Elsevier Editorial System(tm) for Computer Methods in Applied Mechanics and Engineering
Manuscript Draft

Manuscript Number:

Title: A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid

Article Type: Research Paper

Keywords: adaptive finite element method; geometric multigrid; finite-strain; matrix-free; hyperelasticity

Corresponding Author: Dr. Denis Davydov, Ph.D.

Corresponding Author's Institution: University Erlangen-Nurnberg

First Author: Denis Davydov, Ph.D.

Order of Authors: Denis Davydov, Ph.D.; Jean-Paul Pelteret; Daniel Arndt; Paul Steinmann

Abstract: The performance of finite element solvers on modern computer architectures is typically memory bound. The main cause for this is that loading matrix elements from RAM into CPU cache is significantly slower than performing the arithmetic operations when solving the problem. In order to improve the performance of iterative solvers within the high-performance computing context, so-called matrix-free methods are widely adopted in the fluid mechanics community, where matrix-vector products are computed on-the-fly.

In this work, we extend the application of matrix-free approaches to problems in solid mechanics and investigate different implementations of the finite-strain hyperelastic finite element tangent operator numerically. In order to improve the convergence behavior of iterative solvers, we also propose a method by which to construct level tangent operators and employ them to define a geometric multigrid preconditioner. Our implementation employs MPI and Intel Threading Building Blocks parallelization, and SIMD vectorization. The performance of the matrix-free operator and the geometric multigrid preconditioner is compared to the matrix-based implementation with an algebraic multigrid preconditioner for a representative numerical example of a heterogeneous hyperelastic material in two and three dimensions.

Suggested Reviewers: Lukasz Kaczmarczyk Dr
Senior Lecturer, University of Glasgow
Lukasz.Kaczmarczyk@glasgow.ac.uk
knows well FEM and also worked on geometric multigrid methods for solid mechanics

Katharina Kormann Dr
Max-Planck-Institut für Plasmaphysik
katharina.kormann@ipp.mpg.de
expert in matrix-free methods for FE operators

Tzanio Kolev Dr

LLNL
tzanio@llnl.gov
computational mathematician, expert in matrix-free methods

Ulrich Rüde Prof. Dr
Prof, Department of Computer Science, FAU
ulrich.ruede@fau.de
mathematician and computational scientist, solves extremely large scale problems (10^9) with matrix-free methods

Jed Brown Prof. Dr
Assistant Professor, CU Boulder
jed.brown@colorado.edu
computational mathematician, developer of PETSc, also works on matrix-free methods

Veselin Dobrev Dr
LLNL
dobrev@llnl.gov
computational mathematician, expert in FEM and matrix-free

Opposed Reviewers:

Dear Prof. Papadrakakis,

Please find attached the manuscript: "A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid" authored by myself, Jean-Paul Pelteret, Daniel Arndt and Paul Steinmann that we would like to publish in "Computer Methods in Applied Mechanics and Engineering". It has not been published or submitted elsewhere.

Matrix-free methods are widely adopted in the fluid mechanics community, where matrix-vector products are computed on-the-fly.
However they have not yet get enough attention for solid mechanics applications.
In the quest for high performance numerical solution of PDEs in solid mechanics, in this contribution we extend the application of matrix-free approaches to problems in solid mechanics and investigate different implementations of the finite-strain hyperelastic finite element tangent operator numerically.
In order to improve the convergence behavior of iterative solvers, we also propose a method by which to construct level tangent operators and employ them to define a geometric multigrid preconditioner.
The performance of the matrix-free operator and the geometric multigrid preconditioner is compared to the matrix-based implementation with an algebraic multigrid preconditioner for a representative numerical example of a heterogeneous hyperelastic material in two and three dimensions.

With many thanks for your efforts in advance,

Kind Regards,
Denis Davydov.

Highlights:

- apply matrix-free approaches to finite-strain problems in solid mechanics
- propose and investigate different implementations of the finite-strain hyperelastic finite element tangent operator numerically
- propose a method by which to construct level tangent operators and employ them to define a geometric multigrid preconditioner
- compare the performance of the matrix-free operator and the geometric multigrid preconditioner to the matrix-based implementation with an algebraic multigrid preconditioner for a representative numerical example of a heterogeneous hyperelastic material in two and three dimensions.

A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid

Denis Davydov[a,*], Jean-Paul Pelteret[a], Daniel Arndt[b], Paul Steinmann[a,c]

[a] *Chair of Applied Mechanics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Egerlandstr. 5, 91058 Erlangen, Germany*
[b] *Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany*
[c] *Glasgow Computational Engineering Center (GCEC), University of Glasgow, G12 8QQ Glasgow, United Kingdom*

Abstract

The performance of finite element solvers on modern computer architectures is typically memory bound. The main cause for this is that loading matrix elements from RAM into CPU cache is significantly slower than performing the arithmetic operations when solving the problem. In order to improve the performance of iterative solvers within the high-performance computing context, so-called matrix-free methods are widely adopted in the fluid mechanics community, where matrix-vector products are computed on-the-fly.

In this work, we extend the application of matrix-free approaches to problems in solid mechanics and investigate different implementations of the finite-strain hyperelastic finite element tangent operator numerically. In order to improve the convergence behavior of iterative solvers, we also propose a method by which to construct level tangent operators and employ them to define a geometric multigrid preconditioner. Our implementation employs MPI and Intel Threading Building Blocks parallelization, and SIMD vectorization. The performance of the matrix-free operator and the geometric multigrid preconditioner is compared to the matrix-based implementation with an algebraic multigrid preconditioner for a representative numerical example of a heterogeneous hyperelastic material in two

*Corresponding author.
Email addresses: denis.davydov@fau.de (Denis Davydov), jean-paul.pelteret@fau.de (Jean-Paul Pelteret), daniel.arndt@iwr.uni-heidelberg.de (Daniel Arndt), paul.steinmann@fau.de (Paul Steinmann)

and three dimensions.

Keywords: adaptive finite element method, geometric multigrid, finite-strain, matrix-free, hyperelasticity

1. Introduction

The performance of finite element solvers on modern computer architectures is typically memory bound. The main cause for this is that loading pre-computed matrix elements from RAM into CPU cache is significantly slower than performing the arithmetic operations when solving the problem. In order to improve the performance of iterative solvers, so-called matrix-free methods, where matrix-vector products are formed on-the-fly [1–5], are widely adopted in the fluid mechanics community. Such methods can also be efficiently implemented on GPUs [6, 7].

