FerriteHyperelastic: A Julia Package for Finite Element Analysis of Hyperelastic Materials

Amin Alibakhshi[a,c*], Kian Aghani [b*], Luis Saucedo-Mora[c], Guillermo Lorenzo [a], Kevin M. Moerman[d]

[a] *Group of Numerical Methods in Engineering, Department of Mathematics and CITEEC, University of A Coruña, A Coruña, Spain*

[b] *Research assistant, Sahand University of Technology, Tabriz, Iran*

[c] *E.T.S. de Ingeniería Aeronáutica y del Espacio, Universidad Politécnica de Madrid, Pza. Cardenal Cisneros 3, 28040, Madrid, Spain*

[d] *Mechanical Engineering, University of Galway, Galway, Ireland*

[*] *Amin Alibakhshi and Kian Aghani contributed equally to this work.*

Abstract

Hyperelastic models are ubiquitous in many areas of science and engineering. For example, they are extensively used to describe the mechanical behaviour of soft, rubber-like, and biological materials. However, finite element analysis (FEA) for hyperelastic materials poses unique challenges, particularly in terms of computational efficiency and implementation. In this paper, we introduce the novel open source Julia package FerriteHyperelastic, which enables accurate and efficient application of FEA to hyperelastic materials. In addition, the package features hyperelastic model fitting to data, in conjunction with the stability analysis. The package utilises automatic differentiation, enabling easy implementation of hyperelastic formulations and their derivatives. Several example problems were implemented and results showed excellent agreement with literature data as well as outputs from the FEA solver FEBio. The package can be used for a range of large deformation analysis and biomechanical applications.

Keywords: finite element, hyperelasticity, Julia language, curve fitting, biomechanics

1. Introduction

Efficient programming of finite element analysis (FEA) for hyperelasticity poses several theoretical and computational challenges. This includes the treatment of material and geometric nonlinearities, convergence of the nonlinear problem, and handling of nonlinear constraints (such as those caused by the plane-stress assumption [1, 2]).

Several commercial and open source FEA packages with hyperelasticity functionality have been introduced.

Although commercial software functionality tends to be complete and robust, their high cost and closed-source nature poses barriers to wide adoption, customisation, extension, and integration. Conversely, open source packages are free, do not have such barriers, and are developed for research purposes. Typically these packages are written in traditional high performance languages such as Fortran, C, or C++ (e.g. FEBio [3]). However, the significant learning effort and verbose syntax associated with these languages can obscure the elegance of the FEA method and hinder accessibility for new users.

High-level languages, such as Python, are easier to learn, and have also been explored for FEA, and examples include FEniCS [4] and FElupe [5]. However, Python's interpreted nature causes performance limitations, especially when nested loops are required. To combat these limitations, core functionality is sometimes offloaded to C++ implementations. However, this negates the accessibility that Python could otherwise offer.

The Julia language [6] targets this "two-language problem" in scientific computing by combining an easy-to-learn syntax, that is often close to mathematical notation, with high performance. These features, as well as multiple dispatch, just-in-time compilation, and metaprogramming, make Julia ideal for FEA implementations, and examples include Ferrite [7] and Gridap [8]. Gridap and its successor GalerkinToolkit [9], mainly focus on high abstract level and declarative frameworks. Ferrite provides a low-level and explicit framework that gives users direct control over the FEA assembly loop, emphasising flexibility and extensibility. This design facilitates the implementation of sophisticated constitutive models and custom element formulations, which is particularly advantageous for research applications. Although both packages represent important progress for performing FEA in Julia, their built-in capabilities for hyperelasticity are still limited. For example, the implementation of advanced and specialised hyperelastic formulations, robust plane-stress/plane-strain formulations, complex geometries and consideration of complex boundary conditions have not been well addressed (i.e., the documentation for both packages only provides a single simple example, mainly to demonstrate the basic tutorial and foundational usage of each package; Ferrite [10] and Gridap [11]).

This paper presents FerriteHyperelastic, a Julia package featuring a new FEA implementation for hyperelasticity. The package is based on Ferrite and implements 2D and 3D analysis for incompressible, nearly incompressible, and compressible hyperelastic materials. In addition, it provides curve-fitting subroutines for several widely used hyperelastic strain energy functions.

2. Mathematics of hyperelasticity

The FerriteHyperelastic implementation is based on the principle of virtual work, which states that the virtual internal and external energy are equal, namely $\delta W_{int} = \delta W_{ext}$. Using the total Lagrangian method, the formulation is expressed in the reference configuration, for which the internal virtual work is

$$\delta W_{int} = \int_{\Omega_0} \mathbf{P} : \nabla \delta \mathbf{u} \, d\Omega_0 \tag{1}$$

where $\delta \mathbf{u}$ is an admissible virtual displacement vector, Ω_0 the reference domain, $\mathbf{P}$ is the first Piola-Kirchhoff tensor, and ∇ is the gradient with respect to the reference configuration. The first Piola-Kirchhoff tensor is obtained from the strain energy function ψ as $\mathbf{P} = \partial \Psi / \partial \mathbf{F}$, where $\mathbf{F}$ is the deformation gradient tensor. The strain energy function can be formulated for compressible, nearly incompressible , and purely incompressible hyperelastic materials. Each is implemented here and typical isotropic representations are given in Table 1.

Table 1: Three types of isotropic hyperelastic formulations.

