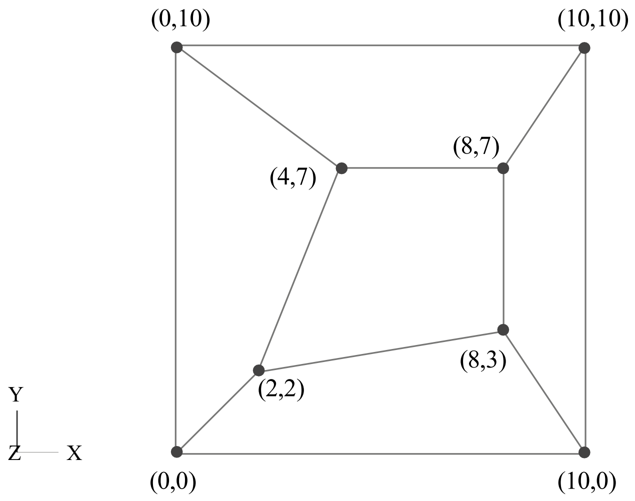


Fig. 7 Distorted mesh of a square patch test suggested in ref. [1]

time-stepping algorithm adjusted the global increment size dynamically in response to local nonlinearity, while the line search strategy complemented this procedure by scaling the search direction in each Newton iteration step locally. Together, these two procedures help to maintain global and local convergence robustness.

Figure 13 shows the mesh used and the results obtained. All elements are nine-node quadrilaterals except for one seven-node triangle, with a total of 74 elements. The von Mises stress band contours correspond to the quarter plate stretched by a displacement of 6.2 units. We see that our constitutive NN shows localized stress fluctuations not seen when the Mooney-Rivlin model is employed.

The fluctuations in the stress do not remain fixed but propagate with the loading from near the hole boundary where the stress concentration naturally occurs. They then gradually move upward into the plate body.

These phenomena are not numerical artifacts but rather reflect the material behavior, that is, the underlying non-smooth features of the data. The model exhibits localized variations in stress response within the stress range of approximately 0.02 to 0.05 in Fig. 6. While the MR_2 model shows a monotonically increasing stress behavior, our model captures the main trend while preserving slight stress fluctuations. At the loading stage shown in Fig. 13,

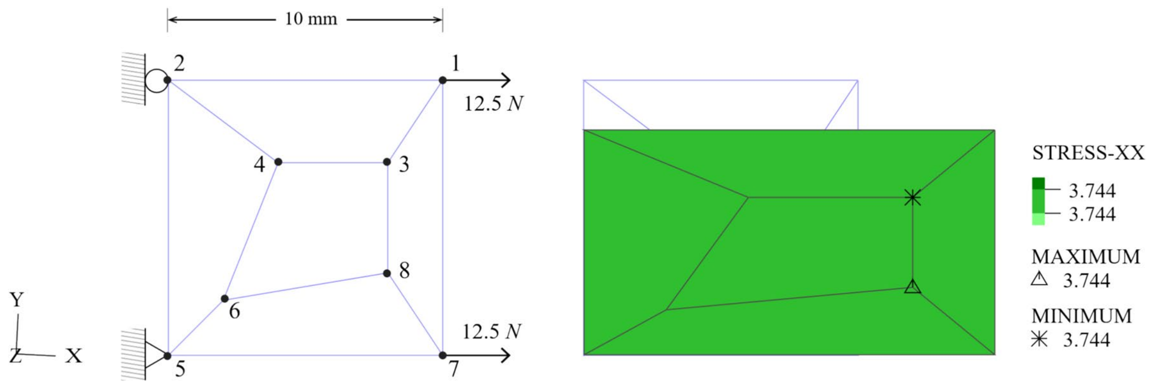
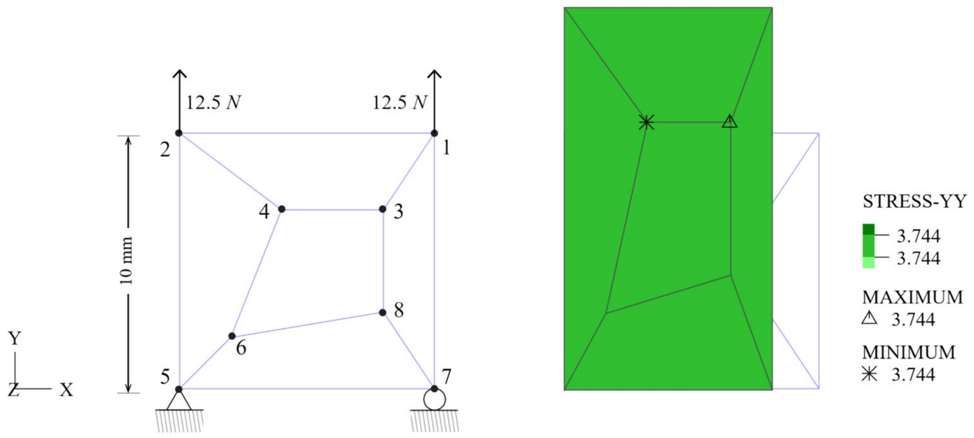