To the best of our knowledge, matrix-free methods are not widely adopted within the solid mechanics community. We are only aware of a single paper currently in preparation that deals with small strain linear elasticity [8] and a recent masters thesis produced by a student of the first two of the authors [9]. One possible reason is that the tangent operators[1] (stemming from linearization of the nonlinear balance equations) are more elaborate as compared to the operators used often in both linear and non-linear fluid mechanics, and it is not obvious whether matrix-free methods can be advantageous in this case. Another reason could be the frequent use of lower order elements in solid mechanics due to lacking smoothness of the underlying solution. The finite element method (FEM) with linear or quadratic elements (that reduce the potential for locking), or mixed or enhanced formulations (that avoid locking) are often adopted in the solid mechanics community, whereas higher order elements are rarely employed. This can potentially reduce the advantage of matrix-free methods, that are generally more competitive for higher order elements. However, we note that, even with linear FEM, large scale computations with 10^{12} unknowns inevitably require a matrix-free approach as there is simply not enough memory to store the sparse tangent matrix

[1]Here and below 'operator' denotes a finite-dimensional linear operator defined on the vector space $\mathbb{R}^N$, where N is the number of unknown degrees of freedom in the FE discretization. Although any linear mapping on a finite-dimensional space is representable by a matrix, we adopt the 'operator' term to avoid confusion between its matrix-free and matrix-based numerical implementation.

[5]. Therefore, we believe that for large scale computations in solid mechanics it is crucial to consider matrix-free approaches and develop efficient multigrid solvers.

In this work, we investigate different implementations of the finite-strain hyperelastic tangent operator with respect to their computational cost. In order to improve the convergence of iterative solvers, we also propose a method by which to construct level matrix-free tangent operators and employ them to define a geometric multigrid preconditioner. Our implementation employs MPI and Intel Threading Building Blocks parallelization, together with SIMD vectorization.

The paper is organized as follows: In Section 2 we briefly introduce the partial differential equations used in finite-strain solid mechanics. Sections 3 and 4 cover the finite element discretization and the different matrix-free operator evaluation approaches. In Section 5 we propose a geometric multigrid preconditioner suitable for finite-strain hyperelasticity. The performance of the matrix-free operator and the geometric multigrid preconditioner is compared to the matrix-based implementation with an algebraic multigrid preconditioner for a representative numerical example of a heterogeneous hyperelastic material simulated in two and three dimensions in Section 6. The results are summarized in Section 7.

2. Theoretical background

2.1. Weak form

The deformation of a body $\mathcal{B}$ from the referential configuration $\mathcal{B}_0$ to the spatial configuration $\mathcal{B}_t$ at time t is defined via the deformation map $\boldsymbol{\varphi}_t : \mathcal{B}_0 \to \mathcal{B}_t$, which places material points of $\mathcal{B}$ into the Euclidean space $\mathbb{E}^3$. The spatial location of a material particle $\boldsymbol{X}$ is given by $\boldsymbol{x} = \boldsymbol{\varphi}_t(\boldsymbol{X})$. The displacement field reads $\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{X}$.

The tangent map is linear such that $\mathrm{d}\boldsymbol{x} = \boldsymbol{F} \cdot \mathrm{d}\boldsymbol{X}$, where $\boldsymbol{F} := \mathrm{Grad}\,\boldsymbol{\varphi} \equiv \boldsymbol{I} + \mathrm{Grad}\,\boldsymbol{u}$ is called the deformation gradient and $\boldsymbol{I}$ is the second-order unit tensor. The mapping has to be one-to-one and must exclude self-penetration. Consequently, the Jacobian $J = \det \boldsymbol{F} > 0$ has to be positive. We shall denote the push-forward transformation of rank-2 and rank-4 tensors $\mathbf{A}$ and $\mathcal{A}$ by a linear transformation χ defined as

$$\chi(\mathbf{A})_{ij} = F_{iA} A_{AB} F_{jB} \tag{1}$$

$$\chi(\mathcal{A})_{ijkl} = F_{iA} F_{jB} \mathcal{A}_{ABCD} F_{kC} F_{lD}. \tag{2}$$

Note that $\mathrm{d}\boldsymbol{x} = \chi(\mathrm{d}\boldsymbol{X})$.

In what follows, we parameterize the material behavior using the right Cauchy-Green tensor $\boldsymbol{C} := \boldsymbol{F}^T \cdot \boldsymbol{F}$. We will also use the Green-Lagrange strain tensor $\boldsymbol{E} := \frac{1}{2}\left[\boldsymbol{C} - \boldsymbol{I}\right]$ and the left Cauchy-Green tensor $\boldsymbol{b} := \boldsymbol{F} \cdot \boldsymbol{F}^T$. Clearly $J = \sqrt{\det \boldsymbol{C}}$ and $\partial(\bullet)/\partial \boldsymbol{E} = 2\,\partial(\bullet)/\partial \boldsymbol{C}$. For conservative systems without body forces, the total potential energy functional $\mathcal{E}$ is introduced as

$$\mathcal{E} = \int_{\mathcal{B}_0} \psi(\boldsymbol{F})\,\mathrm{d}V - \int_{\partial\mathcal{B}_0^N} \overline{\boldsymbol{T}} \cdot \boldsymbol{u}\,\mathrm{d}S\,, \tag{3}$$

where $\overline{\boldsymbol{T}}$ is the prescribed loading at the Neumann part of the boundary $\partial\mathcal{B}_0^N$ in the referential configuration, ψ denotes the strain-energy per unit reference volume and $\boldsymbol{u}$ should satisfy the prescribed Dirichlet boundary conditions $\boldsymbol{u} = \overline{\boldsymbol{u}}$ on $\partial\mathcal{B}_0^D := \partial\mathcal{B}_0 \setminus \partial\mathcal{B}_0^N \neq \emptyset$.

The principle of stationary potential energy at equilibrium requires that the directional derivative with respect to the displacement

$$\mathrm{D}_{\delta\boldsymbol{u}}\mathcal{E} := \frac{\mathrm{d}}{\mathrm{d}\epsilon}\mathcal{E}(\boldsymbol{u} + \epsilon\delta\boldsymbol{u})\Big|_{\epsilon=0} = 0 \qquad \forall\delta\boldsymbol{u}\,. \tag{4}$$

vanishes for all directions $\delta\boldsymbol{u}$ which satisfy homogeneous Dirichlet boundary conditions. This leads to the following scalar-valued non-linear equation

$$F(\boldsymbol{u},\delta\boldsymbol{u}) = \int_{\mathcal{B}_0} \boldsymbol{P} : \mathrm{Grad}\,\delta\boldsymbol{u}\,\mathrm{d}V - \int_{\partial\mathcal{B}_0^N} \overline{\boldsymbol{T}} \cdot \delta\boldsymbol{u}\,\mathrm{d}S = 0\,, \tag{5}$$

where $\boldsymbol{P} := \partial\psi/\partial\boldsymbol{F}$ is the Piola stress tensor. The double contraction in the first term can be re-written in terms of the symmetric Kirchhoff stress tensor $\boldsymbol{\tau} := \boldsymbol{P}\cdot\boldsymbol{F}^T$ as

