RESEARCH ARTICLE OPEN ACCESS

HyperSym: An Educational MATLAB Code for Hyperelasticity

Vinicius Oliveira Fontes | André Xavier Leitão | Anderson Pereira

Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Rio de Janeiro, Brazil

Correspondence: Anderson Pereira (anderson@puc-rio.br)

Received: 27 August 2024 | **Revised:** 19 February 2025 | **Accepted:** 1 April 2025

Funding: This study was financed by Carlos Chagas Filho Foundation for Research Support of the State of Rio de Janeiro (FAPERJ), Process SEI 260003/008107/2022 and E-26/202.573/2022 (277082), the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Finance Code 001, and National Council for Scientific and Technological Development – Brazil (CNPq) (grant number 317319/2021-3).

Keywords: educational software | finite element method | hyperelasticity | MATLAB | symbolic differentiation

ABSTRACT

Engineering students may find two challenges while studying finite element-based structural analysis: the transition from linear to nonlinear analysis theory and implementing finite element algorithms. Unlike damage and plasticity, which require often complex return mapping algorithms to update internal variables, introducing material nonlinearity with hyperelasticity is simpler as the stress tensor therein is computed explicitly from a deformation measure. To simplify the derivation process, we present HyperSym, an educational MATLAB-based tool that leverages symbolic differentiation to derive hyperelastic tensors from the strain energy density functional and automatically generate ready-to-use functions. We integrate these functions into the educational open-source finite element software NLFEA to illustrate the connection between user-defined subroutines and a finite element framework often found in commercial packages. This paper outlines HyperSym's core features and demonstrates its educational potential through numerical examples applicable to lecture and homework settings. Lastly, we explore potential extensions and customizations to HyperSym for further academic projects or research. The complete version of MATLAB implementation of HyperSym is available in a public repository, and some extensions and modifications are provided as Supporting Information.

1 | Introduction

Structural analysis is a core part of engineering, and numerous methods have been developed and improved over the last century. A survey comparing ten structural analysis topics revealed that the finite element (FE) method was perceived as the most challenging by students [1]. Although the FE method was originally used for linear elastic problems in structural mechanics [2], more intricate structural behaviors demand a grasp of nonlinear analysis, posing another obstacle for students.

However, there are educational FE programs available to aid with this learning process. In terms of MATLAB programming language, some notable examples include a graphical user interface integrated with a solver for the analysis of bar and beam elements [3], a toolbox for static and dynamic structural analysis [4], and a platform for slope stability used in geotechnical engineering [5]. These codes have been effectively used in engineering classes, with students reporting a positive impact on their learning experience.

Some authors employed symbolic manipulation software, such as Mathematica [6]. Examples include codes for linear elasticity [7] and plasticity [8]. More comprehensive multiphysics software like FreeFEM++ [9] and the FEniCS Project [10, 11] adopt an object-oriented symbolic derivation approach [12] to bridge the gap between the formulation of partial differential equations and their computational implementation. Despite the unique features, these solutions effectively conceal many implementation details from the user through symbolic or automatic differentiation techniques. For this reason, students may benefit more from a more straightforward framework.

In this work, we developed an educational code for FE analysis of hyperelastic materials: `HyperSym`, a software that uses MATLAB's Symbolic Math Toolbox [13] to generate ready-to-use functions that are easily integrated to FE frameworks. Hyperelastic models comprehend a wide array of materials, from rubber to soft biological tissue, and their study serves as a great introduction to material nonlinear analysis.

We note that some textbooks place the hyperelasticity section as an introduction to finite deformations required for large-strain elasto-plasticity [14, 15], but after the section on its small-strain counterpart. Introducing material nonlinearity with plasticity can be overwhelming for the student, since its implementation requires keeping track of internal variables and implementing return mapping algorithms on top of a nonlinear solution algorithm. On the other hand, hyperelastic tensors are not path dependent [16] and can be computed explicitly from a suitable deformation measure. For this reason, we built the current version of `HyperSym` and its integration into the educational FE software `NLFEA` [17] for the Nonlinear Finite Element classes at the Department of Mechanical Engineering from PUC-Rio. This tool is geared toward students at the master's and doctoral levels, but it may also be applied to teach undergraduates who already have some basic knowledge of the FE analysis.

The objective of this manuscript is to present the features of `HyperSym` integrated to `NLFEA` and how they can be implemented in the engineering curriculum. The outline of this paper is as follows: Section 2 explains the main features of `HyperSym` and its integration to `NLFEA`. In Section 3, we discuss the application of `HyperSym` in the Mechanical Engineering curriculum, including extensions and complements to the original program. Lastly, conclusions are presented in Section 4.

2 | The `HyperSym` Program

`HyperSym` is a code that generates a MATLAB function computing the stress and tangent modulus tensors used in FE analysis of hyperelastic materials. In this section, we describe the implementation details of `HyperSym` in four parts:

1. Definition of a hyperelastic material model.
2. Writing input data in the main script (`HyperSymScript`).
3. Symbolic derivation of material tensors.
4. Integration of generated functions into an FE framework.

For the FE framework, we provide a modified version of `NLFEA`. The versions of the code used in this work are available in our public repository.

2.1 | Hyperelastic Models Definition

The total Lagrangian FE formulation for quasi-static analysis of hyperelastic bodies yields a nonlinear system that is often solved with Newton–Raphson schemes. The derivation for the local tangent stiffness matrix and residual vector is found in many references, for example [18]. In this context, to update the tangent stiffness matrix, we must first compute the second Piola–Kirchhoff stress ($\boldsymbol{S}$) and the material tangent modulus ($\boldsymbol{D}$) tensors at each integration point.

The tensors $\boldsymbol{S}$ and $\boldsymbol{D}$ are, respectively, the first and second derivatives of the strain energy density W with respect to the Green–Lagrange strain tensor $\boldsymbol{E}$. These tensors can also be expressed as functions of the right Cauchy–Green deformation tensor $\boldsymbol{C} = 2\boldsymbol{E} - \boldsymbol{I}$ by means of the chain rule, where $\boldsymbol{I}$ is the second-order identity tensor. This yields the expressions

$$\boldsymbol{S} = \frac{\partial W}{\partial \boldsymbol{E}} = 2\frac{\partial W}{\partial \boldsymbol{C}}, \quad \text{and} \tag{1}$$

$$\boldsymbol{D} = \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{E}} = 2\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{C}} = 4\frac{\partial^2 W}{\partial \boldsymbol{C}\partial \boldsymbol{C}}. \tag{2}$$

In the context of isotropic materials, it is common to define W as a function of tensor $\boldsymbol{C}$ invariants, that can be computed from the principal stretches λ_i ($i = 1, 2, 3$) by

$$I_1 = \operatorname{tr}(\boldsymbol{C}) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \tag{3}$$

$$I_2 = \frac{1}{2}[(\operatorname{tr}(\boldsymbol{C}))^2 - \operatorname{tr}(\boldsymbol{C}^2)] = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2, \quad \text{and} \tag{4}$$

$$I_3 = \det(\boldsymbol{C}) = J^2 = \lambda_1^2\lambda_2^2\lambda_3^2, \tag{5}$$

where λ_i are the square roots of the eigenvalues of $\boldsymbol{C}$, and the Jacobian

$$J = \det(\boldsymbol{F}) \tag{6}$$

is a measure of volumetric dilation, in which $\boldsymbol{F}$ is the deformation gradient tensor.