Fig. 8 Patch test schematic and the deformed configuration using our NN constitutive model. The xx-stress values are identical to the analytical values. All other stresses are very small

Fig. 9 Patch test schematic and the deformed configuration using our NN constitutive model. The yy-stress values are identical to the analytical value. All other stresses are very small



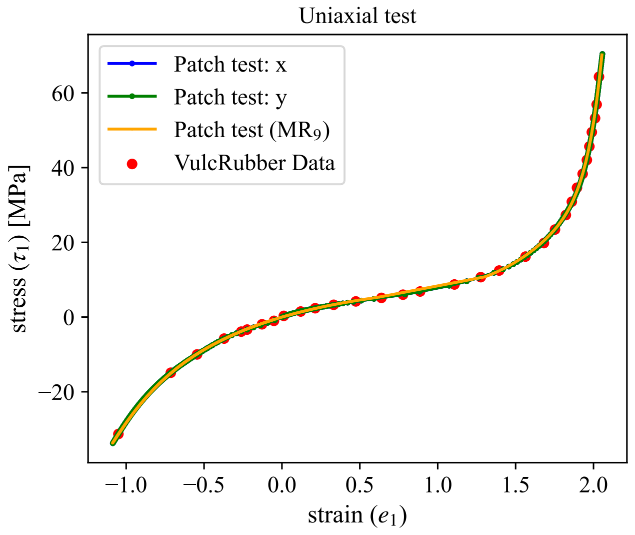


Fig. 10 The stress–strain data for vulcanized rubber used and predicted in the patch tests. The results using our NN constitutive model for the x and y-directions and the results using the standard MR_9 Mooney-Rivlin model in ADINA

the von Mises stress falls into the critical stress range, which is between 0.02 and 0.05, highlighted in the color bar of Fig. 13. This correlation tells us that our model captures the local stress variations and the fine features quite well.

Finally, it is valuable to also solve the problem by applying tensile tractions at the plate end (instead of displacements). We found that the main observations given above do not change.

These findings reveal the practical impact of material modeling choices on nonlinear simulations. The plate with a hole simulation demonstrates the potential of our model to meaningfully reproduce fine-scale material characteristics in practical engineering problems. We realized also that when using the existing Sussman-Bathe model, convergence difficulties are encountered in the solution of this problem if the raw dataset of Fig. 6 is used directly. These observations suggest that our approach effectively generalized the Sussman-Bathe model, and extends its applicability to more complex, non-preprocessed data.

5 Concluding remarks

We proposed a novel neural network inspired by the Sussman-Bathe model for hyperelastic materials. The essence of our development is that we modified the spline-based interpolation of the Sussman-Bathe model from interpolation to regression using B-splines in the neural network representation. Hence our neural network scheme inherits and extends the advantages of the Sussman-Bathe model. An important asset is that in the regression our procedure establishes a curve that is twice differentiable and minimizes the difference between the predicted and given values of any physically reasonable test dataset. Using a regression scheme allows us to track the major trends and even localized peaks while ignoring small non-physical fluctuations. The use of the regression scheme also resolved an overfitting issue that an interpolation can encounter.

We used our constitutive model based on the proposed NN within ADINA and solved the patch test to test the model in large longitudinal strain conditions. This is an important test to pass before a solution scheme can be recommended for engineering use in practice. We also illustrated how our NN model captures "peak and valley" details in supplied test data which can be important to solve physical problems accurately. A user of our model needs to only supply measured test data as input and hence need not decide which formula-based hyperelastic model (like a Mooney-Rivlin model) best fits the data. The NN gives the analytical B-spline representation of the test data, which can then be directly coded into the finite element implementation. Hence, our model is used in a finite element program quite similarly to how classical material models are employed.

To illustrate the use, we solved a typical benchmark problem (a plate with a hole) with a complex hyperelastic

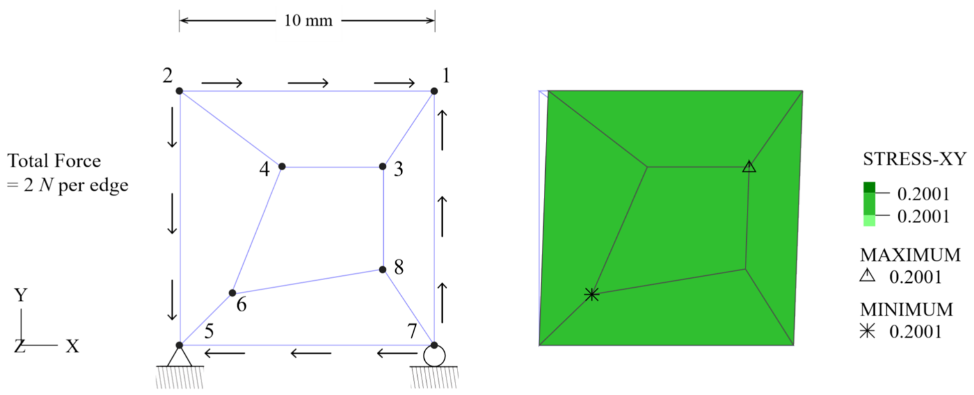


Fig. 11 Patch test schematic and the deformed configuration. The shear stress values are very close to the analytical value and the other stresses are very small