Formulation Type	**Strain Energy Function**
Constrained formulations (incompressible formulations)	$\Psi = \Psi(I_1, I_2) - p(J-1)$
Unconstrained or coupled formulations (compressible formulations)	$\Psi = \Psi(I_1, I_2, J)$
Uncoupled formulations (nearly incompressible formulations)	$\Psi = \Psi_{dev}(\tilde{I}_1, \tilde{I}_2) + \Psi_{vol}(J)$

For implementation of uncoupled (nearly incompressible) materials, the three-field formulation of Simo and Taylor[12] is used, featuring the separate treatment of the deviatoric and volumetric contributions

$$\Psi = \Psi_{dev}(\tilde{\mathbf{C}}) + p(J-\theta) + \Psi_{vol}(\theta) \tag{2}$$

where $\tilde{\mathbf{C}} = \tilde{\mathbf{F}}^\top \tilde{\mathbf{F}}$ with $\tilde{\mathbf{F}} = J^{-\frac{1}{3}}\mathbf{F}$, and $J = \det(\mathbf{F})$. Here $\Psi_{vol}(\theta) = \frac{\kappa}{2}\ln(\theta)^2$ is used, with κ is the bulk modulus, and θ an independent volumetric variable (other volumetric formulations may also be considered [13]).

FerriteHyperelastic supports prescribed displacements, nodal forces, surface tractions, pressures, and body forces. For the case of surface traction, both dead load and follower load [14] are implemented for which respectively the external virtual work is expressed as

$$\delta W_{ext} = \oint_{\Gamma_0} \mathbf{t}_0 \delta\mathbf{u}\, d\Gamma_0 \tag{3}$$

$$\delta W_{ext} = \oint_{\Gamma_0} \mathbf{t} J(\mathbf{u}) \sqrt{\mathbf{N}\mathbf{C}(\mathbf{u})^{-1}\mathbf{N}}\, \delta\mathbf{u}\, d\Gamma_0 \tag{4}$$

where $\mathbf{t}_0$ and $\mathbf{t}$ are surface tractions per unit initial and current area respectively.

Equation (2) is for isotropic materials. To model anisotropy an additional strain energy term Ψ_{fib} can be added. An example implementation is provided here featuring an exponential power law (see Myers et al. 2015[15] and the FEBio user manual):

$$\Psi_{fib}(I_n) = H(I_n - I_0)\frac{\xi}{\alpha\beta}(e^{\alpha(I_n - I_0)^\beta} - 1) \tag{5}$$

Here ξ, α, and β are material parameters with the constraints: $\xi > 0$, $\alpha \geq 0$, and $\beta \geq 2$. The invariant I_n is given by:

$$I_n = \mathbf{n} \cdot \mathbf{C}\mathbf{n} = \lambda_n^2 \tag{6}$$

with $\mathbf{C} = \mathbf{F}^\top\mathbf{F}$ the right Cauchy-Green tensor, $\mathbf{n}$ the fibre direction vector, and λ_n the fibre stretch. The quantity I_0, given by:

$$I_0 = \lambda_0^2 \tag{7}$$

features λ_0, the tensile stretch threshold at which fibres start to act. Finally, the Heaviside step function, $H(\cdot)$, enforces tension-only reinforcement. Finally, the Cauchy stress is derived from:

$$\boldsymbol{\sigma}_{fib} = H(I_n - I_0)\frac{2I_n}{J}\frac{\partial \Psi_{fib}}{\partial I_n}\mathbf{n} \otimes \mathbf{n} \tag{8}$$

The fibre strain energy function of Equation (5) can be used to define a transversely isotropic material when combined with an isotropic hyperelastic formulation to act as ground matrix.

3. Computational functionalities

FerriteHyperelastic is built on top of the FEA library Ferrite [7], which provides several useful functionalities, including support for different element types, control over the number of quadrature points, and specification of the interpolation order. In addition, the Ferrite plugin FESolvers [16], enabled nonlinear solver and time stepping algorithm definition. Furthermore, this work leverages functionality from Julia packages like Comodo [17] to enable complex geometry processing and meshing, which is highly important for biomedical applications.

4. Illustrative examples

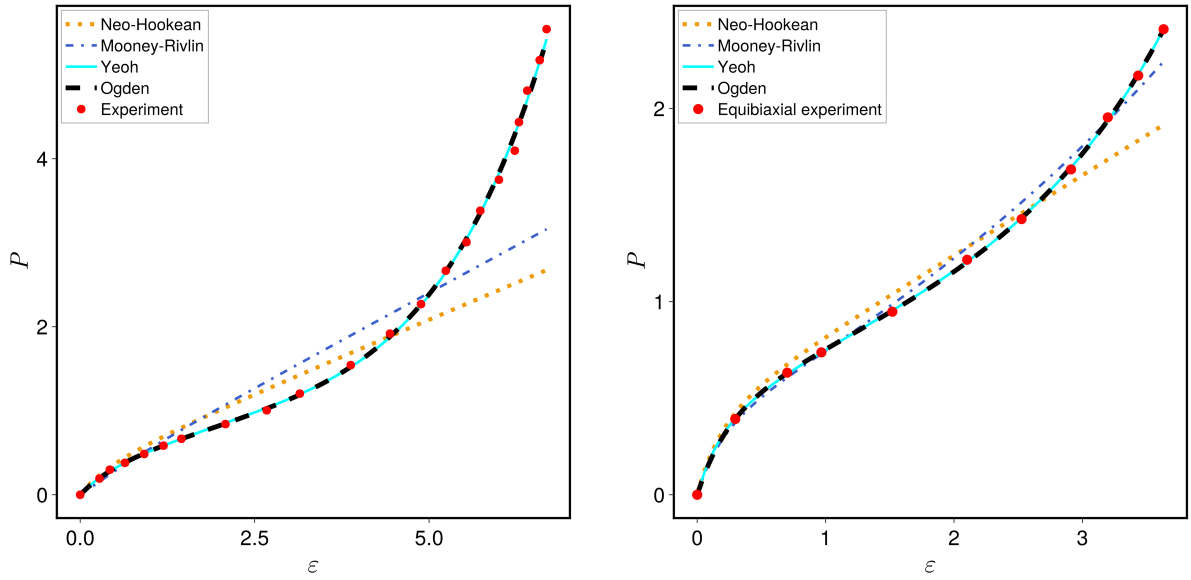
Here, we present curve-fitting and finite element examples for hyperelastic materials, as well as evaluation of the code against literature data and equivalent FEBio implementations. The FEBio implementations were based on the Julia wrapper FEBio.jl [18].