$$\boldsymbol{P} : \mathrm{Grad}\,\delta\boldsymbol{u} = \left[\boldsymbol{\tau}\cdot\boldsymbol{F}^{-T}\right] : \mathrm{Grad}\,\delta\boldsymbol{u} = \boldsymbol{\tau} : \left[\mathrm{Grad}\,\delta\boldsymbol{u}\cdot\boldsymbol{F}^{-1}\right] = \boldsymbol{\tau} : \mathrm{grad}\,\delta\boldsymbol{u}\,, \tag{6}$$

and therefore

$$F(\boldsymbol{u},\delta\boldsymbol{u}) = \int_{\mathcal{B}_0} \boldsymbol{\tau} : \mathrm{grad}^s\,\delta\boldsymbol{u}\,\mathrm{d}V - \int_{\partial\mathcal{B}_0^N} \overline{\boldsymbol{T}} \cdot \delta\boldsymbol{u}\,\mathrm{d}S = 0\,, \tag{7}$$

where $\mathrm{grad}^s\,\delta\boldsymbol{u} := \frac{1}{2}\left[\mathrm{grad}\,\delta\boldsymbol{u} + (\mathrm{grad}\,\delta\boldsymbol{u})^T\right]$ is involved due to the symmetry of $\boldsymbol{\tau}$. Note that $\boldsymbol{\tau}$ is the push-forward transformation of the Piola-Kirchhoff stress $\boldsymbol{S} := \partial\psi/\partial\boldsymbol{E} \equiv 2\partial\psi/\partial\boldsymbol{C}$, that is $\boldsymbol{\tau} = \chi(\boldsymbol{S}) = \boldsymbol{F}\cdot\boldsymbol{S}\cdot\boldsymbol{F}^T$.

2.2. Linearization

In order to solve (7) using Newton's method, a first order approximation around a given solution field $\overline{\boldsymbol{u}}$ is required such that

$$F(\overline{\boldsymbol{u}} + \Delta\boldsymbol{u}, \delta\boldsymbol{u}) \approx F(\overline{\boldsymbol{u}}, \delta\boldsymbol{u}) + \mathrm{D}_{\Delta\boldsymbol{u}} F(\overline{\boldsymbol{u}}, \delta\boldsymbol{u}), \tag{8}$$

where $\mathrm{D}_{\Delta\boldsymbol{u}}(\bullet)$ denotes the directional derivative in the direction $\Delta\boldsymbol{u}$. For conservative traction boundary conditions, the directional derivative is given by

$$\begin{aligned} \mathrm{D}_{\Delta\boldsymbol{u}} F(\overline{\boldsymbol{u}}, \delta\boldsymbol{u}) = &\int_{\mathcal{B}_0} \mathrm{D}_{\Delta\boldsymbol{u}} \left(\boldsymbol{F} \cdot \boldsymbol{S} \cdot \boldsymbol{F}^T\right) : \overline{\mathrm{grad}^s}\, \delta\boldsymbol{u}\, \mathrm{d}V \\ &+ \int_{\mathcal{B}_0} \overline{\boldsymbol{\tau}} : \left[\mathrm{Grad}\, \delta\boldsymbol{u} \cdot \mathrm{D}_{\Delta\boldsymbol{u}} \boldsymbol{F}^{-1}\right] \mathrm{d}V. \end{aligned} \tag{9}$$

Following [10], this can be simplified to[2]

$$\begin{aligned} \mathrm{D}_{\Delta\boldsymbol{u}} F(\overline{\boldsymbol{u}}, \delta\boldsymbol{u}) = &\int_{\mathcal{B}_0} \overline{\mathrm{grad}^s}\Delta\boldsymbol{u} : J\boldsymbol{\mathcal{C}} : \overline{\mathrm{grad}^s}\, \delta\boldsymbol{u}\, \mathrm{d}V \\ &+ \int_{\mathcal{B}_0} \overline{\mathrm{grad}}\delta\boldsymbol{u} : \left[\overline{\mathrm{grad}}\Delta\boldsymbol{u} \cdot \overline{\boldsymbol{\tau}}\right] \mathrm{d}V. \end{aligned} \tag{10}$$

Here $\overline{(\bullet)}$ is used to denote quantities evaluated using the displacement field $\overline{\boldsymbol{u}}$, and the fourth-order material part of the spatial tangent stiffness tensor is the push forward of the material part of the referential tangent stiffness tensor $J\boldsymbol{\mathcal{C}} = \chi\left(4\frac{d^2\psi(\mathbf{C})}{d\mathbf{C}\otimes d\mathbf{C}}\right)$. The first term in (10) is related to the linearization of the Piola-Kirchhoff stress $\boldsymbol{S}$ and is therefore called material part of the directional derivative. The second term in (10) is called the geometric part of the directional derivative since it originates from the linearization of $\boldsymbol{F}$, $\boldsymbol{F}^T$ and $\boldsymbol{F}^{-1}$.

2.3. Constitutive modelling

For the current study, we use the compressible Neo-Hookean model

$$\psi\left(\mathbf{C}\right) = \frac{\mu}{2}\left[\mathrm{tr}\,\mathbf{C} - \mathrm{tr}\,\mathbf{I} - 2\ln\left(J\right)\right] + \lambda \ln^2\left(J\right) \tag{11}$$

[2]To that end, $\mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{E} = \frac{1}{2}\left[\mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{F}^T \cdot \boldsymbol{F} + \boldsymbol{F}^T \cdot \mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{F}\right]$, $\mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{F} = \mathrm{Grad}\Delta\boldsymbol{u}$ and $\mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{F}^{-1} = -\boldsymbol{F}^{-1} \cdot \mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{F} \cdot \boldsymbol{F}^{-1}$ are employed together with $\mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{S} = 2\partial\boldsymbol{S}/\partial\boldsymbol{C} : \mathrm{D}_{\Delta\boldsymbol{u}}\boldsymbol{E}$.

where μ and λ denote the shear modulus and Lamé parameter respectively. It can be shown that the first and second derivatives of the strain energy function are given by

$$\frac{d\psi(\mathbf{C})}{d\mathbf{C}} = \frac{\mu}{2}\mathbf{I} - \frac{1}{2}\left[\mu - 2\lambda\ln(J)\right]\mathbf{C}^{-1} \tag{12}$$

$$\frac{d^2\psi(\mathbf{C})}{d\mathbf{C}\otimes d\mathbf{C}} = \frac{1}{2}\left[\mu - 2\lambda\ln(J)\right]\left[-\frac{d\mathbf{C}^{-1}}{d\mathbf{C}}\right] + \frac{\lambda}{2}\mathbf{C}^{-1}\otimes\mathbf{C}^{-1} \tag{13}$$

