XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

`HyperSym`: a MATLAB code for symbolic differentiation of ready-to-use tensors in hyperelastic finite element analysis

Anderson Pereira[1], André Xavier Leitão[1], Vinicius Oliveira Fontes[1]

[1]*Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro (PUC-Rio)*
Marquês de São Vicente Street, Cardeal Leme Building, Room 101L, Gávea, Rio de Janeiro, RJ, 22453-900, Brazil
anderson@puc-rio.br, andrex.leitao@hotmail.com, vinicius.fontes@aluno.puc-rio.br

Abstract. Implementing nonlinear material models, such as hyperelasticity, in finite element software can be a daunting task for engineering students due to the complexities of transitioning from linear to nonlinear analysis and the interdisciplinary knowledge required, including tensor calculus, programming, and continuum mechanics. To address these challenges, we present HyperSym, an educational MATLAB-based tool designed to simplify the implementation of hyperelastic material models. HyperSym leverages symbolic differentiation to derive hyperelastic tensors from strain energy density functions, generating ready-to-use subroutines that integrate seamlessly with the open-source NLFEA finite element framework. This paper highlights HyperSym's core functionality, demonstrates its educational value through a numerical example suitable for lectures and assignments, and provides access to the complete MATLAB implementation via a public repository.

Keywords: Finite Element Method, Hyperelasticity, MATLAB, Symbolic Differentiation, ANSYS.

1 Introduction

The Finite Element Method (FEM) was originally developed for linear elastic problems in structural mechanics [1], but several types of nonlinear behavior due to geometry, boundary conditions, or material properties are relevant in certain applications. For instance, some types of elastomers and foams exhibit nonlinear elastic "rubbery" behavior that can be accurately described by a quasi-static and rate-independent formulation: a hyperelastic model. Yuan-Cheng Fung, who has been considered the "founder of modern biomechanics" [2], introduced a stress-strain relationship for characterizing soft biological tissues that transcended the constraints of one dimension [3]. In a following work, Tong and Fung [4] proposed a "pseudo strain potential", resembling those employed in hyperelasticity, to model preconditioned skins. These works spurred the extensive adoption of hyperelasticity in the field of biomechanics, resulting in the development of specialized software packages such as the FEBiO suite [5].

The complex derivation of FEM equations and code implementation can be simplified by leveraging symbolic manipulators like like Maple [6] or Mathematica [7]. Eyheramendy and Zimmermann [8] reviewed several of those approaches, dating back to the late 1970s, and discussed the implementation of object-oriented symbolic derivation of FEM. This object-oriented approach is the basis of many modern multiphysics systems including FreeFEM++ [9] and the FEniCS Project [10, 11], which have been used in state-of-the-art research of different types of problems such as thermal analysis [12], fluid-structure interaction [13], and hyperelasticity [14]. Other strategies like Automatic Differentiation (AD) have been implemented to other FEM softwares like AceGen [15], Moose [16], and appear in several works in the context of hyperelasticity [17–19].

While recent works like Eisenträger et al. [20] provide useful resources to simplify the implementation of hyperelastic models to FEM software, most of them require a good grasp in advanced mathematical topics like tensor analysis and calculus of variations. To bridge the gap between the conceptual presentation to practical application of hyperelastic models due to their wide range of applications, we present `HyperSym`, a framework that uses MATLAB's Symbolic Math Toolbox [21] to derive tensors from strain energy density functions and outputs ready-to-use functions. The derivation is executed at easy-to-follow component level, yielding rather efficient symbolically derived functions. `HyperSym` and its extensions was presented by Oliveira Fontes et al. [22] and its full code is available is available in our public repository at `github.com/viniciusofontes/HyperSym`.

2 Hyperelasticity

In this section we present a brief review of the fundamentals of hyperelasticity equations and definitions. Hyperelastic materials are a subset of elastic materials, which display reversible and path-independent stress-strain relationship. Mathematically, hyperelastic materials are defined by a strain-energy density function $W(\boldsymbol{E})$ that depend on the Green-Lagrange strain tensor $\boldsymbol{E}$ [2], which is defined as

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{C} - \boldsymbol{I}\right) = \frac{1}{2}\left[\boldsymbol{\nabla}_0\boldsymbol{u} + (\boldsymbol{\nabla}_0\boldsymbol{u})^T + (\boldsymbol{\nabla}_0\boldsymbol{u})^T\boldsymbol{\nabla}_0\boldsymbol{u}\right], \tag{1}$$

where $\boldsymbol{\nabla}_0\boldsymbol{u}$ is the displacement gradient with respect to the undeformed configuration, $\boldsymbol{I}$ is the second-order identity tensor. The right Cauchy-Green deformation tensor is defined as

$$\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}, \tag{2}$$

where $\boldsymbol{F}$ is the gradient of deformation tensor .