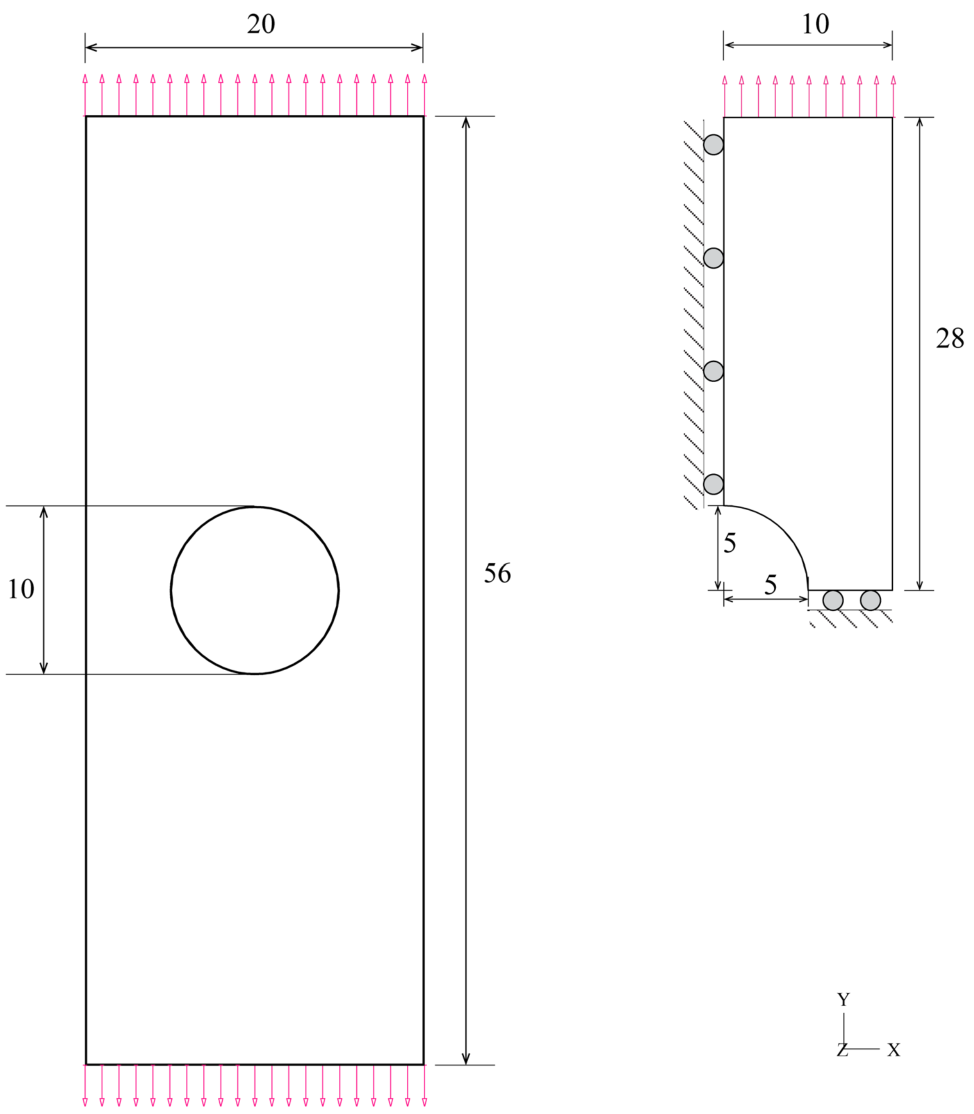


Fig. 12 Schematic of the plate with a hole problem. A quarter of the plate is modeled using symmetry boundary conditions. The plate is subjected to a uniform tensile load in the y-direction by imposing displacements

material behavior not captured using classical material models and obtained very valuable results. We can conclude that our NN procedure used in a finite element program like ADINA can be an important option in representing material behaviors in finite element simulations.

However, further research and developments to employ NN to model hyperelastic material behaviors would be valuable. Our use of the NN focused primarily on the effective use of the optimization procedures available in the proposed NN to reach a relatively simple but general constitutive model of incompressible hyperelastic material behavior. With the wealth of related developments available, see e.g. T. Rabczuk and K.J. Bathe [34], a more encompassing NN based on additional developments and effective use of optimization schemes may be reached. Also, considering hyperelastic material behavior, there is a need to include compressibility and non-isotropic effects. Furthermore, more complex material behaviors should be tackled using NN, for example the behavior of concrete materials. Hence the research presented in this