Conversely, one may use reduced invariants that are unaffected by volumetric dilation and are given by

$$J_1 = I_1 I_3^{-1/3} = I_1 J^{-2/3}, \quad \text{and} \tag{7}$$

$$J_2 = I_2 I_3^{-2/3} = I_2 J^{-4/3}. \tag{8}$$

In general, hyperelastic models may not present the high degree of incompressibility often observed in solids, a shortcoming that is particularly noticeable in the St. Venant–Kirchhoff (SVK) model. By introducing some terms that rely on the Jacobian, we can induce significant resistance to volume change. These are the cases of the modified SVK's (mSVK) versions and the neo-Hookean (nH) models [19, 20] presented in Table 1. Since these models are functions of the first (λ) and second (μ) Lamé's parameters, their resistance to compression cannot be adjusted independently, so we call them "compressible models" in this text.

Another method to enforce nearly incompressible behavior consists in splitting W into a distortional (or isochoric) term W^{iso} and a volume changing W^{vol} in the form

$$W(J_1, J_2, J) = W^{\text{iso}}(J_1, J_2) + W^{\text{vol}}(J). \tag{9}$$

The nearly incompressible Mooney–Rivlin (MR) and Yeoh models presented in Table 2 use two variations of the penalty formulation, where either the bulk modulus $\varkappa$ or N generic incompressibility constants D_j can be adjusted to fine-tune a material's compressive behavior to match experimental data or facilitate convergence. Likewise, the constants A_{mn} ($m, n = 1, 2, 3$) impact solely the material's distortional behavior.

There are other methods for enforcing incompressibility introduced in the literature which are not discussed here for the sake of brevity. For further discussion on the theory of hyperelasticity, the reader is referred to Supporting Information S1: Section 1 and references therein.

2.2 | Input Data in `HyperSymScript`

The main script for `HyperSym`, `HyperSymScript`—see Appendix A—is where the user defines the material model and calls the `HyperSym` function to perform symbolic differentiation. `HyperSymScript` defines the strain energy density function (W) of the desired model and stores its properties in variables `W` and `PROP`, respectively, in terms of an appropriate suitable set of symbolic variables (tensor invariants and material properties). The following lines demonstrate the MR model from Table 2, where `Symbol` is a character array containing the name abbreviation of the functions to be generated:

```
 8 %% Symbolic variables
 9 syms J1 J2 J A10 A01 K real
10 Symbol = 'MR'; % MR: Mooney-Rivlin
11 PROP = [A10 A01 K];
12 W = A10*(J1 - 3) + A01*(J2 - 3) + K/2*(J - 1)^2;
```

Next, `HyperSymScript` calls `HyperSym` (explained in detail later and available in Appendix B) to derive the tensors `S` and `D`:

```
13 %% Call HyperSym
14 [Sv,Dv,C] = HyperSym(W);
```

On Line 14, the right Cauchy–Green tensor `C` is output in matrix format, but the PK2 stress (`Sv`) and material tangent modulus (`Dv`) follow Voigt notation. They are stored as a column vector (indicated by curly brackets) and a matrix (indicated by square brackets), respectively, following the conventions

$$\{\boldsymbol{S}\} = \{S_{11} \quad S_{22} \quad S_{33} \quad S_{12} \quad S_{23} \quad S_{13}\}^T, \quad \text{and} \tag{10}$$

$$[\boldsymbol{D}] = \begin{bmatrix} D_{1111} & D_{1122} & D_{1133} & D_{1112} & D_{1123} & D_{1113} \\ D_{1122} & D_{2222} & D_{2233} & D_{2212} & D_{2223} & D_{2213} \\ D_{1133} & D_{2233} & D_{3333} & D_{3312} & D_{3323} & D_{3313} \\ D_{1112} & D_{2212} & D_{3312} & D_{1212} & D_{1223} & D_{1213} \\ D_{1123} & D_{2223} & D_{3323} & D_{1223} & D_{2323} & D_{2313} \\ D_{1113} & D_{2213} & D_{3313} & D_{1213} & D_{2313} & D_{1313} \end{bmatrix}. \tag{11}$$

Finally, `HyperSymScript` generates a MATLAB function and saves it in the folder "Functions":

```
15 %% Generate and store hyperelastic function
16 Dir = [cd '/Functions/'];  % Directory to save the function
17 if ~exist(Dir,'dir'), mkdir(Dir); end
18 Controls = struct('File',[Dir Symbol],'Optimize',false);
19 matlabFunction(Sv,Dv,'Vars',{C,PROP},Controls);
20 fprintf('Done!\n\nEllapsed time: %1.1f seconds. \n\n',toc)
```

TABLE 1 | Compressible hyperelastic models.

Symbol	Strain energy density function, W
SVK	$\frac{\lambda}{8}(I_1 - 3)^2 + \frac{\mu}{4}(I_1^2 - 2I_2 - 2I_1 + 3)$
mSVK1	$\frac{\lambda}{2}(\ln J)^2 + \frac{\mu}{4}(I_1^2 - 2I_2 - 2I_1 + 3)$
mSVK2	$\lambda(J - \ln J - 1) + \frac{\mu}{4}(I_1^2 - 2I_2 - 2I_1 + 3)$
mSVK3	$\frac{\lambda}{2}(J - 1)^2 + \frac{\mu}{4}(I_1^2 - 2I_2 - 2I_1 + 3)$
nH1	$\frac{\lambda}{2}(\ln J)^2 - \mu \ln J + \frac{\mu}{2}(I_1 - 3)$
nH2	$\lambda(J - \ln J - 1) - \mu \ln J + \frac{\mu}{2}(I_1 - 3)$
nH3	$\frac{\lambda}{2}(J - 1)^2 - \mu \ln J + \frac{\mu}{2}(I_1 - 3)$

2.3 | Tensor Derivation Using `HyperSym`

`HyperSym` derives the tensors $\boldsymbol{S}$ and $\boldsymbol{D}$ for hyperelastic models written in terms of either direct or reduced invariants. The idea behind `HyperSym` is essentially to compute the first and second derivatives of strain energy W with respect to the components of tensor $\boldsymbol{C}$.

First, `HyperSym` initializes a number of symbolic variables on Lines 8–10. Then, from Equations (3)–(5), (7), and (8), it computes on Lines 11–15 symbolic expressions for the invariants of `C` and substitutes them in the variable `W` provided as an input (Lines 16–17):

TABLE 2 | Nearly incompressible hyperelastic models.