The Kirchhoff stress and its associated fourth-order material part of the spatial tangent tensor are

$$\boldsymbol{\tau} \equiv J\boldsymbol{\sigma} = \chi\left(2\frac{d\psi(\mathbf{C})}{d\mathbf{C}}\right) = \mu\mathbf{b} - \left[\mu - 2\lambda\ln(J)\right]\mathbf{I} \tag{14}$$

and

$$J\mathcal{C} = \chi\left(4\frac{d^2\psi(\mathbf{C})}{d\mathbf{C}\otimes d\mathbf{C}}\right) = 2\left[\mu - 2\lambda\ln(J)\right]\boldsymbol{\mathcal{S}} + 2\lambda\mathbf{I}\otimes\mathbf{I} \tag{15}$$

where $\boldsymbol{\mathcal{S}}$ is the fourth-order symmetric identity tensor with $\mathcal{S}_{ijkl} = \frac{1}{2}\left[\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right]$. The action that $J\mathcal{C}$ performs when contracted with an arbitrary rank-2 symmetric tensor is therefore

$$J\mathcal{C} : (\bullet) = 2\left[\mu - 2\lambda\ln(J)\right](\bullet) + 2\lambda\,\mathrm{tr}(\bullet)\,\mathbf{I}. \tag{16}$$

3. Finite Element discretization

We now introduce a FE triangulation $\mathscr{B}_0^h$ of $\mathscr{B}_0$ and the associated FE space of continuous piecewise elements with polynomial space of fixed degree. The displacement fields are given in a vector space spanned by standard vector–valued FE basis functions $\boldsymbol{N}_i(\boldsymbol{x})$ (e.g. polynomials with local support on a patch of elements):

$$\boldsymbol{u}^h =: \sum_{i\in\mathcal{I}} u_i\boldsymbol{N}_i(\boldsymbol{X}) \qquad \delta\boldsymbol{u}^h =: \sum_{i\in\mathcal{I}} \delta u_i\boldsymbol{N}_i(\boldsymbol{X}), \tag{17}$$

where the superscript h denotes that this representation is related to the FE mesh with size function $h(\boldsymbol{X})$ and $\mathcal{I}$ is the set of unknown degrees of freedom (DoF).

The Newton-Raphson solution approach is adopted for the nonlinear problem considered here. Given the current trial solution field $\overline{\boldsymbol{u}^h}$, the correction $\Delta\boldsymbol{u}^h$ field is obtained as the solution to the following system of equations (see (7) and (10)):

$$\sum_{j\in\mathcal{I}} A_{ij}\Delta u_j = -F_i \tag{18}$$

$$A_{ij} \equiv a(\boldsymbol{N}_i,\boldsymbol{N}_j) = \int_{\mathcal{B}_0} \underbrace{\overline{\mathrm{grad}^s}\boldsymbol{N}_i : J\boldsymbol{C} : \overline{\mathrm{grad}^s}\,\boldsymbol{N}_j + \overline{\mathrm{grad}}\boldsymbol{N}_i : \left[\overline{\mathrm{grad}}\boldsymbol{N}_j \cdot \overline{\boldsymbol{\tau}}\right]}_{\tilde{a}(\boldsymbol{N}_i,\boldsymbol{N}_j)} \mathrm{d}V \tag{19}$$

$$F_i = \int_{\mathcal{B}_0} \overline{\boldsymbol{\tau}} : \mathrm{grad}^s\,\boldsymbol{N}_i\,\mathrm{d}V - \int_{\partial\mathcal{B}_0^N} \overline{\boldsymbol{T}}\cdot\boldsymbol{N}_i\,\mathrm{d}S. \tag{20}$$

Here, $\boldsymbol{A}$ is the discrete tangent operator, $\boldsymbol{F}$ is the discrete gradient of the potential energy and $\tilde{a}(\boldsymbol{N}_i,\boldsymbol{N}_j)$ is a representation for the integrand in the bilinear form $a(\boldsymbol{N}_i,\boldsymbol{N}_j)$ for the trial solution field $\overline{\boldsymbol{u}^h}$. Note that $\overline{\mathrm{grad}^s}\boldsymbol{N}_i$ and $\overline{\mathrm{grad}}\boldsymbol{N}_i$ are gradients of shape functions in the spatial configuration, whereas the integration is done in the referential configuration.

4. Matrix-free operator evaluation

Classically, in order to solve (18) the matrix $\boldsymbol{A}$ and the residual force vector $\boldsymbol{F}$ corresponding to the discretization are assembled (that is, the non-zero matrix elements A_{ij} are individually computed and stored) and a direct or iterative solver is used to solve the linear system. In this case the the matrix-vector product $\boldsymbol{Ax}$ results from the product of the individual matrix elements with the elements in the vector, and is usually implemented with the aid of specialized linear algebra packages. On modern computer architectures however, it transpires that loading sparse matrix data into the CPU registers is significantly slower than performing the arithmetic operations when solving. For this reason, recent implementations often focus on so-called matrix-free approaches where the matrix is not assembled but rather the result of its operation on a vector is directly calculated. The idea is to perform the operations within the solver on-the-fly rather than loading matrix elements from memory. For iterative solvers, this is possible since it is sufficient to compute matrix-vector products alone. The matrix-vector product $\boldsymbol{Ax}$ (i.e. the action of operator $\boldsymbol{A}$ on a vector $\boldsymbol{x}$) can be expressed by

$$\begin{aligned}(\boldsymbol{A}\boldsymbol{x})_i &= \sum_j a(\boldsymbol{N}_i, \boldsymbol{N}_j) x_j \\ &\approx \sum_K \sum_q \sum_j \tilde{a}(\boldsymbol{N}_i, \boldsymbol{N}_j)(\boldsymbol{\xi}_q) x_j w_q J_q^K\end{aligned} \tag{21}$$

where $\tilde{a}(\boldsymbol{N}_i, \boldsymbol{N}_j)(\boldsymbol{\xi}_q)$ is a representation for the evaluation of the bilinear form in one quadrature point $\boldsymbol{\xi}_q$, J_q^K is the Jacobian of the mapping from the isoparametric element to the element K and w_q is the quadrature weight. For the sake of demonstration, let us assume that $\boldsymbol{A}$ represents a mass operator with scalar valued shape functions $\{N_i(\boldsymbol{x})\}$. In this case, it holds that

$$\tilde{a}(N_i, N_j)(\boldsymbol{\xi}_q) = N_i(\boldsymbol{\xi}_q) N_j(\boldsymbol{\xi}_q).$$

Further assuming that the number of quadrature points n in one-dimension matches the degree of the ansatz space plus one, the evaluation of all of the shape functions (n^d) in all quadrature points (n^d) has complexity $O(n^{2d})$ in d space dimensions. For the total matrix-vector product we get $O(2n^{2d})$, and this is the same as for a regular matrix-vector product where the entries are already precomputed. Assuming tensor product ansatz spaces and a tensor product quadrature rule, evaluation of this operations can be performed more efficiently using a technique called “sum factorization”.