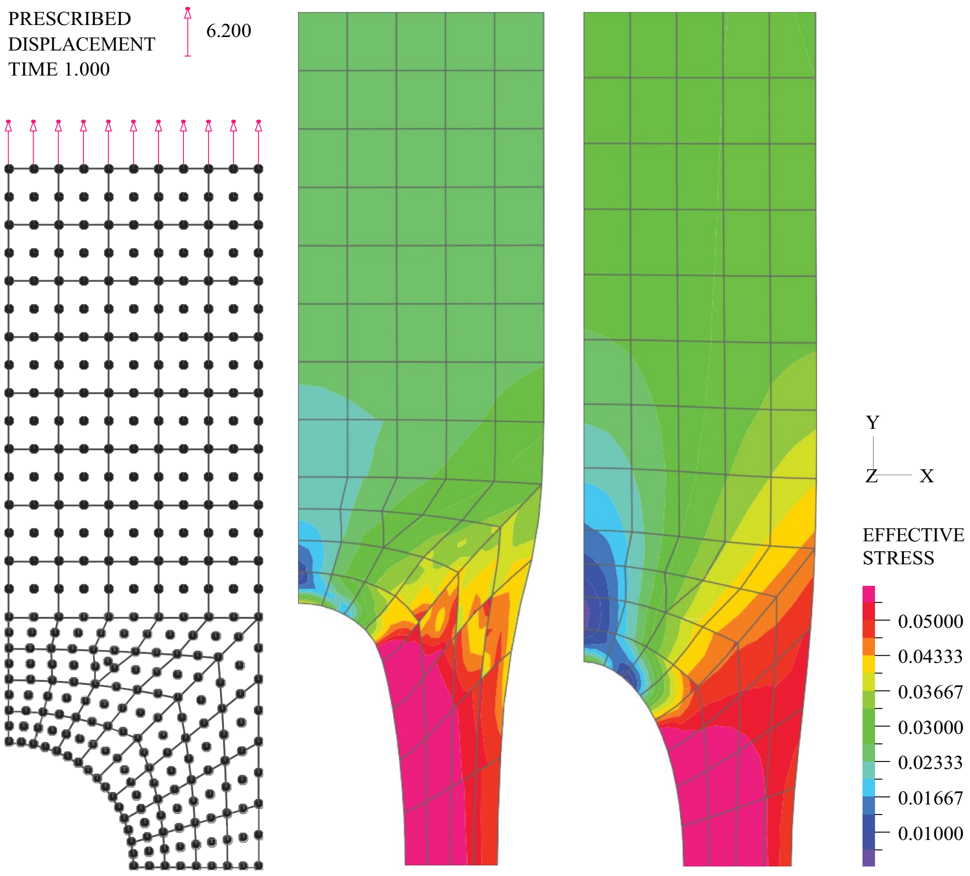


Fig. 13 The plate with a hole problem. The left figure shows the original mesh, the middle figure shows the deformed mesh and predicted effective stress using our NN constitutive model, and the right figure shows the results obtained using the Mooney-Rivlin model in ADINA

paper, apart from proposing a specific NN based constitutive model, might also be seen as opening an avenue for further research on modeling various material behaviors with NN procedures.

Appendix A Sussman-Bathe model

Considering pure uniaxial tension–compression, we have $e_1 = e,\ e_2 = e_3 = -\frac{1}{2}e$. Then, from Eq. (2), we have

$$\tau_1 = w'\left(e_1\right) + p_H \tag{20}$$

$$\tau_2 = w'\left(e_2\right) + p_H \tag{21}$$

Since only the first direction is subjected to direct external loading, $\tau_2 = 0$. Thus, we obtain from Eq. (21)

$$p_H = -w'\left(e_2\right) \tag{22}$$

And with Eq. (20) we have

$$\tau_1 = w'\left(e_1\right) - w'\left(e_2\right) \tag{23}$$

We use $\tau_1 = \tau(e)$ and obtain Eq. (3).

The Sussman-Bathe model defines the energy function $w(e)$ in terms of $\tau(e)$ since Eq. (4) satisfies Eq. (3):

$$\begin{aligned} w'(e) - w'\left(-\frac{1}{2}e\right) &= \sum_{k=0}^{\infty} \tau\left(\left(-\frac{1}{2}\right)^k \cdot e\right) - \sum_{k=0}^{\infty} \tau\left(\left(-\frac{1}{2}\right)^{k+1} \cdot e\right) \\ &= \left[\tau(e) + \sum_{k=1}^{\infty} \tau\left(\left(-\frac{1}{2}\right)^k \cdot e\right)\right] - \sum_{k'=1}^{\infty} \tau\left(\left(-\frac{1}{2}\right)^{k'} \cdot e\right) \\ &= \tau(e) \end{aligned} \tag{24}$$

Hence, with τ obtained from the uniaxial test data, $w'(e)$, the derivative of the energy function, can be recovered using Eq. (4). Although the strain energy function W can be calculated by integrating the function $w'(e)$ and summing as in Eq. (1), the integration is not necessary because only the derivatives appear in Eqs. (20) and (21) for computing stress values.

Appendix B Example of a B-Splines basis function

We provide here an example derivation of a B-Spline basis function. A step-by-step derivation of the B-Spline basis function $B_{i,p}$ using the Cox-de Boor recursion formula is given, as stated in Eq. (6).