Isotropic hyperelastic models can be defined as functions of the principal invariants I_i $(i = 1, 2, 3)$ of $\boldsymbol{C}$ Nearly-compressible materials may be defined instead as function of reduced invariants

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

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

which are not affected by volume change. Instead, a separate term is typically added to control compressibility as a function of $J = \det(\boldsymbol{F})$, which measures the change in volume directly.

We compute the second Piola-Kirchhoff (PK2) stress tensor $\boldsymbol{S}$ and the tangent modulus $\boldsymbol{D}$ from the energy functional by taking the partial derivatives as follows:

$$\boldsymbol{S} = \frac{\partial W}{\partial \boldsymbol{E}} = 2\frac{\partial W}{\partial \boldsymbol{C}}, \tag{5}$$

$$\boldsymbol{D} = \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{E}} = 2\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{C}} = 4\frac{\partial^2 W}{\partial \boldsymbol{C}\partial \boldsymbol{C}}. \tag{6}$$

The simplest isotropic hyperelastic model is likely the Saint-Venant Kirchhoff (SVK) model that presents the linear stress-strain relationship in the form $\boldsymbol{S} = \boldsymbol{D} : \boldsymbol{E}$. To address its poor behavior in compression, Curnier [23], Klarbring and Strömberg [24] introduced some terms to it, generating the modeified SVK (mSVK), show in Table 1, which also includes similar modified neo-Hookean (nH) models. Here, λ and μ are the first and second Lamé parameters.

Table 1. Compressible hyperelastic models.

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

To accuratelly simulate the nearly-incompressible behavior observed in solids, and ammeliorate possible numerical problems, nearly-incompressible models like those in Table 2 present one or more independent parameters (e.g., κ and D_j) that control the compressibility factor separately. The Mooney-Rivlin (MR) and Yeoh models are often found in commercial FEM softwares with nearly-incompressible formulations.

Table 2. Nearly incompressible hyperelastic models.

Symbol	W^{iso}	W^{vol}	**Parameters**
MR	$A_{10}(J_1-3)+A_{01}(J_2-3)$	$\frac{\kappa}{2}(J-1)^2$	A_{10}, A_{01}, κ
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

3 MATLAB implementation

In this work, we used a modified version of `NLFEA`. whose original source code is available at [25]. In what follows, we describe how to generate a function for the nearly-incompressible MR model in Table 2 using `HyperSym` and how to incorporate it to the original `NLFEA` by changing a single line of code.

3.1 Input data in `HyperSymScript`

The main script for `HyperSym`, `HyperSymScript` (available in Appendix A of [22]) starts by defining the strain energy density function W of the desired model and its properties. These are stored in variables `W` and `PROP`, respectively, in terms of a suitable set of predefined symbolic variables (tensor invariants and material properties). This is shown in the lines below

```
 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;
```

where `Symbol` is a character array containing the name abbreviation of the functions to be generated. Next, the main script calls `HyperSym` to derive the tensors `S` and `D`:

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

On line 15 the tensors `Sv` and `Dv` are stored in the contracted matrix notation. The exception is the right Cauchy-Green tensor `C` which is given in matrix format, so it can be easily computed from eq. (2).

Finally, `HyperSymScript` generates a MATLAB function and stores 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\nElapsed time: %1.1f seconds. \n\n',toc)
```

To use a different material model one needs to properly modify the variable `Symbol`, the vector `PROP`, and the energy function `W`. For instance, for the SVK model (see Table 1), lines 8-12 must be changed to:

```
 8 %% Symbolic variables
 9 syms I1 I2 mu lambda real
10 Symbol = 'SVK'; % SVK: St. Venant-Kirchhoff
11 PROP = [lambda mu];
12 W = 1/8*lambda*(I1 - 3)^2 + mu/4*(I1^2 - 2*I2 - 2*I1 + 3);
```

3.2 Tensor derivation using `HyperSym`

`HyperSym` (available in Appendix B of [22]) derives the tensors $\boldsymbol{S}$ and $\boldsymbol{D}$ for any hyperelastic model (defined in terms of either direct or reduced invariants). The idea behind `HyperSym` is essentially to compute the

first and second derivatives of W with respect to the components of $\boldsymbol{C}$. First we define the deformation metric in terms of the components C_{ij}, and substitute it in the variable `W` provided as an input. On lines 11-17 we define the invariants of $\boldsymbol{C}$ and substitute them in `W` (line 17). Nevertheless, the user is free to add any other metrics required to define other hyperelastic models.