Symbol	W^{iso}	W^{vol}	Parameters
MR	$A_{10}(J_1 - 3) + A_{01}(J_2 - 3)$	$\frac{\varkappa}{2}(J - 1)^2$	$A_{10}, A_{01}, \varkappa$
Yeoh	$\sum_{i=1}^{3} A_{i0}(J_1 - 3)^i$	$\sum_{j=1}^{N} \frac{1}{D_j}(J - 1)^{2j}$	A_{i0}, D_j

```
8  %% Symbolic Variables
9  syms I1 I2 I3 J J1 J2 real
10 C = sym('C', [3,3],'real'); % Right Cauchy-Green def. tensor, C = F'*F
11 %% Principal Invariants of C
12 mI1 = trace(C);                  mI2 = (trace(C)^2 - trace(C^2))/2;
13 mJ = sqrt(det(C));               mI3 = mJ^2;
14 %% Reduced Invariants of C
15 mJ1 = mI3^(-1/3)*mI1;            mJ2 = mI3^(-2/3)*mI2;
16 %% Strain Energy Function, W(C)
17 W = subs(W,{I1,I2,I3,J,J1,J2},{mI1,mI2,mI3,mJ,mJ1,mJ2});
```

Next, according to Equation (1), the second Piola–Kirchhoff stress tensor is computed with the MATLAB Symbolic Math Toolbox's function `diff`, as shown on Lines 18–25. On Line 25, the symmetry of PK2 stress tensor is imposed, as done by [21]:

```
18 %% Derivation of PK2 Stress Tensor, S
19 S = sym(zeros(3,3));
20 for i = 1:3
21     for j = 1:3
22         S(i,j) = 2*diff(W(1),C(i,j));
23     end
24 end
25 S = (S + transpose(S))/2; % Enforce symmetry of S
```

At last, S is derived with respect to the components of C following Equation (2) to find the fourth-order tensor $\boldsymbol{D}$. This is done in Lines 26–36 of HyperSym with four nested loops:

```
26 %% Derivation of Elasticity Tensor, D
27 D = sym(zeros(3,3,3,3));
28 for i = 1:3
29     for j = 1:3
30         for k = 1:3
31             for l = 1:3
32                 D(i,j,k,l) = 2*diff(S(i,j),C(k,l));
33             end
34         end
35     end
36 end
```

Although the derivations could be performed more efficiently by exploiting the symmetries of the tensors, this method presents derivatives more clearly by using the full matrix representation before the Voigt notation. Finally, the last lines of HyperSym are dedicated to reduce the number of components of S and D based on the symmetries involved and recast them in Voigt notation, respectively, Sv and Dv:

```
37 %% Impose symmetry and write in Voigt notation
38 S = simplify(subs(S,[C(2,1),C(3,2),C(3,1)],[C(1,2),C(2,3),C(1,3)]));
39 D = simplify(subs(D,[C(2,1),C(3,2),C(3,1)],[C(1,2),C(2,3),C(1,3)]));
40 Sv = [S(1,1) S(2,2) S(3,3) S(1,2) S(2,3) S(1,3)].';
41 Dv = [...
42     D(1,1,1,1) D(1,1,2,2) D(1,1,3,3) D(1,1,1,2) D(1,1,2,3) D(1,1,3,1);
43     D(2,2,1,1) D(2,2,2,2) D(2,2,3,3) D(2,2,1,2) D(2,2,2,3) D(2,2,3,1);
44     D(3,3,1,1) D(3,3,2,2) D(3,3,3,3) D(3,3,1,2) D(3,3,2,3) D(3,3,3,1);
45     D(1,2,1,1) D(1,2,2,2) D(1,2,3,3) D(1,2,1,2) D(1,2,2,3) D(1,2,3,1);
46     D(2,3,1,1) D(2,3,2,2) D(2,3,3,3) D(2,3,1,2) D(2,3,2,3) D(2,3,3,1);
47     D(3,1,1,1) D(3,1,2,2) D(3,1,3,3) D(3,1,1,2) D(3,1,2,3) D(3,1,3,1)];
```

2.4 | Incorporating a Hyperelastic Model Into NLFEA

The functions generated by the HyperSymScript (Section 2.2) can now be incorporated into the NLFEA framework, that originally features solely the MR model. In what follows, we specify how to change the MATLAB function Mooney from the original NLFEA to the one obtained via HyperSym, that is, MR.

To do so, one should update the NLFEA's function that computes and assembles the internal force vector and tangent stiffness matrix, HYPER3D. This is done by replacing Line 42 of HYPER3D, that originally reads

```
[STRESS DTAN] = Mooney(F, PROP(1), PROP(2),
PROP(3), LTAN);
```

to

```
[STRESS, DTAN] = MR(F'*F, PROP);
```

Here, F is the deformation gradient tensor, so F'*F stands for the right Cauchy–Green deformation tensor. Furthermore, PROP is an array of properties—which has the format PROP = $[A_{10}\ A_{01}\ \kappa]$ for the MR model, and LTAN is a Boolean indicating whether the elasticity tensor should be computed. The outputs STRESS and DTAN are, respectively, the PK2 stress vector and the tangent modulus matrix, both in Voigt notation.

The final version of the HYPER3D used for solving the numerical examples in Section 3 uses the function feval that enables the user to use any model previously derived with HyperSym identified by the string variable MID. This is shown on the following line, where C is a 3×3 matrix:

```
STRESS(:) = feval(MID, C, PROP).
```

3 | Applications of HyperSym in the Curriculum

The intended use of HyperSym and NLFEA in the classroom is divided in three parts: introducing with practical examples, assigning homework to modify the code to analyze different structures, and assigning projects for developing useful extensions to the framework. To illustrate these applications, we present some examples and extensions in this section, followed by a brief discussion of HyperSym's applications in research.

For the numerical examples presented here, we generated MATLAB functions from HyperSym for the hyperelastic models in Tables 1 and 2. The three problems selected for this section can be used to introduce students to progressively more complex features of nonlinear elastic structural analysis and include the following:

1. **Cube under uniaxial traction**: compare the behavior of different models and how to validate the numerical implementation with previously known analytical solutions.
2. **Thin square strip**: this example was modified from the literature to illustrate the importance of resistance to compression in large deformations.
3. **Infinite thick-walled cylinder**: shows a more practical application of a pressure vessel and compares the results with the literature and a commercial software.

TABLE 3 | Convergence parameters.

Description	Value
Maximum number of NR iterations	30
Maximum number of consecutive bisections	6
Bisection tolerance	10^5
Convergence tolerance	10^{-6}

In these examples, we modeled the domain with 8-node linear hexahedrals (brick) elements and set the default convergence parameters of the Newton–Raphson procedure and bisection method to those in Table 3. The convergence criterion adopted was the infinity norm of the residual vector at the free degrees of freedom. See the original publication [18] for further information on the convergence features of NLFEA.

3.1 | Example 1: Cube Under Uniaxial Traction

The first example is a cube elongated in Direction 1 discretized by a single brick element, as shown in Figure 1. Since all (displacement) degrees of freedom are prescribed in this example, no convergence is needed, so we will only compare the stress of different compressible hyperelastic models.