Appendix B.1 Distinct knots

Consider a case of five distinct knots $t_1 < t_2 < t_3 < t_4 < t_5$. To obtain a B-spline basis function $B_1(x) = B_{1,3}(x)$ of degree $p = 3$, we start with the lowest order $k = 0$:

$$B_{i,0}(x) = \begin{cases} 1 \text{ if } t_i \leq x < t_{i+1} \\ 0 \text{ otherwise} \end{cases} \quad i = 1, 2, 3, 4 \tag{25}$$

Next, we recursively compute $B_{i,k}(x)$ for $k = 1, 2, 3$:

$$\begin{aligned} B_{i,1}(x) &= \frac{x-t_i}{t_{i+1}-t_i} B_{i,0}(x) + \frac{t_{i+2}-x}{t_{i+2}-t_{i+1}} B_{i+1,0}(x) \quad i = 1, 2, 3 \\ B_{i,2}(x) &= \frac{x-t_i}{t_{i+2}-t_i} B_{i,1}(x) + \frac{t_{i+3}-x}{t_{i+3}-t_{i+1}} B_{i+1,1}(x) \quad i = 1, 2 \\ B_{i,3}(x) &= \frac{x-t_i}{t_{i+3}-t_i} B_{i,2}(x) + \frac{t_{i+4}-x}{t_{i+4}-t_{i+1}} B_{i+1,2}(x) \quad i = 1 \end{aligned} \tag{26}$$

We note that $B_{i,0}(x)$ is a unit function on the interval $[t_i, t_{i+1})$, and $B_{i,1}(x)$ is a hat function defined on the interval $[t_i, t_{i+2})$.

Appendix B.2 Repeated knots

If knots are repeated in the knot vector, that is, $t_j = t_{j+1}$ for some j, the smoothness of the B-spline basis function is affected. If $t_{i+k} = t_i$ for some i and k, we have a denominator equal to zero, which could introduce numerical instability. To resolve this issue, the following convention is adopted

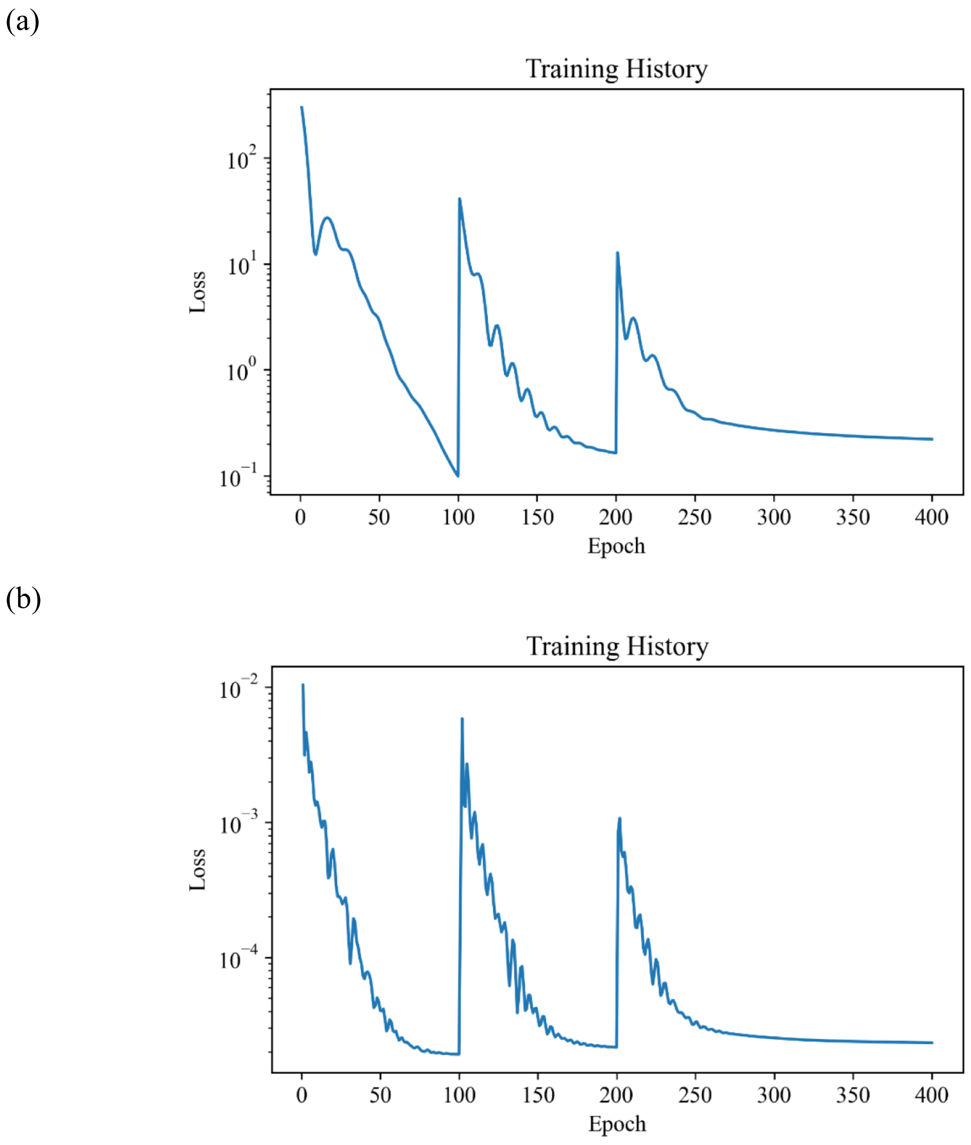


Fig. 14 Plots of training history showing the loss without normalization for (**a**) The rubber example and (**b**) The artery example