```
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});
```

Now we can efficiently compute the 2nd Piola-Kirchhoff stress tensor via eq. (5) with the MATLAB Symbolic Math Toolbox's function `diff` as shown on lines 18-25. On the last line, we impose the symmetry of the PK2 stress after taking the derivatives, as done by Suchocki [26].

```
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, we derive `S` with respect to the components of `C` following eq. (6) 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
```

Finally, in the last lines of `HyperSym`, we impose the symmetry of tensors `S` and `D` and write `Sv` and `Dv` in Voigt notation.

3.3 Incorporating hyperelastic model into `NLFEA`

The functions generated by the `HyperSymScript` (section 3.1) can now be incorporated into the `NLFEA` framework, that originally features solely the Mooney-Rivlin model. In what follows, we specify how to change the MATLAB function `Mooney` from the original `NLFEA` to the one obtained via `HyperSym`, i.e., `MR`. To do so, one should update the function `HYPER3D`, which computes and assembles the internal force vector and tangent stiffness matrix. This is done by replacing line 42 of `NLFEA` which 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 in matrix form, so `F'*F` is 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 Mooney-Rivlin 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 representation. `LTAN` is not required for `MR`, because it only computes `DTAN` when there is more than one output argument.

4 Numerical examples

This section is dedicated to present the solution of a benchmark proble using hyperelastic models whose functions were derived with `HyperSym`. The finite element mesh is composed of 8-node linear hexahedrals (brick) elements, and results are compared to those obtained ANSYS Parametric Design Language (APDL) [27] and the literature. Additionally, we set the convergence tolerance of the Newton-Raphson method to 10^{-6} and used the bissection strategy detailed in the textbook by the original author of `NLFEA` [28].

The chosen example deals with an infinite thick-walled cylinder subject to internal pressure $p_{in} = 150$ psi (1.034 MPa), vid. Fig. 1. 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 $20 \times 20 \times 1$ linear brick elements.

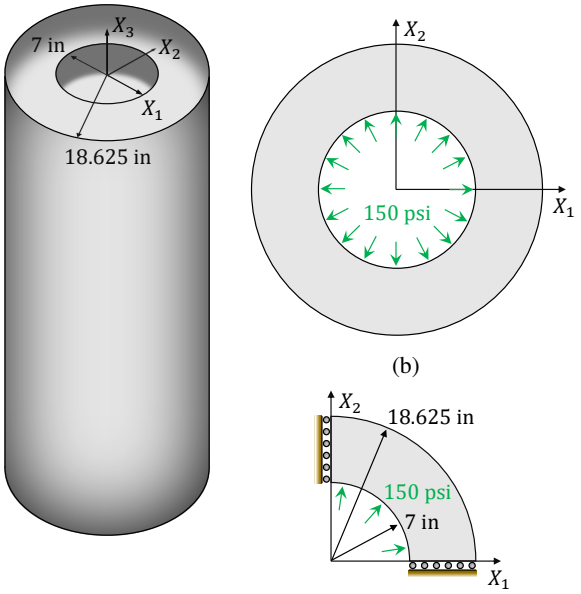


Figure 1. Thick-walled cylinder: (a) 3D model; (b) top view, illustrating the internal pressure; (c) FE model: one-quarter of the initial geometry, and boundary conditions.

In this example, we decided to use a convergence tolerance of 0.01 to speed up the analysis, without altering the final solution significantly while achieving convergence. We adopted the MR model with the following parameters in order to enable comparison with Kim [28]: $A_{10} = 80$ psi (0.5516 MPa), $A_{01} = 20$ psi (0.1379 MPa) and $\kappa = 10^4$ psi (68.95 MPa).

Through numerical experimentation, we found that dividing the loads into 10 equal increments was sufficient to achieve results close to the target values of Table 3. Though the load is path dependent (as the nodal forces change with the change of the area where the internal pressure is applied), the load step is small enough such that convergence was still achieved without introducing the load-dependent term to the consistent stiffness matrix. Kim [28] computed the stress at the center of the first element in a mesh with five elements in the thickness. This corresponds to 10% of its value, whereas we interpolated the radial component of the stress at the nodes.