The homogeneous uniaxial deformation of this cube is described by

$$x_1 = \gamma X_1,\ x_2 = X_2, \quad \text{and} \quad x_3 = X_3, \tag{12}$$

where X_i and x_i are the material and spatial coordinates, respectively, and γ is the stretch.

In this particular case, the final configuration is known; for the prescribed displacement of $u_1 = 1$ mm, the maximum value for the stretch in that direction is $\gamma = 2$. Then, from Section 2.1 and Equation (12), we have that

$$\begin{aligned} \boldsymbol{F} &= \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \begin{bmatrix} \gamma & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ \boldsymbol{C} &= \boldsymbol{F}^T\boldsymbol{F} = \begin{bmatrix} \gamma^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{and} \\ J &= \det(\boldsymbol{F}) = \gamma. \end{aligned}$$

We study both the analytical and numerical responses of the cube for the material models in Table 1. We adopted $E = 1000$ MPa, and $\nu = 0.3$ [20], from which the Lamé's parameters are calculated by

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \text{and}$$
$$\mu = \frac{E}{2(1+\nu)}.$$

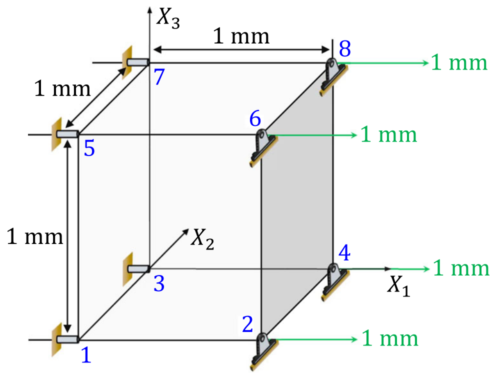


FIGURE 1 | Cube under uniaxial traction.

TABLE 4 | σ_{11} component of SVK- and nH-based models under uniaxial traction.

Model	σ_{11}
SVK	$\frac{\gamma\lambda}{2}(\gamma^2-1)+\gamma\mu(\gamma^2-1)$
mSVK1	$\frac{\lambda}{\gamma}\ln(\gamma)+\gamma\mu(\gamma^2-1)$
mSVK2	$\frac{\lambda}{\gamma}(\gamma-1)+\gamma\mu(\gamma^2-1)$
mSVK3	$\lambda(\gamma-1)+\gamma\mu(\gamma^2-1)$
nH1	$\frac{\lambda}{\gamma}\ln(\gamma)+\frac{\mu}{\gamma}(\gamma^2-1)$
nH2	$\frac{\lambda}{\gamma}(\gamma-1)+\frac{\mu}{\gamma}(\gamma^2-1)$
nH3	$\lambda(\gamma-1)+\frac{\mu}{\gamma}(\gamma^2-1)$

The analytical solution for the component σ_{11} displayed in Table 4 is obtained from the *push-forward* of the PK2 stress, that yields the Cauchy stress

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{FSF}^T, \tag{13}$$

where the expression for $\boldsymbol{S}$ can be derived from Equation (1) or found in [20].

Figure 2 shows the component σ_{11} of the Cauchy stress as the cube is stretched. The highlighted regions represent the area where the material models have similar behavior. The analytical solution is also plotted in the range $0.2 \leqslant \gamma \leqslant 1$, which corresponds to the compression of the cube.

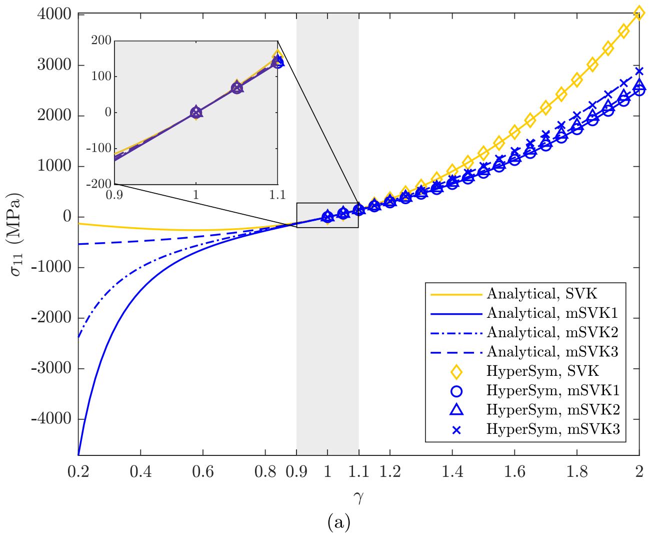
One might observe the traditional SVK model attains a minimum at $\gamma = \sqrt{3}/3$, since it satisfies the necessary and sufficient conditions that $\partial\sigma_{11}/\partial\gamma = 0$ and $\partial^2\sigma_{11}/\partial\gamma^2 > 0$. This variation in the stress direction during compression is plainly incompatible with the predicted physical behavior of most solids, where the compressive stress is expected to decrease as the material shrinks. The other material models of this example were designed to overcome this limitation, which can be seen in Figure 2a,b as $\sigma_{11} \to -\infty$ as $\gamma \to 0$, with the exception of mSVK3.

3.2 | Example 2: Thin Square Strip

Here, we investigate the behavior of a thin square strip based on [22]. The strip is clamped on both sides and stretched in Direction 1, according to Figure 3. A thickness of 0.05 in (1.27 mm) is considered, as done by [23]. Due to symmetries, only a quarter of the strip is modeled and analyzed.

The prescribed displacements follow the form of $u_1 = (\gamma - 1)L/2$, in which L is the length of the strip, and γ is the stretch in X_1 direction. Thus, for $L = 8$ in (203.2 mm) and $\gamma = 3$, $u_1 = 406.4$ mm. Initially, an MR model is adopted with parameters $A_{10} = 24$ psi (0.1655 MPa) and $A_{01} = 1.5$ psi (0.0103 MPa).

However, unlike the reference [22] who used an incompressibility formulation, we estimated the bulk modulus κ from the Poisson's ratio ν. Then,



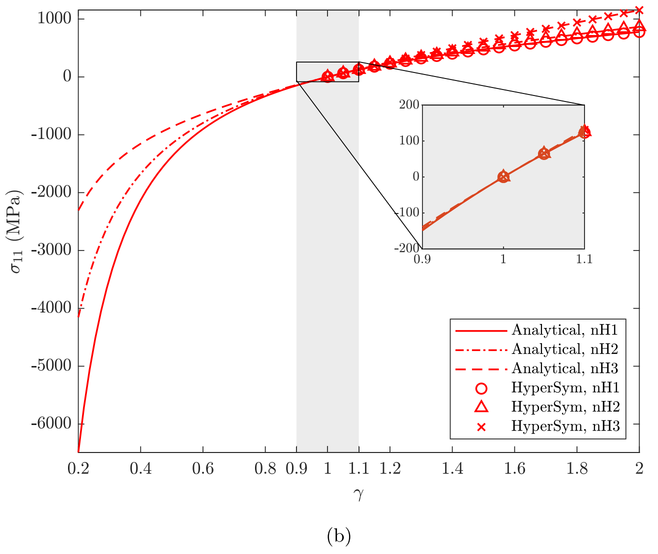


FIGURE 2 | Component σ_{11} of Cauchy stress × stretch γ for (a) SVK-based models and (b) nH-based models under uniaxial traction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\kappa = \frac{2\mu(1+\nu)}{3(1-2\nu)},$$

where $\mu = 2(A_{10} + A_{01})$ is an approximation obtained from the consistency condition presented in [18].