$$\frac{x - t_i}{t_{i+k} - t_i} = 0 \quad \text{if} \quad t_{i+k} = t_i \tag{27}$$

where $k > 0$.

We note that a basis function has nonzero support only over the interval spanned by its defining knots. Since $t_i = \cdots = t_{i+k}$, the interval collapses to a single point and $B_{i,k-1}(x)$ becomes zero everywhere by definition. By setting the coefficient zero, we ensure that the entire term involving such basis functions (*e.g.* $B_{i,k-1}(x)$ with $t_i = \cdots = t_{i+k}$) vanishes and we maintain the local influence within the interval produced by distinct knots. As a result, one repetition of knots reduces the differentiability by one order.

In the extreme case, if a knot is repeated 4 times, the B-spline basis function generated by the the knots (and one additional knot) will exactly interpolate a data point at that knot. For example, if we are given $t_1 = t_2 = t_3 = t_4 < t_5$, we obtain.

$$\begin{aligned} B_{1,3}(x) &= 0 \cdot B_{1,2}(x) + \tfrac{t_5-x}{t_5-t_2} B_{2,2}(x) \\ B_{2,2}(x) &= 0 \cdot B_{2,1}(x) + \tfrac{t_5-x}{t_5-t_3} B_{3,1}(x) \\ B_{3,1}(x) &= 0 \cdot B_{3,0}(x) + \tfrac{t_5-x}{t_5-t_4} B_{4,0}(x) \\ B_{4,0}(x) &= \begin{cases} 1 & \text{if } t_4 \le x < t_5 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{28}$$