Table 3. Target and this work's results for the infinite thick-walled cylinder problem considering MR material model.

Criterion	**Units**	**Target value** [28]	**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	

The subsequent results are presented for nodes along a radial path and at $X_3 = t$ in the undeformed configuration. Fig. 2 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 FEM implementation – using `NLFEA` with material model functions generated by `HyperSym` – and those obtained in ANSYS, except at the inner and outer surfaces of the cylinder. At these locations, the boundary affects the stresses, and it is more severe in the radial component, σ_r. This behavior was also observed in the works of Oden and Key [29] and Scharnhorst and Pian [30]. In order to improve the accuracy of the stress distribution, one may use more advanced techniques (stress recovery, selective reduced integration, to name a few), which are beyond the scope of this work.

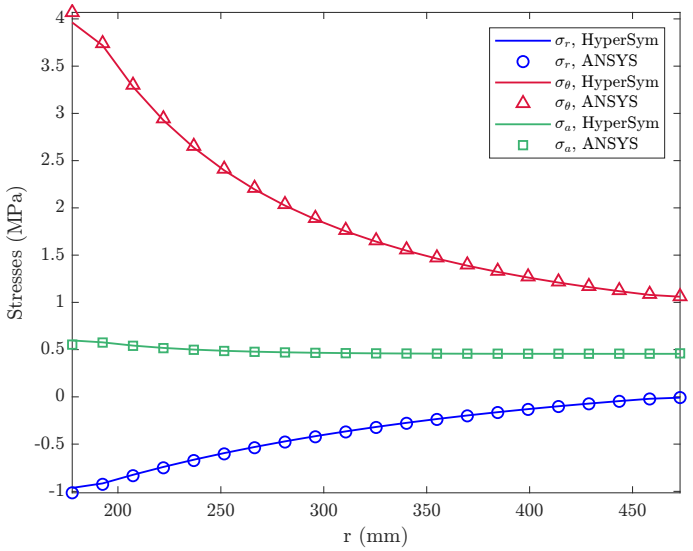


Figure 2. Equilibrium path for a node at the inner wall of cylinder considering MR material model.

5 Final remarks

In this work we presented a symbolic tool for the derivation of tensors required for the implementation of hyperelastic materials, `HyperSym`. Unlike full-fledged FEM softwares with symbolic manipulation, `HyperSym` only derives the tensors used to compute local matrices. Despite the additional effort required, we believe this represents a reasonable compromise between the convenience and generality of a symbolic derivation algorithm.

Although previous publications have presented similar tools, they have some drawbacks, such as being dependent on commercial softwares, being too complex for rapid prototyping, or being difficult to adapt to other applications. The presented solution is designed to be simple and facilitate modifications and extensions to different class of problems. Additional examples and extensions to `HyperSym` are available in our public repository, and we expect they will be useful tools for engineers and scientists.

Acknowledgements. 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).

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] K.-J. Bathe. *Finite Element Procedures*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1st edition, 1996.
[2] R. Hackett. *Hyperelasticity Primer*. Springer, 2016.
[3] Y.-C. Fung. Biorheology of soft tissues. *Biorheology*, vol. 10, n. 2, pp. 139–155, 1973.
[4] P. Tong and Y.-C. Fung. The stress-strain relationship for the skin. *Journal of Biomechanics*, vol. 9, n. 10, pp. 649–657, 1976.

[5] S. Maas, B. Ellis, G. Ateshian, and J. Weiss. Febio: Finite elements for biomechanics. *Journal of Biomechanical Engineering*, vol. 134, n. 1. 011005, 2012.
[6] Maplesoft. *Maple User Manual*. Maplesoft, a division of Waterloo Maple Inc., 1996-2023.
[7] I. Wolfram Research. *Mathematica, Version 13.3*. Wolfram Research, Inc., Champaign, Illinois, USA, 2023.
[8] D. Eyheramendy and T. Zimmermann. Object-oriented symbolic derivation and automatic programming of finite elements in mechanics. *Engineering with Computers*, vol. 15, pp. 12–36, 1999.
[9] F. Hecht. New development in freefem++. *Journal of Numerical Mathematics*, vol. 20, n. 3-4, pp. 251–265, 2012.
[10] A. Logg, K.-A. Mardal, and G. Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.
[11] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. Rognes, and G. Wells. The fenics project version 1.5. *Archive of Numerical Software*, vol. 3, n. 100, 2015.