4.1. Curve fitting

A major challenge when modelling hyperelastic materials is parameter identification. Here parameter estimation based on curve fitting to experimental data is presented. Discrepancies between experimental data and model predictions are minimised using the Levenberg-Marquardt nonlinear least-squares method. In addition, to ensure parameter choices are physically permissible and stable only parameters yielding positive definite material tensors are permitted. The latter features the Drucker stability criterion, which requires the incremental work done by stress $\boldsymbol{\sigma}$ during a strain increment $\boldsymbol{\varepsilon}$ to be non-negative ($\mathrm{d}\boldsymbol{\sigma} : \mathrm{d}\boldsymbol{\varepsilon} \geq 0$). Table 2 lists the formulations currently implemented for curve fitting, and users may adjust these to add their own implementations.

Table 2: Hyperelastic formulations implemented for curve fitting (see also: [19, 20])

Formulation	Strain energy density	Number of parameters
neo-Hookean	$\Psi = C_{10}(I_1 - 3)$	1
Mooney-Rivlin	$\Psi = C_{10}(I_1 - 3) + C_{01}(I_2 - 3)$	2
Yeoh	$\Psi = C_{10}(I_1-3)+C_{20}(I_1-3)^2+C_{30}(I_1-3)^3$	3
Ogden (3-term)	$\Psi = \sum_{p=1}^{N} \frac{2\mu_p}{\alpha_p^2}\left(\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p} + \lambda_3^{\alpha_p} - 3\right)$	6



(a) Uniaxial test (b) Equibiaxial test

Figure 1: Curve fitting to the experimental data by Treloar et al. [21]. P denotes the nominal stress and ε denotes linear (Biot) strain.

The curve-fitting capability is evaluated using Treloar's experimental rubber data [21]. Figure 1 compares the model predictions with the measurements, while Table 3 lists the fitted parameters. The Yeoh and Ogden models closely reproduce the data, whereas the neo-Hookean and Mooney–Rivlin models fail to capture strain stiffening.

Table 3: Fitted model parameters for uniaxial and equibiaxial tests.

Model	Uniaxial		Equibiaxial	
	R^2	Parameters	R^2	Parameters
neo-Hookean	0.568281	$C_1 = 0.174468$	0.927637	$C_1 = 0.206858$
Mooney–Rivlin	0.732441	$C_1 = 0.223574$ $C_2 = -0.133790$	0.991355	$C_1 = 0.177264$ $C_2 = 0.003027$
Yeoh	0.998875	$C_1 = 0.151009$ $C_2 = -0.000848$ $C_3 = 3.1 \times 10^{-5}$	0.999882	$C_1 = 0.194033$ $C_2 = -0.000528$ $C_3 = 2.3 \times 10^{-5}$
Ogden	0.998838	$\mu_1 = 0.366079$ $\alpha_1 = 1.810491$ $\mu_2 = 0.000178$ $\alpha_2 = 6.448410$ $\mu_3 = -0.082103$ $\alpha_3 = -4.239964$	0.999890	$\mu_1 = 0.391319$ $\alpha_1 = 1.825820$ $\mu_2 = -0.118011$ $\alpha_2 = 6.285254$ $\mu_3 = 0.059539$ $\alpha_3 = -3.141955$

4.2. Finite element analysis

In this section, we demonstrate the efficiency and accuracy of FerriteHyperelastic for FEA of hyperelasticity using four benchmark problems: (i) Uniaxial tension of a cube, (ii) twisting of a beam, (iii) uniaxial compression of a transversely isotropic cube, and (iv) Cook's membrane subjected to a traction load. The geometries, boundary conditions, and deformed states for each are visualised in Figure 2. For each example the results are compared either to results from equivalent FEBio implementations or directly to literature data. In addition the root mean squared error (RSME) is computed and listed for each example. For each example the models are constructed using Comodo [17]. Hexahedral meshes could be generated directly while Comodo's TetGen [22] integration was used for tetrahedral meshing. Finally all visualisations presented here were created using Comodo's Makie [23] integration.

4.2.1. Example 1: Uniaxial tension of a cube

In this example a 10x10x10 mm cube is subjected to 60% uniaxial extension. The cube was meshed using 125 trilinear hexahedral elements. The material is modelled using the following compressible isotropic neo-Hookean formulation:

$$\Psi = \frac{\mu}{2}(I_1 - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2 \tag{9}$$

where $I_1 = \mathrm{tr}(\mathbf{C})$, and $J = \det(\mathbf{F})$. The material parameters μ and λ (Lame's constants) can be related to the Young's modulus E and Poisson's ratio ν using:

$$\mu = \frac{E}{2(1+\nu)} \tag{10}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{11}$$

This example uses $E = 1$ MPa and $\nu = 0.4$.