Let us illustrate this in 2D. Tensor-product quadrature points can be expressed as a combination of one dimensional quadrature points

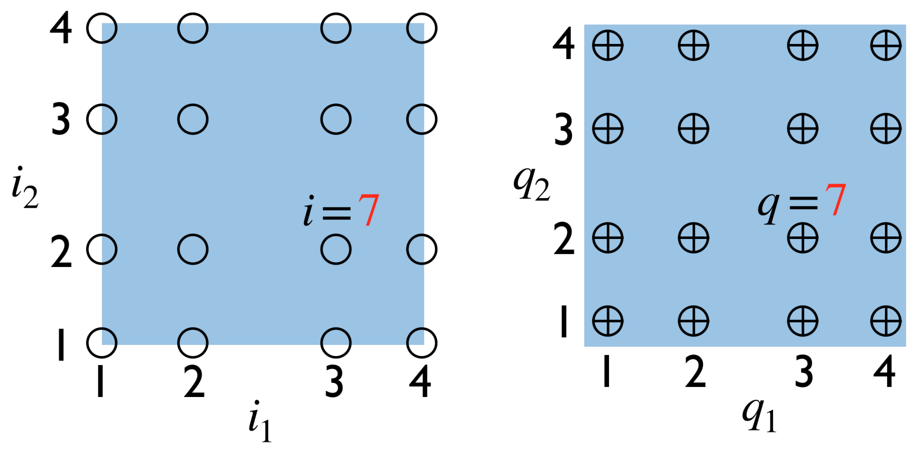
$$\boldsymbol{\xi}_q = (\widetilde{\xi}_{q_1}, \widetilde{\xi}_{q_2}), \tag{22}$$

where for each quadrature points q we can associate a multi-index (q_1, q_2). Weights associated with the quadrature formula w_q can be expressed as a product of one-dimensional weights

$$w_q = \widetilde{w}_{q_1} \widetilde{w}_{q_2} \,. \tag{23}$$

Shape functions can also be expressed as product of one dimensional basis functions

$$N_i(\boldsymbol{\xi}_q) = \widetilde{N}_{i_1}(\widetilde{\xi}_{q_1}) \widetilde{N}_{i_2}(\widetilde{\xi}_{q_2}) \tag{24}$$



(a) Nodes of cubic Lagrange element (b) Gaussian quadrature points

Figure 1: Illustration of tensorial indexing for nodes and quadrature points. In this sketch a node/quadrature point with the number 7 can be indexed with a multi-index $\{3,2\}$.

where for each DoF index i we can associate a multi-index (i_1, i_2). See Figure 1 for an illustration. Therefore, the result of the application of the scalar-valued mass operator, represented as a vector y_i, on a single element K can be written as

$$
\begin{aligned}
y_i \equiv Y_{i_1\, i_2} &= \sum_{q_1}\sum_{q_2}\sum_{j_1}\sum_{j_2} \widetilde{N}_{i_1}(\widetilde{\xi}_{q_1})\widetilde{N}_{i_2}(\widetilde{\xi}_{q_2})\widetilde{N}_{j_1}(\widetilde{\xi}_{q_1})\widetilde{N}_{j_2}(\widetilde{\xi}_{q_2})X_{j_1\, j_2}\widetilde{w}_{q_1}\widetilde{w}_{q_2}J^K_{q_1\, q_2} \\
&= \sum_{q_1}\widetilde{N}_{i_1}(\widetilde{\xi}_{q_1})\widetilde{w}_{q_1}\sum_{q_2}\widetilde{N}_{i_2}(\widetilde{\xi}_{q_2})\widetilde{w}_{q_2}J^K_{q_1\, q_2}\left[\sum_{j_1}\widetilde{N}_{j_1}(\widetilde{\xi}_{q_1})\sum_{j_2}\widetilde{N}_{j_2}(\widetilde{\xi}_{q_2})X_{j_1\, j_2}\right] \forall i_1\, i_2
\end{aligned}
$$

Note that $x_j \equiv X_{j_1\, j_2}$; the former accesses the element source vector using a linear index j, whereas the latter uses the associated multi-index accesses $\{j_1\, j_2\}$. These four loops all have the same structure. We either keep the shape function fixed and iterate over the quadrature points in one spatial direction (in the first two sums) or keep the quadrature point fixed and iterate over all the shape functions in one spatial direction (in the last two sums). Since each of these $2 \times d$ loops has a complexity of n and we perform them for n^d values in quadrature points or values in degrees of freedoms simultaneously, this results in an algorithm with an arithmetic complexity of $O(2dn^{d+1})$. Hence, we can expect that such a matrix-free approach is faster than a regular matrix-based approach, especially for the 3D case, and where the discretization and evaluation employs high polynomial degree basis functions with the corresponding quadrature rule. More information on matrix-free techniques can be found in [1, 11].

We briefly note that the SIMD vectorization is applied at the finite element level, i.e. the matrix-free operator is applied simultaneously on several elements (called "blocks") whereas the shared memory parallelization using Intel Threading Building Blocks is adopted at the element blocks.

Given that the matrix-free operators are applied on-the-fly, it is crucial to find a good balance between pre-calculation of certain values (caching) and performance of the algorithm. Below, we propose three algorithms to implement the tangent operator of finite-strain elasticity (see (19)) using the matrix-free approach. In order to obtain gradients with respect to the spatial configuration we used the *MappingQEulerian* class from the `deal.II` [12] library. In addition to the standard mapping from the reference (isoparametric) element to the element in real space, this class computes the mapping to the spatial configuration given a displacement field. Due to the specifics of matrix-free operator evaluation implemented in `deal.II`, the integration is eventually performed over the spatial configuration. Therefore, we have to additionally divide by the Jacobian J of the deformation map at a given quadrature point to express the integral in the referential configuration.

Algorithm 1 caches a single scalar which involves the logarithm of the Jacobian – a computationally demanding operation compared to additions and multiplications. As evident from Section 2, we need to re-evaluate the Kirchhoff stress at each quadrature point in order to apply the tangent operator. This in turn requires evaluating the deformation gradient $\boldsymbol{F}$ and therefore gradients with respect to the referential configuration.

In order to avoid recalculation of the Kirchhoff stress, Algorithm 2 caches its value for each element and quadrature point. In this case, we also avoid the need to evaluate gradients of displacement field with respect to the referential configuration. This algorithm also utilizes the chosen constitutive relationship in that the operation of $J\boldsymbol{\mathcal{C}}$ remains directly expressed using (16). Therefore, the action of the material part of the fourth-order spatial tangent stiffness tensor $J\boldsymbol{\mathcal{C}}$ on the second-order symmetric tensor $\overline{\text{grad}^s \boldsymbol{x}}$ is cheap to evaluate. To that end, we cache two scalars that depend on the Jacobian J.