[12] M. Carraturo, A. Viguerie, A. Reali, and F. Auricchio. Two-level method part-scale thermal analysis of laser powder bed fusion additive manufacturing. *Engineering with Computers*, vol. 38, pp. 1–14, 2022.
[13] G. Neighbor, H. Zhao, M. Saraeian, M.-C. Hsu, and D. Kamensky. Leveraging code generation for transparent immersogeometric fluid–structure interaction analysis on deforming domains. *Engineering with Computers*, vol. 39, pp. 1–22, 2022.
[14] A. Mazier, A. Bilger, A. E. Forte, I. Peterlik, J. Hale, and S. Bordas. Inverse deformation analysis: an experimental and numerical assessment using the fenics project. *Engineering with Computers*, vol. 38, n. 5, pp. 4099–4113, 2022.
[15] J. Korelc. Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with computers*, vol. 18, pp. 312–327, 2002.
[16] C. Permann, D. Gaston, D. Andrš, R. Carlsen, F. Kong, A. Lindsay, J. Miller, J. Peterson, A. Slaughter, R. Stogner, and R. Martineau. Moose: Enabling massively parallel multiphysics simulation. *SoftwareX*, vol. 11, pp. 100430, 2020.
[17] J. Young, J. Yao, A. Ramasubramanian, L. Taber, and R. Perucchio. Automatic generation of user material subroutines for biomechanical growth analysis. *Journal of Biomechanical Engineering*, vol. 132, n. 10, 2010.
[18] A. Vigliotti and F. Auricchio. Automatic differentiation for solid mechanics. *Archives of Computational Methods in Engineering*, vol. 28, n. 3, pp. 875–895, 2021.
[19] M. Hillgärtner, T. Guo, and M. Itskov. Automatic differentiation of strain-energy functions in the context of user-defined materials for the fem. *Proceedings in Applied Mathematics and Mechanics*, vol. 20, n. 1, pp. e202000050, 2021.
[20] S. Eisenträger, L. Maurer, D. Juhre, H. Altenbach, and J. Eisentraeger. Implementation of isotropic hyperelastic material models: a≫ template ≪approach. *Acta Mechanica*, vol. 236, pp. 1–36, 2025.
[21] T. M. Inc. Matlab Version 9.12.0.1956245 (R2022a). https://www.mathworks.com/products/matlab, 2022.
[22] V. Oliveira Fontes, A. Xavier Leitão, and A. Pereira. Hypersym: An educational matlab code for hyperelasticity. *Computer Applications in Engineering Education*, vol. 33, n. 3, pp. e70037, 2025.
[23] A. Curnier. *Computational Methods in Solid Mechanics*, volume 29. Kluwer Academic Publishers, Dordretch, The Netherlands, 1994.
[24] A. Klarbring and N. Strömberg. Topology optimization of hyperelastic bodies including non-zero prescribed displacements. *Structural and Multidisciplinary Optimization*, vol. 47, n. 1, pp. 37–48, 2013.
[25] N.-H. Kim. Introduction to nonlinear finite element analysis. `https://mae.ufl.edu/nkim/INFEM/`. Accessed: 2023-07-27, 2023.
[26] C. Suchocki. Finite element implementation of slightly compressible and incompressible first invariant-based hyperelasticity: theory, coding, exemplary problems. *Journal of Theoretical and Applied Mechanics*, vol. 55, 2017.
[27] A. Inc. *Ansys® Help System, Release 2021 R1*. Canonsburg, PA. ANSYS® Mechanical APDL Product Launcher, 2021.
[28] N.-H. Kim. *Introduction to Nonlinear Finite Element Analysis*. Springer Science & Business Media, New York, NY, 2014.
[29] J. Oden and J. Key. Numerical analysis of finite axisymmetric deformations of incompressible elastic solids of revolution. *International Journal of Solids and Structures*, vol. 6, n. 5, pp. 497–518, 1970.
[30] T. Scharnhorst and T. Pian. Finite element analysis of rubber-like materials by a mixed model. *International Journal for Numerical Methods in Engineering*, vol. 12, n. 4, pp. 665–676, 1978.