4.2.2. Example 2: Twisting of a beam

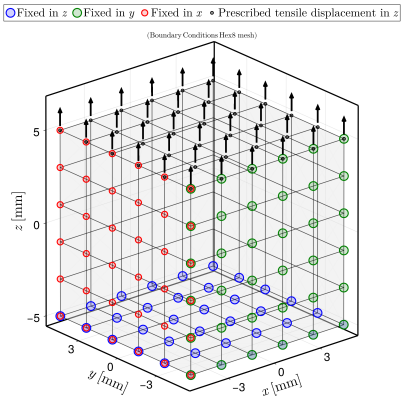
In this example a 10x40x10 mm beam is constrained on one end and subjected to a 360 degree prescribed longitudinal twist at the other end. The mesh features 500 trilinear hexahedral elements. The material is modelled using the following uncoupled Ogden formulation:

$$\Psi = \frac{c_i}{m_i^2}(\tilde{\lambda}_1^{m_i} + \tilde{\lambda}_2^{m_i} + \tilde{\lambda}_3^{m_i} - 3) + \frac{\kappa}{2}(\ln J)^2 \tag{12}$$

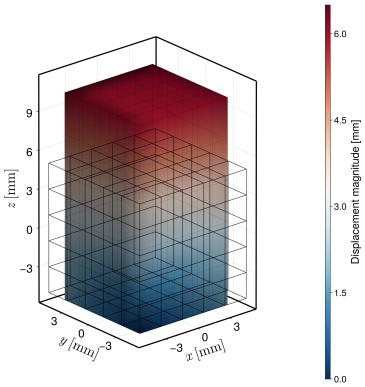
where $\tilde{\lambda}_i = J^{-\frac{1}{3}}\lambda_i$, with λ_i are the principal stretches. For this case, we use $c_1 = 1$ kPa, $m_1 = 12$, $c_2 = c_1$, $m_2 = -m_1$, and $\kappa = 100(c_1 + c_2)/2 = 0.1$ MPa. (note the use of $c_2 = c_1$, $m_2 = -m_1$ yields the symmetry defined in [24]).

4.2.3. Example 3: Uniaxial compression of a transversely isotropic cube

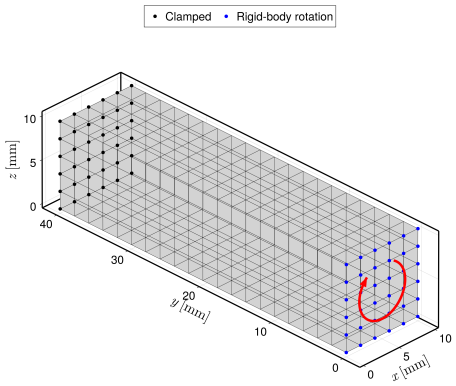
The geometry and boundary conditions are the same as for example 1. The only difference is that the material is subjected to 50% compression and that fibrous reinforcement of the neo-hookean material is added as per the formulation of Equation (5). The following parameters are used $\xi = 1$ MPa, $\alpha = 0, \beta = 2, \theta = 90°$, $\phi = 90°$, $E = 1$ MPa, and $\nu = 0.4$. Note that the fibres are oriented along the y-direction and hence resist expansion in this direction.



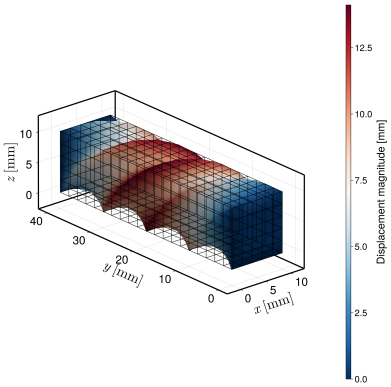
(a) Uniaxial tension of an isotropic cube



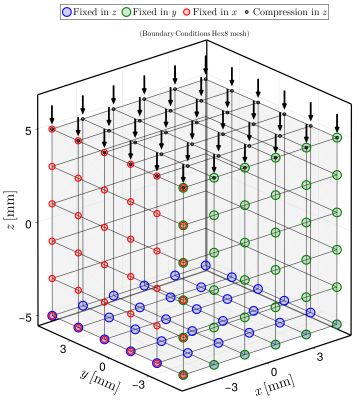
(b) Deformed state



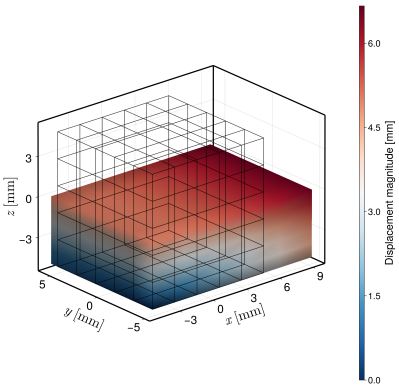
(c) Twisting of a beam



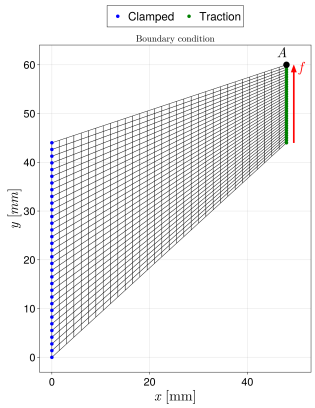
(d) Deformed state



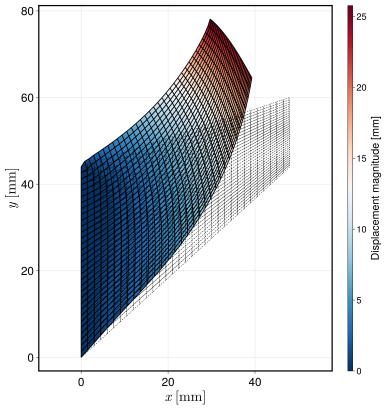
(e) Uniaxial compression of a transversely isotropic cube