In this work, ν was arbitrarily set to 0.4, a relatively small value, to study the change of volume in different models, which include the compressible models from Table 1 in our study. Thus, we expect a rather compressible response of the strip so we can study its response under large deformations. From these parameters, we finally compute

$$\lambda = \kappa - \frac{2}{3}\mu.$$

Initially, we look at Figure 4, which shows the top view of the deformed structure for the models in Table 1 and MR. We observe that the mSVK seemingly deformed the least and the

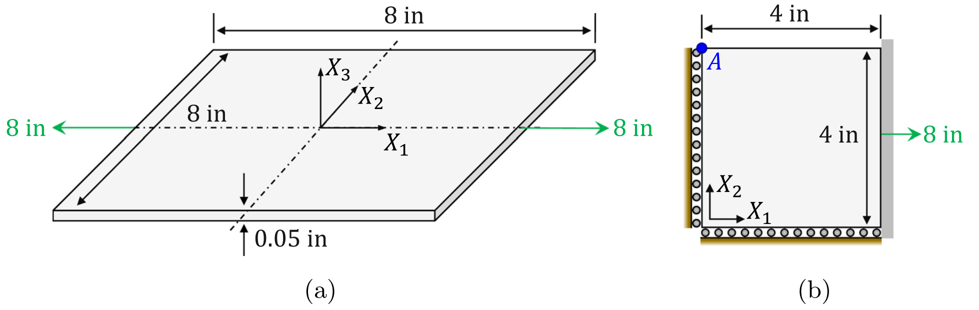


FIGURE 3 | Strip. (a) 3D model; (b) top view, illustrating the symmetry and boundary conditions used in the FE model. Point A, located at $X_3 = 0$, indicates the node taken to study the equilibrium path.

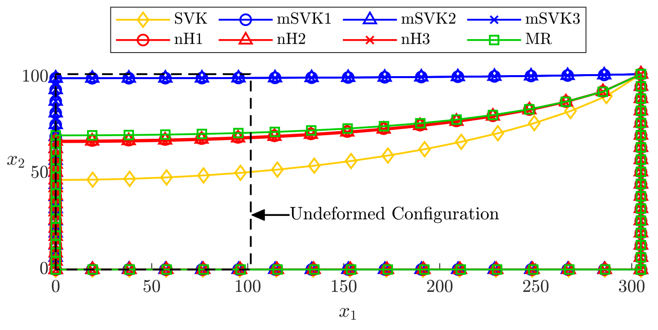


FIGURE 4 | Top view of the deformed structure for compressible models MR. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

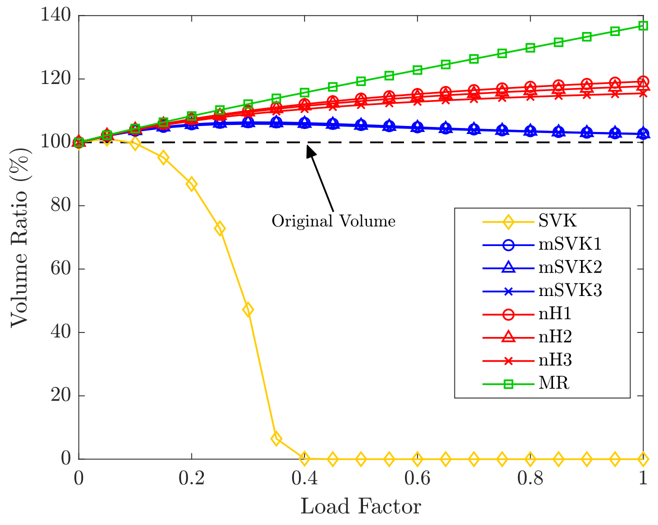


FIGURE 5 | Ratio between original and current volume for compressible models and MR. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

SVK deformed the most due to its lack of significant resistance to compression.

We also study the change in volume provoked due to Poisson's effect, for as the strip is stretched in one direction, the other dimensions tend to reduce. Figure 5 shows that the investigated models do not preserve the original volume as the structure is deformed. The MR and nH-based models increased the volume, whereas the mSVK models attained a maximum volume ratio for load factors ranging from 0.2 to 0.4. The difficulty in keeping the original volume is due to ν not being sufficiently close to 0.5, the theoretical value for a completely incompressible material. Lastly, we observe that the traditional SVK material does not present realistic behavior as the volume of the strip tends to zero. This further illustrates that this type of material should be limited to small deformation applications.

3.3 | Example 3: Infinite Thick-Walled Cylinder

The last example deals with an infinite thick-walled cylinder subject to internal pressure $p_{in} = 150$ psi (1.034 MPa), vid. Figure 6. The cylinder has inner radius $r_{in} = 7$ in (177.80 mm), outer radius $r_{out} = 18.625$ in (473.08 mm), and thickness $t = 0.775$ in (19.685 mm). Only a quarter of the cylinder section is meshed with $20 \times 20 \times 1$ elements, but it could also be modeled under plane strain or axisymmetric assumptions [18, 24].

In this example, we decided to use a convergence tolerance of 0.01 to speed up the analysis, without significantly altering the final solution while achieving convergence. We adopted the MR model with the following parameters to enable comparison with [18]: $A_{10} = 80$ psi (0.5516 MPa), $A_{01} = 20$ psi (0.1379 MPa), and $\kappa = 10^4$ psi (68.95 MPa).

In this case, the external force vector is displacement-dependent due to the pressure boundary condition. However, for this example's parameters we can neglect its influence in the tangent stiffness matrix and still achieving reasonable convergence with 10 equal load increments, as shown in Table 5. The adopted reference [18] computed the stress at the center of the first element in a mesh with five elements in the radial direction, whereas we interpolated the radial component of the stress at the nodes.

The subsequent results are presented for nodes along a radial path and at $X_3 = t$ in the undeformed configuration. Figure 7 shows the radial, hoop, and axial stresses, σ_r, σ_θ, and σ_z, respectively, along a radial path in the final configuration. A close agreement is observed between our FE implementation—using `NLFEA` with material model functions generated by `HyperSym`—and those obtained with ANSYS [25], except at the boundaries of the cylinder. At these locations, the boundary affects the stresses, and it is more severe in the radial component σ_r—a phenomenon also observed in [24, 26].

3.4 | Projects: Extensions to HyperSym

The original version of HyperSym discussed above is an open-source software focused on simplicity and ease of use for educational purposes, especially for those being introduced to the structural analysis of nonlinear elasticity. Written in MATLAB, HyperSym is customizable and extendable into an array of applications to serve students who want to deepen their knowledge in the subject or researchers who want an accessible tool to work with or improve upon. Some extension ideas are presented include, but are not limited to the following.