Finally, Algorithm 3 represents the most general case which does not assume any form of the material part of the fourth-order spatial tangent stiffness tensor. In this case, we have to cache the fourth-order symmetric tensor $\boldsymbol{\mathcal{C}}$ and the Kirchhoff stress $\boldsymbol{\tau}$ for each element and quadrature point, and perform the double contraction with the second-order symmetric tensor $\overline{\text{grad}^s \boldsymbol{x}}$ on-the-fly.

Algorithm 1: Matrix-free tangent operator: cache scalar quantities only

```
Given  : Source FE vector $\boldsymbol{x}$, current FE solution $\overline{\boldsymbol{u}^h}$, cached
         $c_1 := \mu - 2\lambda \log(J)$ for each cell and quadrature point
Return: action of the FE tangent operator (19) on $\boldsymbol{x}$
1  foreach element $K \in \Omega^h$ do
2      evaluate the following tensors at each quadrature point using sum
         factorization ;
3      Grad $\overline{\boldsymbol{u}^h}$ ;                                   // 2nd order
4      $\boldsymbol{g}_s := \overline{\text{grad}^s}\boldsymbol{x}$ ;                       // 2nd order symmetric
5      $\boldsymbol{g} := \overline{\text{grad}}\boldsymbol{x}$ ;                                  // 2nd order
6      foreach quadrature point $q$ on $K$ do
7          evaluate $\boldsymbol{F} = \boldsymbol{I} + \text{Grad}\,\overline{\boldsymbol{u}^h}$ ;                 // 2nd order
8          evaluate $J = \det(\boldsymbol{F})$ ;                           // scalar
9          evaluate $\boldsymbol{b} = \boldsymbol{F} \cdot \boldsymbol{F}^T$ ;              // 2nd order symmetric
10         evaluate $\boldsymbol{\tau} = \mu \boldsymbol{b} - c_1 \boldsymbol{I}$ ;             // 2nd order symmetric
11         evaluate $\mathcal{G}\boldsymbol{g} := \boldsymbol{g} \cdot \boldsymbol{\tau}/J$ ;                    // 2nd order
12         queue $\mathcal{G}\boldsymbol{g}$ for contraction $\overline{\text{grad}}\boldsymbol{N}_i : \mathcal{G}\boldsymbol{g}$;
13         evaluate $\boldsymbol{C}\boldsymbol{g}_s := [2c_1\boldsymbol{g}_s + 2\lambda\,\text{tr}(\boldsymbol{g}_s)\boldsymbol{I}]\,/J$ ;         // 2nd order
             symmetric
14         queue $\boldsymbol{C}\boldsymbol{g}_s$ for contraction $\overline{\text{grad}^s}\boldsymbol{N}_i : \boldsymbol{C}\boldsymbol{g}_s$ ;
15     end
16     evaluate queued contractions using sum factorization and distribute
         results to the destination vector
17 end
```

5. Geometric multigrid preconditioning

Efficient and scalable matrix-vector products are not enough to obtain a method suitable for large computations. In particular, linear scaling with respect to the number N of unknown DoFs is desirable. The efficiency of standard iterative solvers and preconditioners deteriorates for large systems of linear equations due to increasing iteration counts under mesh refinement. In contrast to other preconditioners, geometric multigrid (GMG) preconditioners [2, 13–15] can be proven to result in iteration counts that are independent of the number of mesh refinements by smoothing the residual on a hierarchy of meshes. For further information about geometric multigrid methods we refer the reader to [14, 16, 17].

Algorithm 2: Matrix-free tangent operator: cache second-order Kirchhoff stress $\boldsymbol{\tau}$ and thereby avoid the need to evaluate referential quantities like $\boldsymbol{F}$ at when executed.

```
    Given : Source FE vector $\boldsymbol{x}$, cached $c_1 := 2\left[\mu - 2\lambda \log(J)\right]/J$,
            $c_2 := 2\lambda/J$ and $\boldsymbol{\tau}/J$ for each cell and quadrature point, evaluated
            based on $\overline{\boldsymbol{u}^h}$
    Return: action of the FE tangent operator (19) on $\boldsymbol{x}$
 1  foreach element $K \in \Omega^h$ do
 2      evaluate the following tensors at each quadrature point using sum
          factorization ;
 3      $\boldsymbol{g}_s := \overline{\mathrm{grad}^s} \boldsymbol{x}$ ;                      // 2nd order symmetric
 4      $\boldsymbol{g} := \overline{\mathrm{grad}} \boldsymbol{x}$ ;                              // 2nd order
 5      foreach quadrature point q on K do
 6          evaluate $\mathcal{G}\boldsymbol{g} := \boldsymbol{g} \cdot [\boldsymbol{\tau}/J]$ ;                  // 2nd order
 7          queue $\mathcal{G}\boldsymbol{g}$ for contraction $\overline{\mathrm{grad}} \boldsymbol{N}_i : \mathcal{G}\boldsymbol{g}$ ;
 8          evaluate $\mathcal{C}\boldsymbol{g}_s := c_1 \boldsymbol{g}_s + c_2 \,\mathrm{tr}(\boldsymbol{g}_s) \boldsymbol{I}$ ;   // 2nd order symmetric
 9          queue $\mathcal{C}\boldsymbol{g}_s$ contraction $\overline{\mathrm{grad}^s} \boldsymbol{N}_i : \mathcal{C}\boldsymbol{g}_s$ ;
10      end
11      evaluate queued contractions using sum factorization and distribute
          results to the destination vector
12  end
```

For this work we adopt the matrix-free level operators within the GMG and employ Chebyshev pre- and post-smoothing on each level. In general, given a heterogeneous non-linear material, the definition of transfer operators[3] (for restriction and prolongation) and level operators is not trivial. One approach is to start from an arbitrary heterogeneous material at the fine scale and employ homogenization theory [18–21] to design suitable transfer operators [22][4]. In this approach, the fine-scale displacement is decomposed into long-wave and short-wave contributions by an additive split, where the former is associated to the homogeneous response of a patch of elements and the latter represents fluctuations

[3] Finite dimensional linear operators from a vector space associated with the fine-scale mesh to the coarse-scale mesh, and in the opposite direction.

[4] We note that this approach is limited to the case where for each quadrature point within an element the fourth-order tangent stiffness tensor is the same.

Algorithm 3: Matrix-free tangent operator: cache material part of the fourth-order spatial tangent stiffness tensor $\boldsymbol{C}$ and Kirchhoff stress $\boldsymbol{\tau}$.