(f) Deformed state



(g) Cook's membrane



(h) Deformed state

Figure 2: The geometry, applied loads, boundary conditions, and deformed states relevant to the problems under analysis, with lengths in mm. (a), (b) Uniaxial tension cube: $n_x = n_y = n_z = 5$, linear hexahedral element type, and applied displacement $u_z = 6$ mm with the compressible neo-Hookean model. (c), (d) Twist of a beam: $n_x = 5$, $n_y = 20$, $n_z = 5$, linear hexahedral element type, and $\theta = 2\pi$ with the nearly incompressible Ogden model. (e), (f) Uniaxial compression cube: $n_x = n_y = n_z = 5$, linear hexahedral element type, and applied displacement $u_z = -5$ mm with the fibrous exponential power-law model. (g), (h) Cook's membrane: $n_x = n_y = 32$, linear quadrilateral element type, with the nearly incompressible neo-Hookean model.

4.2.4. Example 4: Cooke's membrane

For the Cook's membrane, we employ the model and parameters in Ahmadi et al. [1]. The model consists of 1024 linear quadrilateral elements. The material is modelled using the following uncoupled formulation

$$\Psi = \frac{\mu}{2}(\tilde{I}_1 - 3) + \frac{\kappa}{4}(J^2 - 1 - 2\ln J) \tag{13}$$

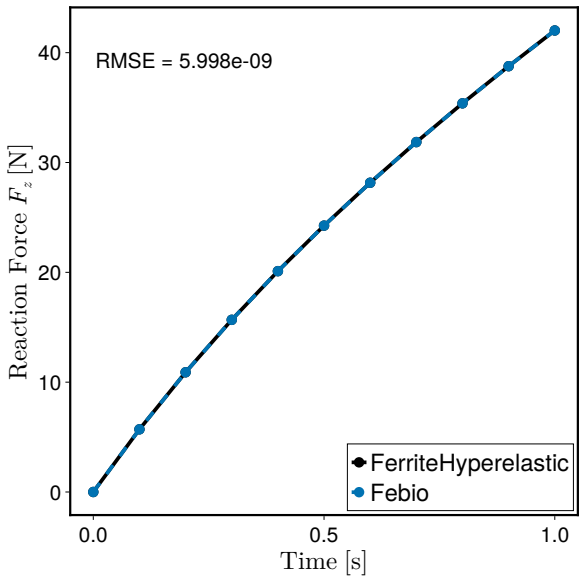
where $\tilde{I}_1 = J^{-\frac{2}{3}} I_1$, κ is the bulk modulus. The latter can be related to μ and the Poisson's ratio ν using:

$$\kappa = \frac{2\mu(1+\nu)}{3-6\nu} \tag{14}$$

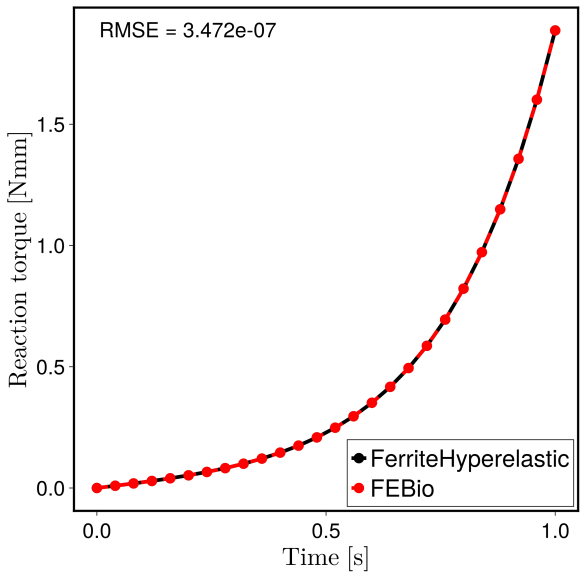
Here $\mu = 80.1938$ MPa was used, and κ was set such that $\nu = 0.4999$.

4.2.5. Example implementation findings

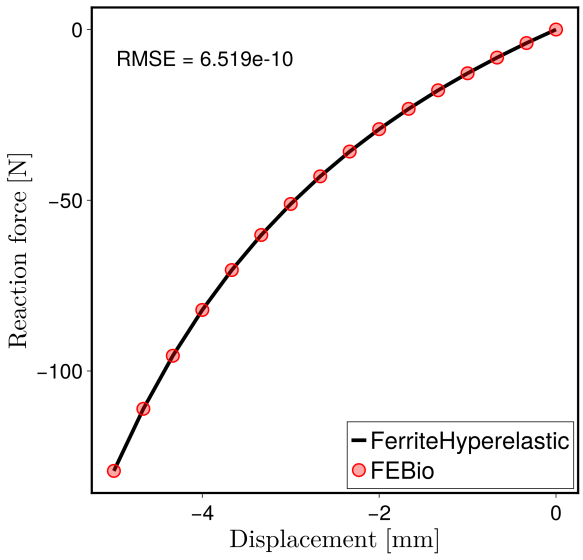
Figure 2 shows the deformed states, and Figure 3 presents the comparisons. For the cubes and beam, we compare the results with FEBio, and for the Cook's membrane, with Ahmadi et al. [1]. The graphs clearly show FerriteHyperelastic is in close agreement with the data from Ahmadi et al., while a near perfect match to the FEBio implementation is obtained.