1. **Two-Dimensional Analysis:** A simple extension to HyperSym would be integrating it to a two-dimensional structural analysis educational software. This extension can be used as a project in which the student can learn how to properly adapt the hyperelastic tensor components into plane stress and plane strain analysis. In this project, the student can either modify HyperSym to derive efficient hyperelastic functions for two-dimensional analysis, or just use the necessary components of the hyperelastic tensors. Care should be taken in plane stress applications, as it may require a nonlinear solver when updating the element stiffness matrix to enforce that there are no out-of-plane stress components.
2. **Selective Reduced Integration:** The isochoric-volumetric split of Equation (9) can be exploited to derive separate stress and tangent modulus tensors. By integrating the volumetric part of these tensors with fewer Gauss integration points, the phenomenon known as volumetric locking can be reduced or avoided. HyperSym users can modify the original code to compute separate sets of tensors that facilitate the implementation of selective reduced integration (SRI).
3. **Biomechanics:** The hyperelastic models in Tables 1 and 2 are isotropic, as they are functions of principal or reduced invariants of a deformation tensor. Living tissue, among other materials, often presents a highly anisotropic material behavior. Researchers and students may modify HyperSym to derive the corresponding tensors for these models, such as those in [27, 28]. The Fung material model and its many variations are often used as the starting point for biomechanical analysis [29], which may also include temperature and time dependency (viscoelasticity). This extension can also serve as an introduction to concepts and features of specialized biomechanics packages like the FEBiO suite [30].
4. **Integration to Commercial Software:** This permits the HyperSym user to integrate the hyperelastic functions into commercial FE packages by automatically writing the appropriate lines into Fortran user material subroutines like UMAT in Abaqus or USERMAT in ANSYS. Care

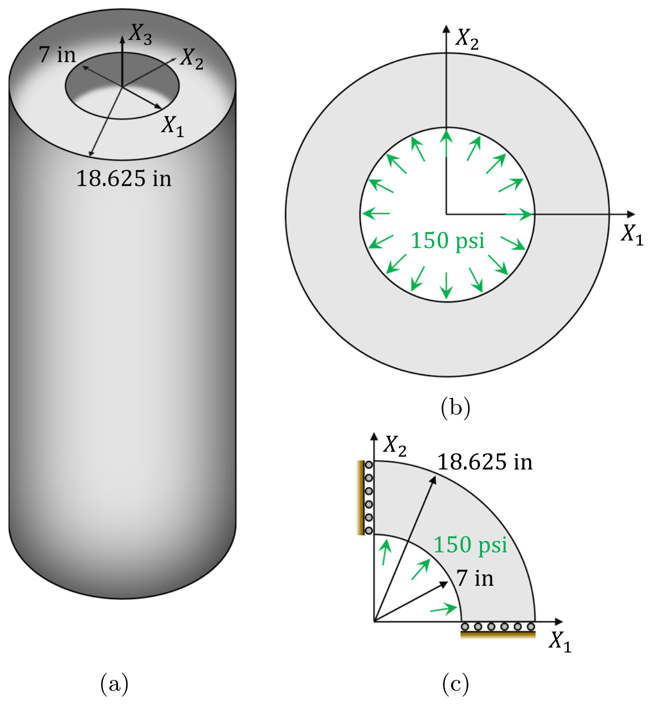


FIGURE 6 | Thick-walled cylinder: (a) 3D model; (b) internal pressure at the cross section; (c) FE model: one-quarter of the initial geometry, and boundary conditions.

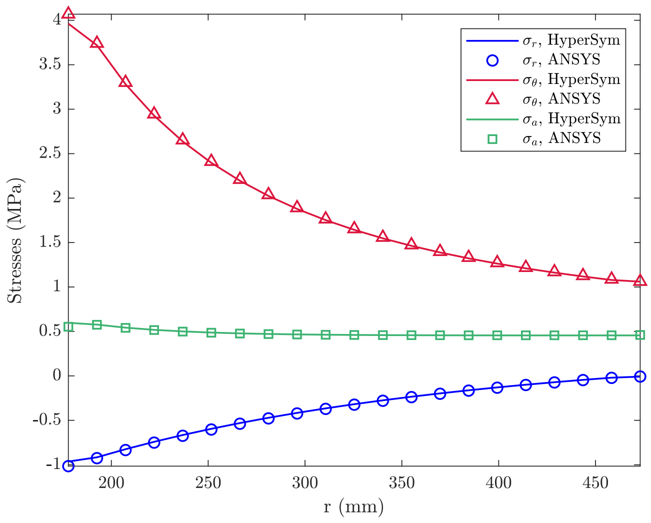


FIGURE 7 | Equilibrium path (as stress vs. radius) for a node at the inner wall of cylinder and MR material. Comparison between our framework (HyperSym) and a commercial software (ANSYS).

TABLE 5 | Target and this work's results for the infinite thick-walled cylinder problem considering MR material model.

Criterion	Units	Target value [18]	This work	Relative error (%)
Displacement at the inner radius	mm	182.4	188.1	3.12
	in	7.180	7.404	
Radial stress at 10% of the thickness	MPa	−0.8411	−0.8249	1.93
	psi	−122.0	−119.6	

should be taken to perform the push-forward operations on the hyperelastic tensors, since these commercial software often use a spatial formulation in their implementation.

The suggestions above are ideas of how students and researchers can leverage `HyperSym` to shorten learning and development cycles by exploiting features like symbolic math and automatic code generation found in software like MATLAB. For convenience, `HyperSym` is already integrated to a modified version of `NLFEA`, since it requires no additional setup or licenses on top of MATLAB Symbolic Math Toolbox. However, integrating `HyperSym`-generated functions into commercial software, like Abaqus or ANSYS, is a good way of leveraging from the convenience of the former and the features and high performance of the latter.

Finally, we introduce `HyperFit`, a complement to our framework, responsible for adjusting material parameters for a hyperelastic model from the strain energy density function (similar to `HyperSym`) and experimental test data, as illustrated in Figure 8. Although equivalent tools are readily available in most commercial FE software, students may access and edit `HyperFit` to grasp the basic ideas behind fitting data to experimental results—also an important aspect of structural analysis.

The code in our repository also supports deriving isochoric and volumetric functions for SRI—see Equation (9)—and automatically generates efficient MEX versions of MATLAB functions. The reader is referred to Supporting Information S1: Sections 2 and 3 for additional reading on the modifications to `HyperSymScript`.

3.5 | Use of `HyperSym` in Master's and Doctoral Theses

`HyperSym` and its extensions are resources intended to facilitate the implementation of hyperelastic models into structural analysis, avoiding long derivations that are prone to errors due to its complexity. Furthermore, the component-wise symbolic derivation and the automatic MEX file generation yield efficient MATLAB functions that can be used in research.

In our research group, `HyperSym` can be used in topology optimization applications [32]. Recently, this tool was employed in a doctoral dissertation to compute stress and elasticity tensors for a study involving geometric and material nonlinearities, which demonstrated consistency in both two-dimensional FE problems [33].