and, in turn, we obtain

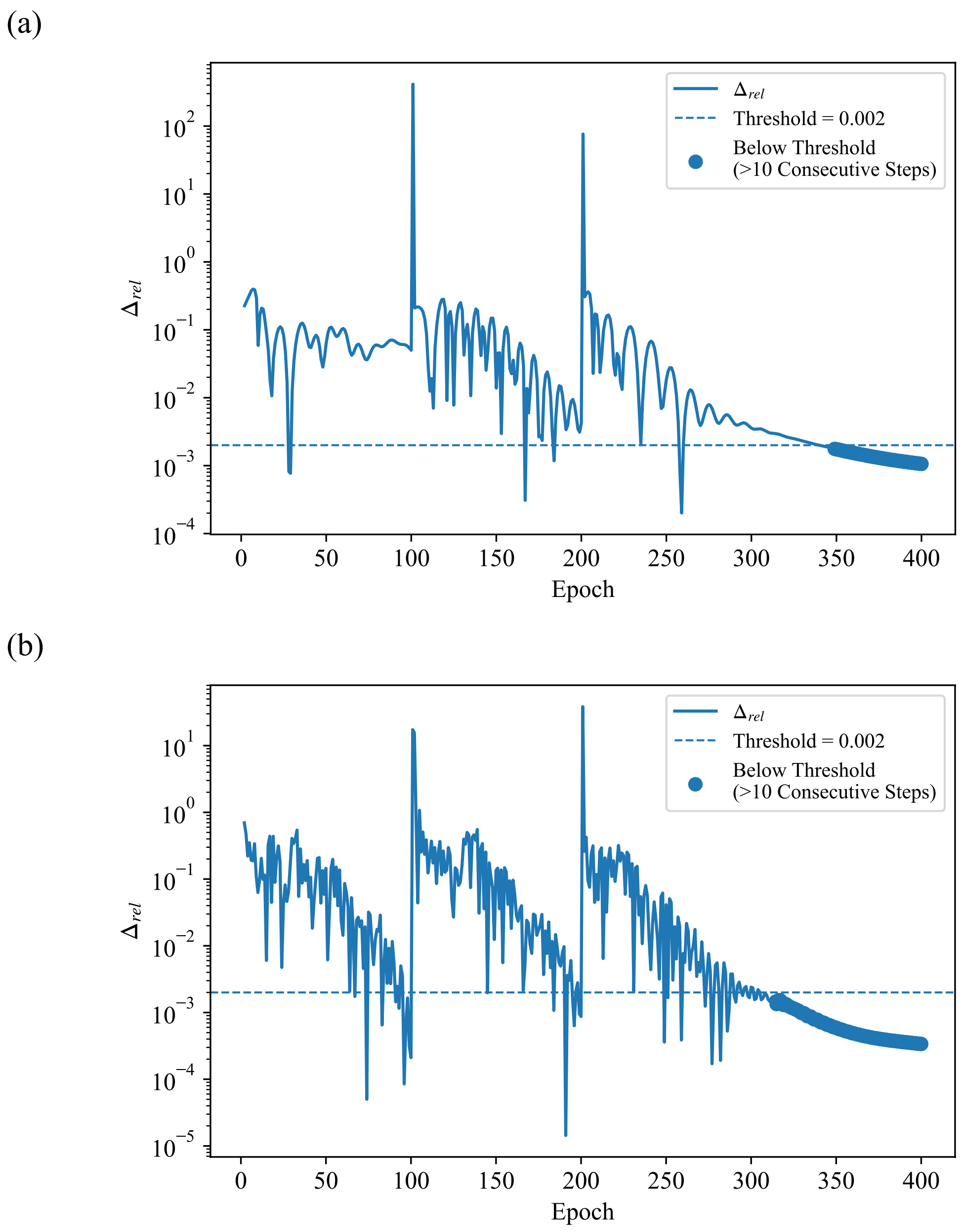


Fig. 15 Plots of relative loss change for (**a**) The rubber example and (**b**) The artery example

$$
\begin{aligned}
B_{1,3}(x) &= \frac{t_5 - x}{t_5 - t_2} B_{2,2}(x) \\
&= \frac{t_5 - x}{t_5 - t_2} \cdot \frac{t_5 - x}{t_5 - t_3} B_{3,1}(x) \\
&= \frac{t_5 - x}{t_5 - t_2} \cdot \frac{t_5 - x}{t_5 - t_3} \cdot \frac{t_5 - x}{t_5 - t_4} B_{4,0}(x) \\
&= \begin{cases} \left(\frac{t_5 - x}{t_5 - t_4}\right)^3 & \text{if } t_4 \le x < t_5 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{29}
$$

which yields $B_{1,3}(t_4) = 1$.

Appendix C. Criteria to stop the iteration

The convergence histories of each training in Sections 4.1.1 and 4.1.2 are shown in Fig. 14. The peaks correspond to the reconstructions of the network by updating the layer configurations. In both training cases, the loss curves begin to flatten around epoch 300, reflecting a reduced rate of learning progress.

The loss scales vary considerably between the two cases since the stress values in the original test datasets differ significantly in magnitude. To obtain a normalized rate of loss decrease we introduce the relative change defined by

$$
\Delta_{rel} = \frac{|L_t - L_{t-1}|}{|L_{t-1}|} \tag{30}
$$

where L_t denotes the training loss at epoch t. This relative change in loss is indicative of how much a loss has changed from one step to the next, relative to its previous loss value.

We can then define a convergence criterion based on this relative loss change. We consider the training as converged when the change remained below the threshold 0.002 for 10 consecutive epochs which occurred in our example material models at around epochs 350, see Fig. 15.

Author contribution KJ Bathe suggested the research and Sanghee Lee performed the work. Both authors collaborated closely, wrote, reviewed and edited the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interest The authors declare no competing interests.