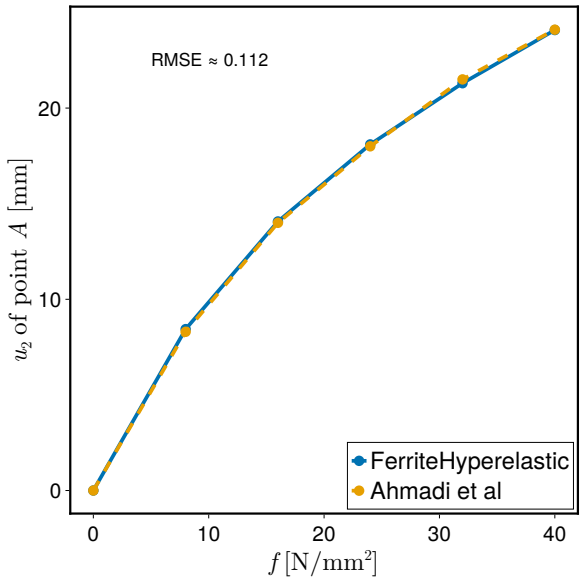
(a) Uniaxial tension of cube



(b) Twist of beam



(c) Uniaxial compression of cube



(d) Cook's membrane

Figure 3: (a) Reaction force vs. time steps for uniaxial tension of the cube. (b) Reaction torque vs. time for the twist of the beam. (c) Reaction force vs. displacement of the uniaxial compression of the cube. (d) The u_2 displacement of point A, see Fig. 2.

Performance of FerriteHyperelastic was estimated by comparison to FEBio. Computational cost evaluation depends on the specific hardware, but here a laptop was used (MacBook Air M2, 8-core @3.49 GHz, 8 GB RAM). The comparisons used the uniaxial cube model and the number of elements and threads were varied (Table 4). Parallelisation significantly reduced computational time and brought performance close to that of FEBio.

4.3. Biomedical applications

This section presents two example biomedical applications: 1) the human prostate subjected to Winkler boundary conditions, 2) and the human breast subjected to gravity

Table 4: Computational time comparison between FerriteHyperelastic and FEBio for different numbers of finite elements.

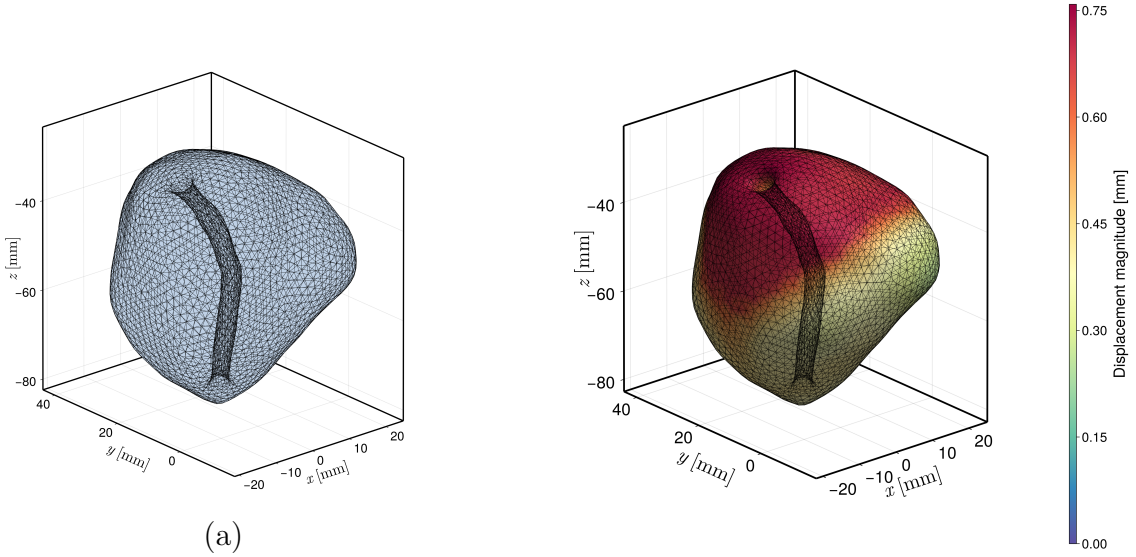
Case	Number of elements	FerriteHyperelastic one thread [s]	FerriteHyperelastic four threads [s]	FEBio four threads [s]
1	125	0.76	0.62	0.28
2	1000	2.4	1.46	1.5
3	3375	6.9	5.5	5.3

loading conditions.

4.3.1. Human prostate

Biomechanical modelling of the prostate is relevant to prostate cancer and benign prostatic hyperplasia research [25]. The prostate was here modelled using the neo-Hookean formulation of equation (9). Next Winkler boundary conditions [25] were applied to the external surface, while the urethra was traction free.

The prostate geometry was obtained through segmentation of anonymised MRI data (public repository: `i2cvb.github.io`) [26] using 3D Slicer [27]. Segmentated surfaces were remeshed using a Geogram [28] implementation in Julia [29], and the domain was meshed using 113372 linear tetrahedral elements. Figure 4 shows the mesh and deformed state.