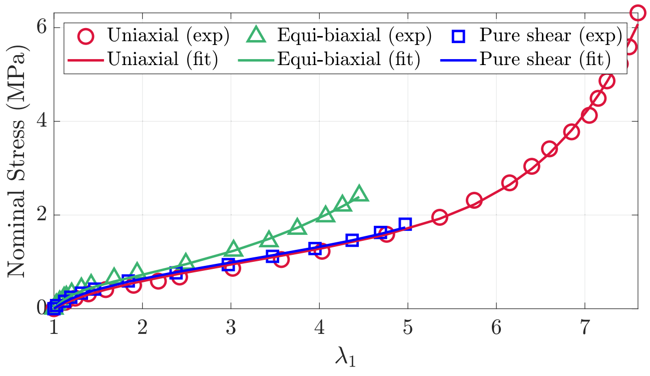


FIGURE 8 | Ogden model fitted to Treloar data [31] for nominal stress versus nominal stretch (λ_1) of three tests: simple tension, equi-biaxial, and pure shear.

The main application of `HyperSym` is to facilitate quick prototyping that expedites preliminary tests in research, which is further enhanced with the features provided by research-oriented languages such as MATLAB. The final version of our software is often written in more efficient programming languages like `Fortran` and `C`. The symbolic expressions derived with `HyperSym` can then be translated to these languages easily with functions `Fortran` and `ccode` from the MATLAB's Symbolic Math Toolbox.

4 | Conclusions

In this work, we discussed the main concepts and implementation of `HyperSym`, a symbolic tool for the derivation of tensors required for the integration of hyperelastic materials into FE analysis. Unlike full-fledged FE software with symbolic manipulation, `HyperSym` only derives the tensors used to compute local quantities, which may readily be integrated to existing FE packages for educational purposes. We illustrated this integration with a modified version of the `NLFEA` framework and offered a few numerical examples to highlight the relevance of underlying concepts related to hyperelasticity.

Extensions to `HyperSym` were proposed as additional projects to be assigned to students, where they can deepen their understanding of nonlinear analysis or integration with commercial software. We briefly describe some extensions provided in the current version of `HyperSym`, and we also present `HyperFit`, a companion educational code that computes hyperelastic material parameters from experimental test data.

Author Contributions

Vinicius Oliveira Fontes: formal analysis, methodology, software, validation, visualization, writing – original draft. **André Xavier Leitão:** resources, software, validation, visualization, writing – review and editing. **Anderson Pereira:** conceptualization, supervision, writing – review and editing.

Acknowledgments

The authors kindly thank Professor Nam-Ho Kim for authorizing the use and modification of his finite element source code MATLAB code, NLFEA, and Professor Raymond Ogden for sharing the experimental data used for the curve fitting algorithm. The Article Processing Charge for the publication of this research was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) (ROR identifier: 00x0ma614).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

All data and source code written in MATLAB needed to evaluate the conclusions in this paper are available at https://github.com/

viniciusofontes/HyperSym. The finite element source code is based on NLFEA and is available at https://web.mae.ufl.edu/nkim/INFEM/.

References

1. D. Johnson and I. M. May, "The Teaching of Structural Analysis," *Structural Engineer* 86, no. 22 (2008): 32–39.

2. K. J. Bathe, *Finite Element Procedures*, 1st ed. (Prentice-Hall Inc, 1996).

3. P. C. Lopes, R. L. Rangel, and L. F. Martha, "An Interactive User Interface for a Structural Analysis Software Using Computer Graphics Techniques in MATLAB," *Computer Applications in Engineering Education* 29, no. 6 (2021): 1505–1525, https://doi.org/10.1002/cae.22406.

4. S. François, M. Schevenels, D. Dooms, et al., "Stabil: An Educational Matlab Toolbox for Static and Dynamic Structural Analysis," *Computer Applications in Engineering Education* 29, no. 5 (2021): 1372–1389, https://doi.org/10.1002/cae.22391.

5. Z. Y. Yin, J. C. Teng, H. L. Wang, and Y. F. Jin, "A MATLAB-Based Educational Platform for Analysis of Slope Stability," *Computer Applications in Engineering Education* 30, no. 2 (2022): 575–588, https://doi.org/10.1002/cae.22474.

6. Wolfram Research, Inc., "Mathematica," accessed July 2023, https://www.wolfram.com/mathematica.

7. D. L. Cecílio and d. T. D. Santos, "A Finite Element Elasticity Programming in Mathematica Software," *Computer Applications in Engineering Education* 26, no. 6 (2018): 1968–1985, https://doi.org/10.1002/cae.21958.

8. Y. Jiang and C. Wang, "On Teaching Finite Element Method in Plasticity With Mathematica," *Computer Applications in Engineering Education* 16, no. 3 (2008): 233–242, https://doi.org/10.1002/cae.20135.

9. F. Hecht, "New Development in Freefem++," *Journal of Numerical Mathematics* 20, no. 3–4 (2012): 251–265, https://doi.org/10.1515/jnum-2012-0013.

10. A. Logg, K. A. Mardal, and G. Wells, *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. 84 (Springer Science & Business Media, 2012).

11. M. S. Alnæs, J. Blechta, J. Hake, et al., "The FEniCS Project Version 1.5," *Archive of Numerical Software* 3, no. 100 (2015): 9–23, https://doi.org/10.11588/ans.2015.100.20553.

12. D. Eyheramendy and T. Zimmermann, "Object-Oriented Symbolic Derivation and Automatic Programming of Finite Elements in Mechanics," *Engineering With Computers* 15 (1999): 12–36, https://doi.org/10.1007/s003660050003.

13. The Mathworks, Inc., "MATLAB Version 9.12.0.1956245 (R2022a)," accessed July 2023, https://www.mathworks.com/products/matlab.

14. J. Bonet and R. D. Wood, *Nonlinear Continuum Mechanics for Finite Element Analysis*, 2nd ed. (Cambridge University Press, 2008).

15. d. R. Borst, M. A. Crisfield, J. J. C. Remmers, and C. V. Verhoosel, *Nonlinear Finite Element Analysis of Solids and Structures* (John Wiley & Sons, 2012).

16. R. M. Hackett, *Hyperelasticity Primer* (Springer, 2016).

17. N. H. Kim, "Introduction to Nonlinear Finite Element Analysis," accessed February 2025, https://web.mae.ufl.edu/nkim/INFEM/.

18. N. H. Kim, *Introduction to Nonlinear Finite Element Analysis* (Springer Science & Business Media, 2014).

19. A. Curnier, *Computational Methods in Solid Mechanics*. 29 (Kluwer Academic Publishers, 1994).

20. A. Klarbring and N. Strömberg, "Topology Optimization of Hyperelastic Bodies Including Non-Zero Prescribed Displacements," *Structural and Multidisciplinary Optimization* 47, no. 1 (2013): 37–48, https://doi.org/10.1007/s00158-012-0819-z.

21. C. Suchocki, "Finite Element Implementation of Slightly Compressible and Incompressible First Invariant-Based Hyperelasticity: Theory, Coding, Exemplary Problems," *Journal of Theoretical and Applied Mechanics* 55 (2017): 787–800, https://doi.org/10.15632/jtam-pl.55.3.787.