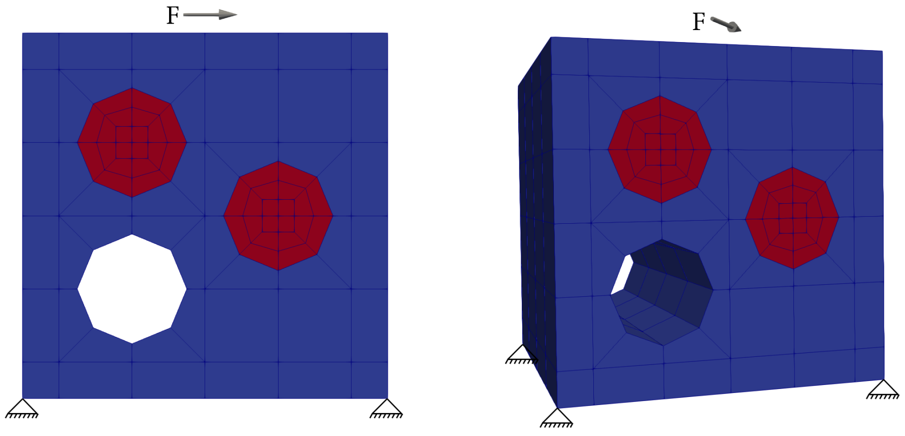
```
   Given  : Source FE vector $\boldsymbol{x}$, cached $\boldsymbol{\tau}/J$ and $\overline{\boldsymbol{C}}$ for each cell and
            quadrature point, evaluated based on $\overline{\boldsymbol{u}^h}$
   Return: action of the FE tangent operator (19) on $\boldsymbol{x}$
1  foreach element $K \in \Omega^h$ do
2      evaluate the following tensors at each quadrature point using sum
        factorization ;
3      $\boldsymbol{g}_s := \overline{\mathrm{grad}^s}\boldsymbol{x}$ ;                 // 2nd order symmetric
4      $\boldsymbol{g} := \overline{\mathrm{grad}}\boldsymbol{x}$ ;                          // 2nd order
5      foreach quadrature point q on K do
6          evaluate $\mathcal{G}\boldsymbol{g} := \boldsymbol{g} \cdot [\boldsymbol{\tau}/J]$ ;              // 2nd order
7          queue $\mathcal{G}\boldsymbol{g}$ for contraction $\overline{\mathrm{grad}}\boldsymbol{N}_i : \mathcal{G}\boldsymbol{g}$;
8          evaluate $\boldsymbol{C}\boldsymbol{g}_s := \overline{\boldsymbol{C}}\boldsymbol{:}\boldsymbol{g}_s$ ;        // 2nd order symmetric
9          queue $\boldsymbol{C}\boldsymbol{g}_s$ for contraction $\overline{\mathrm{grad}^s}\boldsymbol{N}_i : \boldsymbol{C}\boldsymbol{g}_s$ ;
10     end
11     evaluate queued contractions using sum factorization and distribute
        results to the destination vector
12 end
```

due to the heterogeneous structure of the patch. However, for a patch of elements consisting of the same material, the transfer operators reduce to the standard geometric transfer. Based on this observation we adopt the following approach: It is assumed that the mesh at the coarsest level can accurately describe the heterogeneous finite-strain elastic material. In this case, the restriction and prolongation operations for patches of elements are always performed for the same material and thus admits usage of standard geometric transfer operators. We define the level operators as follows: Recall that the fine scale tangent operator (19) is obtained by linearization around the current displacement field $\boldsymbol{u}^h$. We then restrict this displacement field to all multigrid levels $\{l\}$ and evaluate tangent operators using the matrix-free algorithms from Section 4. Essentially, level operators correspond to the linearization around the smoothed representation of displacement field[5]. Fi-

[5]Note that this differs from the classical approach where the (tangent) operator $\boldsymbol{A}^{l+1}$ on level $l+1$ is directly related to the (tangent) operator $\boldsymbol{A}^l$ on level l via $\boldsymbol{A}^{l+1} = \boldsymbol{I}_l^{l+1}\boldsymbol{A}^l\boldsymbol{I}_{l+1}^l$, where $\boldsymbol{I}_{l+1}^l$ and $\boldsymbol{I}_l^{l+1}$ are global prolongation (coarse-to-fine) and restriction (fine-to-coarse) operators, respectively.

nally, at the coarsest level we use a CG solver to reduce the residual by three orders of magnitude within (at most) one hundred steps. The multigrid algorithm is a V-cycle with two pre- and post-smoothing steps. For examples of other approaches that combine homogenization and multigrid methods, see [23, 24] for applications using small strain elastic materials and [25] with applications in fracture.

6. Numerical examples



(a) 2D coarse mesh (b) 3D coarse mesh

Figure 2: Discretization of the heterogeneous material at the coarsest mesh level and the prescribed boundary conditions.

In this section, we apply the proposed matrix-free operator algorithms for finite-strain hyperelastic materials as well as the geometric multigrid preconditioner to a benchmark problem from [22]. The coarse meshes for the 2D and 3D problems are illustrated in Figure 2. The 2D material consists of a square (depicted in blue), a hole and two spherical inclusions (depicted in red). The size of the square domain is 10^{-3}mm. The matrix material is taken to have Poisson's ratio 0.3 and shear modulus $\mu = 0.4225 \times 10^6\,\mathrm{N/mm}^2$. The inclusion is taken to be 100 times stiffer. The 3D material is obtained by extrusion of the 2D geometry into the third dimension. Note that although a relatively coarse mesh is used at the coarsest multigrid level, the geometry of the inclusions is captured accurately by manifold descriptions of the boundaries and interfaces. Both domains are fully fixed along the bottom surface and a distributed load is applied at the top in the $(1, 0)$ or $(1, 1, 0)$ direction for the 2D and the 3D problems, respectively. The force density $12.5 \times 10^3\text{N/mm}^2$ or $12.5\sqrt{2} \times 10^3\text{N/mm}^3$ is applied in 5 steps for the 2D and the 3D problems, respectively. With respect to the nonlinear solver, the displacement tolerance of the ℓ_2 norm for the Newton update is taken to be 10^{-5} whereas the relative and absolute tolerances for the residual forces is 10^{-8}. The relative convergence criteria for the linear solver is 10^{-6}. Shared memory parallelization using Intel Threading Building Blocks was disabled for this study.

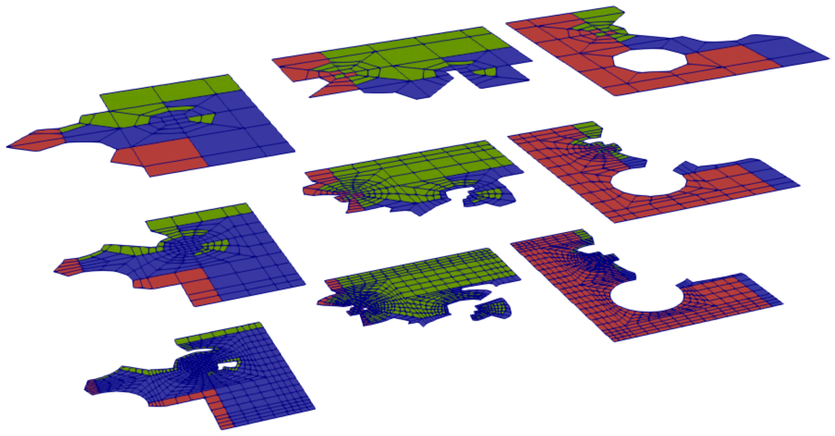


Figure 3: 2D multigrid mesh after two global refinements distributed into three MPI processes. The color indicates the process owning the element.