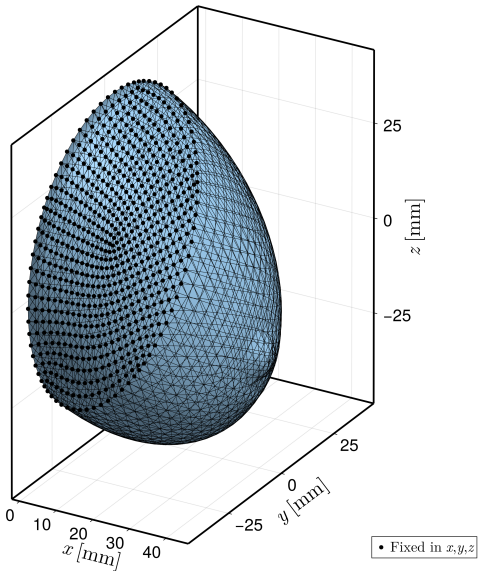


(a) (b)

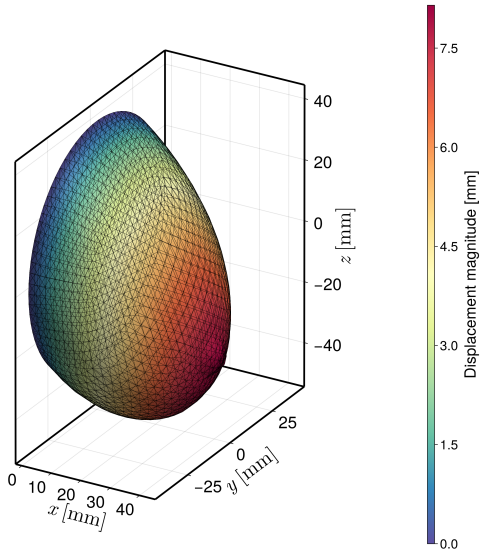
Figure 4: Mesh and displacement magnitude of the prostate. The Young's modulus of the peripheral zone was $E_{PZ} = 3$ kPa, while that of the central gland was $E_{CG} = 6$ kPa. The Poisson's ratio was $\nu = 0.4$ for both regions.

4.3.2. Human breast

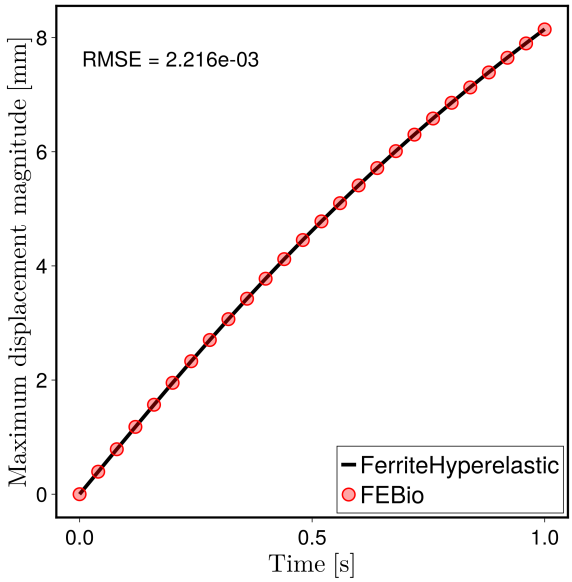
The second example is the deformation of a breast under gravity and modelled with the Ogden model of equation (12). The breast mesh is obtained from the parameterised model in Mammo.jl [30], featuring 7609 linear tetrahedral elements. Figure 5 shows the mesh and deformed state with a comparison to Mammo's FEBio implementation.



(a) Geometry, mesh, and boundary condition



(b) Displacement magnitude on the surface



(c) Maximum displacement versus time

Figure 5: Human breast gravity loading. (a) Geometry, mesh, and boundary conditions, and (b) Surface displacement magnitude, and (a) Maximum displacement magnitude for FerriteHyperelastic and FEBio. The Ogden material parameters set as $c_1 = 0.666$ kPa, $c_2 = c_1$, $m_1 = 12$, $m_2 = -m_1$, and $\kappa = 100 \cdot c_1 = 66.6$ kPa.

.

5. Conclusions

Finite element analysis (FEA) and hyperelastic formulations are fundamental tools in computational biomechanics research. Ensuring efficiency for biomedical applications is challenging given the nonlinearity of the material and boundary conditions, and the complexity of the geometry. This work presents FerriteHyperelastic, a novel open source Julia package for FEA of hyperelastic materials. FerriteHyperelastic supports 2D and

3D problems and supports compressible, near-incompressible, and purely incompressible hyperelasticity. A set of example benchmarks as well as biomedical applications are presented here. In addition, curve fitting, subject to stability analysis, is shared. FerriteHyperelastic showed a near perfect match to the C++ solver FEBio and presents with a similar computational cost. The presented package is suitable for a wide range of applications, such as biomechanics, soft electronics, and compliant mechanical structures.

Acknowledgments

GL and AA acknowledge grant PID2023-146347OA-I00 funded by $MICIU/AEI/10.13039/501100011033$ and $ERDF/EU$. GL also acknowledges grant RYC2022-036010-I funded by $MICIU/AEI/10.13039/501100011033$ and $ESF+$. Funding for open access charge: Universidade da Coruña/CISUG.

Author Contributions

A. Alibakhshi: Conceptualization, Data curation, Investigation, Visualization, Writing – original draft, Software. **K. Aghani** Conceptualization, Data curation, Investigation, Visualization, Writing – review and editing. **L. Saucedo-Mora**: Project administration, Supervision, Validation, Writing – review & editing. **G. Lorenzo**: Resources, Supervision, Validation, Writing – review & editing. **K.M. Moerman**: Software, Supervision, Validation, Writing – review & editing

Data available

The package and demos are available at https://github.com/Aminofa70/FerriteHyperelastic.jl.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declaration of Generative AI and AI-assisted technologies

A. Alibakhshi used Claude to assist with debugging, improving Julia code syntax, and reviewing some algebraic computations. All codes were carefully checked, edited, and validated by the author, who takes full responsibility for the accuracy and content of the code. AI has not been used for writing.

References

[1] Masoud Ahmadi, Andrew McBride, Paul Steinmann, and Prashant Saxena. Plane stress finite element modelling of arbitrary compressible hyperelastic materials: M. ahmadi et al. *Acta Mechanica*, pages 1–20, 2025.