22. J. T. Oden and T. Sato, "Finite Strains and Displacements of Elastic Membranes by the Finite Element Method," *International Journal of Solids and Structures* 3, no. 4 (1967): 471–488, https://doi.org/10.1016/0020-7683(67)90002-9.

23. J. T. Oden, *Finite Elements of Nonlinear Continua* (Courier Corporation, 1972).

24. J. T. Oden and J. E. Key, "Numerical Analysis of Finite Axisymmetric Deformations of Incompressible Elastic Solids of Revolution," *International Journal of Solids and Structures* 6, no. 5 (1970): 497–518, https://doi.org/10.1016/0020-7683(70)90027-2.

25. ANSYS, Inc. Ansys® Help System, Release 2021 R1 (ANSYS, Inc., 2021). ANSYS® Mechanical APDL Product Launcher.

26. T. Scharnhorst and T. H. H. Pian, "Finite Element Analysis of Rubber-Like Materials by a Mixed Model," *International Journal for Numerical Methods in Engineering* 12, no. 4 (1978): 665–676, https://doi.org/10.1002/nme.1620120410.

27. Y. C. Fung, "Biorheology of Soft Tissues," *Biorheology* 10, no. 2 (1973): 139–155, https://doi.org/10.3233/BIR-1973-10208.

28. P. Tong and Y. C. Fung, "The Stress–Strain Relationship for the Skin," *Journal of Biomechanics* 9, no. 10 (1976): 649–657, https://doi.org/10.1016/0021-9290(76)90107-X.

29. L. A. Taber, *Nonlinear Theory of Elasticity: Applications in Biomechanics* (World Scientific, 2023).

30. S. A. Maas, B. J. Ellis, G. A. Ateshian, and J. A. Weiss, "FEBio: Finite Elements for Biomechanics," *Journal of Biomechanical Engineering* 134, no. 1 (2012): 011005, https://doi.org/10.1115/1.4005694.

31. L. R. G. Treloar, "Stress–Strain Data for Vulcanized Rubber Under Various Types of Deformation," *Rubber Chemistry and Technology* 17, no. 4 (1944): 813–825, https://doi.org/10.5254/1.3546701.

32. V. O. Fontes, A. X. Leitão, and A. Pereira, "On Interpolation Methods for Modeling Low-Density Elements in Topology Optimization of Geometrically Nonlinear Structures," in *Associação Brasileira de Métodos Computacionais em Engenharia* (2020), https://publicacoes.softaliza.com.br/cilamce2020/article/view/6516.

33. A. X. Leitão, "Stress-Constrained Topology Optimization of Hyperelastic Structures" (PhD thesis, Pontifical Catholic University of Rio de Janeiro, 2024), https://doi.org/10.17771/PUCRio.acad.69580.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.

Appendix A

HyperSymScript

```
% HyperSymScript ---------------------------------------------------------------
% Article: Fontes, V.O., Leitão, A.X., & Pereira, A. (2025).
%          HyperSym: an educational MATLAB code for hyperelasticity,
%          Computer Applications in Engineering Education.
%          DOI: 10.1002/cae.70037
% ------------------------------------------------------------------------------
clc; clear; close all; tic
%% Symbolic variables
syms J1 J2 J A10 A01 K real
Symbol = 'MR'; % MR: Mooney-Rivlin
PROP = [A10 A01 K];
W = A10*(J1 - 3) + A01*(J2 - 3) + K/2*(J - 1)^2;
%% Call HyperSym
[Sv,Dv,C] = HyperSym(W);
%% Generate and store hyperelastic function
Dir = [cd '/Functions/'];  % Directory to save the function
if ~exist(Dir,'dir'), mkdir(Dir); end
Controls = struct('File',[Dir Symbol],'Optimize',false);
matlabFunction(Sv,Dv,'Vars',{C,PROP},Controls);
fprintf('Done!\n\nElapsed time: %1.1f seconds. \n\n',toc)
```

Appendix B

HyperSym

```
% HyperSym ---------------------------------------------------------------------
% Article: Fontes, V.O., Leitão, A.X., & Pereira, A. (2025).
%          HyperSym: an educational MATLAB code for hyperelasticity,
%          Computer Applications in Engineering Education.
%          DOI: 10.1002/cae.70037
% ------------------------------------------------------------------------------
function [Sv,Dv,C] = HyperSym(W)
%% Symbolic Variables
syms I1 I2 I3 J J1 J2 real
C = sym('C', [3,3],'real'); % Right Cauchy-Green def. tensor, C = F'*F
%% Principal Invariants of C
mI1 = trace(C);               mI2 = (trace(C)^2 - trace(C^2))/2;
mJ = sqrt(det(C));            mI3 = mJ^2;
%% Reduced Invariants of C
mJ1 = mI3^(-1/3)*mI1;         mJ2 = mI3^(-2/3)*mI2;
%% Strain Energy Function, W(C)
W = subs(W,{I1,I2,I3,J,J1,J2},{mI1,mI2,mI3,mJ,mJ1,mJ2});
%% Derivation of PK2 Stress Tensor, S
S = sym(zeros(3,3));
for i = 1:3
    for j = 1:3
        S(i,j) = 2*diff(W(1),C(i,j));
    end
end
S = (S + transpose(S))/2; % Enforce symmetry of S
%% Derivation of Elasticity Tensor, D
D = sym(zeros(3,3,3,3));
for i = 1:3
    for j = 1:3
        for k = 1:3
            for l = 1:3
                D(i,j,k,l) = 2*diff(S(i,j),C(k,l));
            end
        end
    end
end
%% Impose symmetry and write in Voigt notation
S = simplify(subs(S,[C(2,1),C(3,2),C(3,1)],[C(1,2),C(2,3),C(1,3)]));
D = simplify(subs(D,[C(2,1),C(3,2),C(3,1)],[C(1,2),C(2,3),C(1,3)]));
Sv = [S(1,1) S(2,2) S(3,3) S(1,2) S(2,3) S(1,3)].';
Dv = [...
    D(1,1,1,1) D(1,1,2,2) D(1,1,3,3) D(1,1,1,2) D(1,1,2,3) D(1,1,3,1);
    D(2,2,1,1) D(2,2,2,2) D(2,2,3,3) D(2,2,1,2) D(2,2,2,3) D(2,2,3,1);
    D(3,3,1,1) D(3,3,2,2) D(3,3,3,3) D(3,3,1,2) D(3,3,2,3) D(3,3,3,1);
    D(1,2,1,1) D(1,2,2,2) D(1,2,3,3) D(1,2,1,2) D(1,2,2,3) D(1,2,3,1);
    D(2,3,1,1) D(2,3,2,2) D(2,3,3,3) D(2,3,1,2) D(2,3,2,3) D(2,3,3,1);
    D(3,1,1,1) D(3,1,2,2) D(3,1,3,3) D(3,1,1,2) D(3,1,2,3) D(3,1,3,1)];
end
```