References

1. Bathe KJ. Finite element procedures. First edition Prentice Hall, 1996; 2nd Edition 2014, KJ Bathe amazon.com; Higher Education Press China, 2016. Springer Verlag, in press.
2. Bathe KJ. Finite element procedures - En Plus. Springer Verlag, in press.
3. Mooney M. A theory of large elastic deformation. J Appl Phys. 1940;11:582–92. https://doi.org/10.1063/1.1712836.
4. Treloar LRG. Stress-strain data for vulcanised rubber under various types of deformation. Trans Faraday Soc. 1944;40:59. https://doi.org/10.1039/tf9444000059.
5. Rivlin RS. "Large elastic deformations of isotropic materials IV. further developments of the general theory." Philos Trans Royal Soc London Ser A Math Phys Sci. 1948;241:379–97. https://doi.org/10.1098/rsta.1948.0024.
6. Ogden RW. Non-linear elastic deformations. E. Horwood, 1984.
7. Arruda EM, Boyce MC. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. J Mech Phys Solids. 1993;41:389–412. https://doi.org/10.1016/0022-5096(93)90013-6.
8. Yeoh OH. Some forms of the strain energy function for rubber. Rubber Chem Technol. 1993;66:754–71.
9. Gent AN. A new constitutive relation for rubber. Rubber Chem Technol. 1996;69:59–61.
10. Sussman T, Bathe KJ. A model of incompressible isotropic hyperelastic material behavior using spline interpolations of tension–compression test data. Commun Numer Methods Eng. 2009;25:53–63. https://doi.org/10.1002/cnm.1105.
11. Latorre M, Montáns FJ. Extension of the Sussman-Bathe spline-based hyperelastic model to incompressible transversely isotropic materials. Comput Struct. 2013;122:13–26. https://doi.org/10.1016/j.compstruc.2013.01.018.
12. Montáns FJ, Cueto E, Bathe KJ. "Machine learning in computer-aided engineering", Chapter in machine learning in modeling and simulation. In: Rabczuk T, Bathe KJ, editors. Springer Nature, 2023. https://doi.org/10.1007/978-3-031-36644-4
13. Ghaboussi J, Garrett JH, Wu X. Knowledge-based modeling of material behavior with neural networks. J Eng Mech. 1991;117:132–53. https://doi.org/10.1061/(ASCE)0733-9399(1991)117:1(132).
14. Shen Y, Chandrashekhara K, Breig WF, Oliver LR. Neural network based constitutive model for rubber material. Rubber Chem Technol. 2004;77:257–77. https://doi.org/10.5254/1.3547822.
15. Linka K, Hillgärtner M, Abdolazizi KP, Aydin RC, Itskov M, Cyron CJ. Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning. J Comput Phys. 2021;429:Article 110010. https://doi.org/10.1016/j.jcp.2020.110010.
16. Anitescu C, Ates BI, Rabczuk T. "Physics-informed neural networks: Theory and applications". Chapter in machine learning in modeling and simulation. In: Rabczuk T, Bathe KJ, editors. Springer Nature, 2023. https://doi.org/10.1007/978-3-031-36644-4
17. Thakolkaran P, Joshi A, Zheng Y, Flaschel M, De Lorenzis L, Kumar S. NN-EUCLID: Deep-learning hyperelasticity without stress data. J Mech Phys Solids. 2022;169:Article 105076. https://doi.org/10.1016/j.jmps.2022.105076.
18. Linden L, Klein DK, Kalina KA, Brummund J, Weeger O, Kästner M. "Neural networks meet hyperelasticity: A guide to enforcing physics," J Mech Phys Solids, 2023;179:Article 105363. arXiv:2302.02403. https://doi.org/10.1016/j.jmps.2023.105363
19. Ghaderi A, Morovati V, Dargazany R. A physics-informed assembly of feed-forward neural network engines to predict inelasticity in cross-linked polymers. Polymers. 2020;12:Article 2628. https://doi.org/10.3390/polym12112628.
20. Linka K, Kuhl E. A new family of constitutive artificial neural networks towards automated model discovery. Comput Methods Appl Mech Eng. 2023;403:Article 115731. https://doi.org/10.1016/j.cma.2022.115731.

21. Bentley Systems, Inc. 2024. www.adina.com
22. Thakolkaran P, Guo Y, Saini S, Peirlinck M, Alheit B, Kumar S. Can KAN CANs? Input-convex Kolmogorov-Arnold Networks (KANs) as hyperelastic constitutive artificial neural networks (CANs). Comput Methods Appl Mech Eng. 2025; 443. https://doi.org/10.1016/j.cma.2024.118089
23. Abdolazizi K, Aydin RC, Cyron CJ, Linka K. Constitutive Kolmogorov–Arnold Networks (CKANs): Combining accuracy and interpretability in data-driven material modeling. J Mech Phys Solids 2025;203:106212. https://doi.org/10.1016/j.jmps.2025.106212
24. Dornheim J, Morand L, Nallani HJ, Helm D. Neural networks for constitutive modeling: from universal function approximators to advanced models and the integration of physics. Arch Comput Methods Eng. 2024;31:1097–127. https://doi.org/10.1007/s11831-023-10009-y.
25. Chung I, Im S, Cho M. "A neural network constitutive model for hyperelasticity based on molecular dynamics simulations," Int J Numerical Methods Eng. 2020;122. https://doi.org/10.1002/nme.6459
26. Mendizabal A, Márquez-Neila P, Cotin S. Simulation of hyperelastic materials in real-time using deep learning. Medical Image Analysis. 2019;59:Article 101569. https://doi.org/10.1016/j.media.2019.101569.
27. Jung J, Yoon K, Lee PS. Deep learned finite elements. Comput Methods Appl Mech Eng. 2020;372:Article 113401. https://doi.org/10.1016/j.cma.2020.113401.
28. Lloyd S. Least squares quantization in PCM. IEEE Trans Inf Theory. 1982;28(2):129–37. https://doi.org/10.1109/TIT.1982.1056489.
29. MacQueen JB. Some methods for classification and analysis of multivariate observations. University of California Press, 1967. pp. 281–297.
30. Kingma DP, Ba J. "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980v9, 2017.
31. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. "Pytorch: An imperative style, high-performance deep learning library," In: Wallach H, Larochelle H, Beygelzimer A, d'Alché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems 32. 2019, pp. 8024–8035, Curran Associates, Inc.
32. Parshin DV, Lipovka AI, Yunoshev AS, Ovsyannikov KS, Dubovoy AV, Chupakhin AP. On the optimal choice of a hyperelastic model of ruptured and unruptured cerebral aneurysm. Sci Rep. 2019;9:Article 15865. https://doi.org/10.1038/s41598-019-52229-y.
33. Noh G, Bathe KJ. Imposing displacements in implicit direct time integration & a patch test. Adv Eng Software. 2023;175:Article 103286. https://doi.org/10.1016/j.advengsoft.2022.103286.
34. Rabczuk T, Bathe KJ editors. Machine learning in modeling and simulation. Springer Nature. 2023. https://doi.org/10.1007/978-3-031-36644-4