[2] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.

[3] Steve A Maas, Benjamin J Ellis, Gerard A Ateshian, and Jeffrey A Weiss. Febio: finite elements for biomechanics. *Journal of Biomechanical Engineering*, 2012.

[4] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. S. Hale, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells. DOLFINx: The next generation FEniCS problem solving environment. preprint, 2023.

[5] Andreas Dutzler and Martin Leitner. Felupe: Finite element analysis for continuum mechanics of solid bodies. *Journal of Open Source Software*, 10(114):9160, 2025.

[6] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[7] K. Carlsson, F. Ekre, and Ferrite.jl contributors. Ferrite.jl (v1.1.0), 2025. Software.

[8] Santiago Badia and Francesc Verdugo. Gridap: An extensible finite element toolbox in julia. *Journal of Open Source Software*, 5(52):2520, 2020.

[9] Francesc Verdugo. GalerkinToolkit.jl: A finite element toolbox for julia. https://github.com/GalerkinToolkit/GalerkinToolkit.jl, 2024. Julia package, version 0.2.0, accessed 28 May 2026.

[10] Ferrite.jl Developers. Hyperelasticity. https://ferrite-fem.github.io/Ferrite.jl/stable/tutorials/hyperelasticity/, 2026. Ferrite.jl documentation, accessed 28 May 2026.

[11] Gridap Developers. Hyperelasticity. https://gridap.github.io/Tutorials/dev/pages/t005_hyperelasticity/, 2026. Gridap tutorials, accessed 28 May 2026.

[12] Juan C Simo and Robert L Taylor. Quasi-incompressible finite elasticity in principal stretches. continuum basis and numerical algorithms. *Computer methods in applied mechanics and engineering*, 85(3):273–310, 1991.

[13] Kevin M Moerman, Behrooz Fereidoonnezhad, and J Patrick McGarry. Novel hyperelastic models for large volumetric deformations. *International Journal of Solids and Structures*, 193:474–491, 2020.

[14] Elias Karabelas, Matthias AF Gsell, Gundolf Haase, Gernot Plank, and Christoph M Augustin. An accurate, robust, and efficient finite element framework with applications to anisotropic, nearly and fully incompressible elasticity. *Computer methods in applied mechanics and engineering*, 394:114887, 2022.

[15] Kristin M Myers, Christine P Hendon, Yu Gan, Wang Yao, Kyoko Yoshida, Michael Fernandez, Joy Vink, and Ronald J Wapner. A continuous fiber distribution material model for human cervical tissue. *Journal of biomechanics*, 48(9):1533–1540, 2015.

[16] Knut Andreas Meyer and contributors. FESolvers.jl: Solvers for ferrite.jl problems. https://github.com/KnutAM/FESolvers.jl, 2026. Julia package, accessed 28 May 2026.

[17] Kevin Mattheus Moerman, Mehmet Hakan Satman, Daniel VandenHeuvel, Chethana Rao, Juan Ignacio Polanco, and Simon Danish. Comodo.jl: A Julia Package for Computational Mechanics and Design, September 2025.

[18] Kevin Mattheus Moerman and Amin Alibakhshi. Febio.jl: A julia wrapper for the febio finite element solver, June 2026.

[19] Po-Sen Lin, Olivier Le Roux de Bretagne, Marzio Grasso, James Brighton, Chris StLeger-Harris, and Owen Carless. Comparative analysis of various hyperelastic models and element types for finite element analysis. *Designs*, 7(6):135, 2023.

[20] Salar Farahmand-Tabar and Kian Aghani. *Practical Programming of Finite Element Procedures for Solids and Structures with MATLAB®: From Elasticity to Plasticity*. Elsevier, 2023.

[21] Leslie RG Treloar. Stress-strain data for vulcanized rubber under various types of deformation. *Rubber Chemistry and Technology*, 17(4):813–825, 1944.

[22] Si Hang. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw*, 41(2):11, 2015.

[23] Simon Danisch and Julius Krumbiegel. Makie. jl: Flexible high-performance data visualization for julia. *Journal of Open Source Software*, 6(65):3349, 2021.

[24] Kevin M. Moerman, Ciaran K. Simms, and Thomas Nagel. Control of tension-compression asymmetry in Ogden hyperelasticity with application to soft tissue modelling. *Journal of the Mechanical Behavior of Biomedical Materials*, 56:218–228, March 2016.

[25] Guillermo Lorenzo, Thomas JR Hughes, Pablo Dominguez-Frojan, Alessandro Reali, and Hector Gomez. Computer simulations suggest that prostate enlargement due to benign prostatic hyperplasia mechanically impedes prostate cancer growth. *Proceedings of the National Academy of Sciences*, 116(4):1152–1161, 2019.

[26] Guillaume Lemaître, Robert Martí, Jordi Freixenet, Joan C Vilanova, Paul M Walker, and Fabrice Meriaudeau. Computer-aided detection and diagnosis for prostate cancer based on mono and multi-parametric mri: a review. *Computers in biology and medicine*, 60:8–31, 2015.

[27] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, et al. 3d slicer as an image computing platform for the quantitative imaging network. *Magnetic resonance imaging*, 30(9):1323–1341, 2012.

[28] Bruno Lévy and Alain Filbois. Geogram: A library for geometric algorithms, 2015. Software library.

[29] Kevin Mattheus Moerman. Geogram.jl: A basic julia wrapper for the geogram library. https://github.com/COMODO-research/Geogram.jl, June 2026.

[30] Kevin Mattheus Moerman. Mammo.jl a julia package for breast tissue biomechancs and mammography simulation. https://github.com/COMODO-research/Mammo.jl, June 2026.