Figure 3 illustrates the hierarchy of multigrid meshes for the 2D problem after two global refinements. Figure 4 shows deformed meshes at the final loading step with quadratic elements and two global mesh refinements.

The benchmark calculations are performed for 2D and 3D problems with various combination of polynomial degrees and number of global refinements, as is stated in Tables 1 and 2. We are interested in the following metrics:

- the wall-clock time and memory requirement per DoF for matrix-vector multiplication;
- the average number of CG iterations throughout the entire simulation (i.e. each Newton-Raphson iteration and each loading step);
- the wall-clock time per DoF for the CG solver;

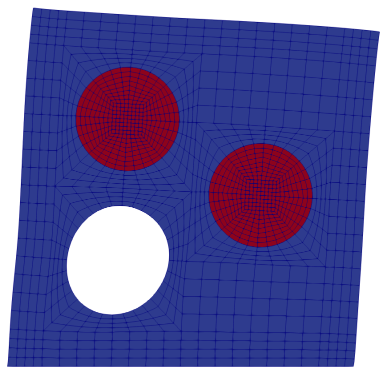
The results of the matrix-free approach with a geometric-multigrid preconditioner are compared to the matrix-based approach in conjunction with the algebraic

p	q	N_{gref}	N_{el}	N_{DoF}
1	2	7	1441792	2887680
2	3	6	360448	2887680
3	4	5	90112	1625088
4	5	5	90112	2887680
5	6	5	90112	4510720
6	7	4	22528	1625088
7	8	4	22528	2211328
8	9	4	22528	2887680

Table 1: Parameters for the 2D benchmark: p is the polynomial degree, q is the number of quadrature points in 1D, N_{gref} is the number of global mesh refinements, N_{el} is the number of elements and N_{DoF} is the number of DoFs.

p	q	N_{gref}	N_{el}	N_{DoF}
1	2	4	1441792	4442880
2	3	3	180224	4442880
3	4	2	22528	1891008
4	5	2	22528	4442880

Table 2: Parameters for the 3D benchmark: p is the polynomial degree, q is the number of quadrature points in 1D, N_{gref} is the number of global mesh refinements, N_{el} is the number of elements and N_{DoF} is the number of DoFs.


(a) 2D deformed mesh

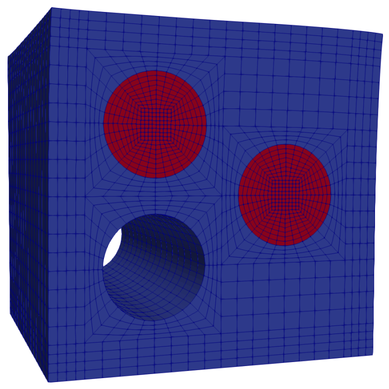
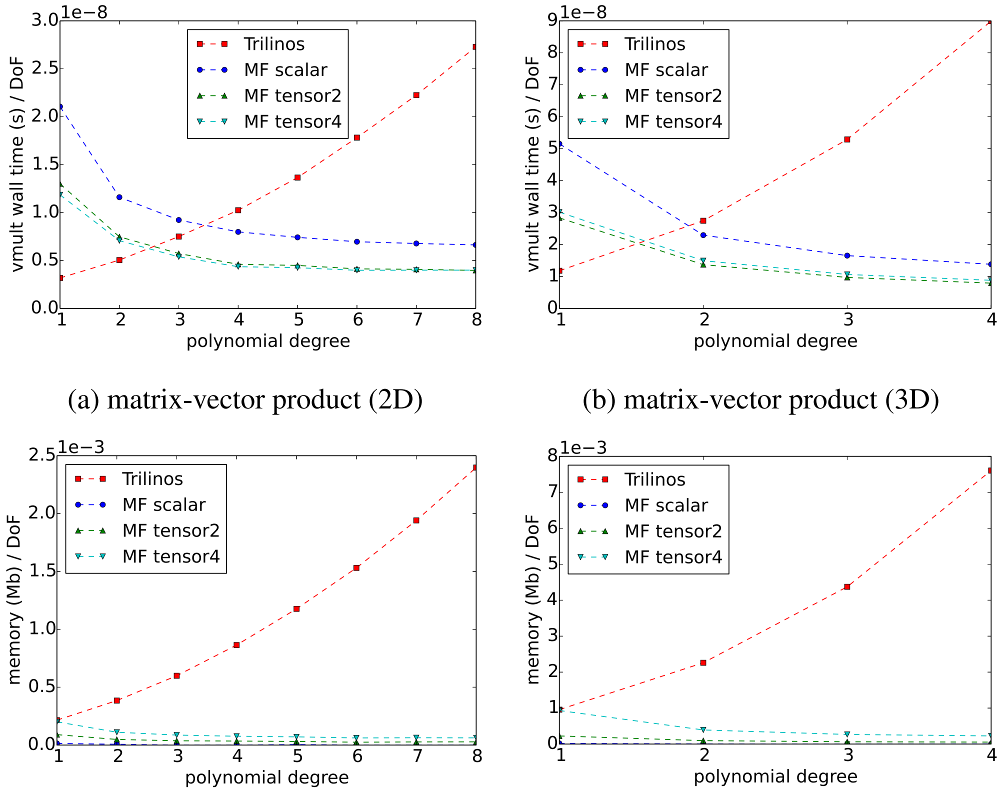

(b) 3D deformed mesh

Figure 4: Deformed meshes at the final loading step with quadratic elements and two global mesh refinements.

multigrid (AMG) preconditioner using the package ML of the Trilinos [26] library, version 12.12.1. The aggregation threshold for the AMG preconditioner is taken as 10^{-4}. Computations are done on a single node of two clusters: A node on the “Emmy” cluster at RRZE, FAU has two Xeon 2660v2 “Ivy Bridge” chips (10 cores per chip + SMT) running at 2.2 GHz with 25 MB shared cache per chip and 64 GB of RAM. Intel’s compiler version 18.03 with flags “-O3 -march=native” and Intel MPI version 18.03 are used. The “IWR” results were obtained on a single machine with eight Xeon E7-8870 “Sandy Bridge” chips (10 cores per chip + SMT) running at 2.4 GHz with 30 MB shared cache per chip. For the simulation, GCC version 8.1.0 with flags “-O3 -march=native” and OpenMPI version 3.0.0 were used.

6.1. Matrix-vector multiplication

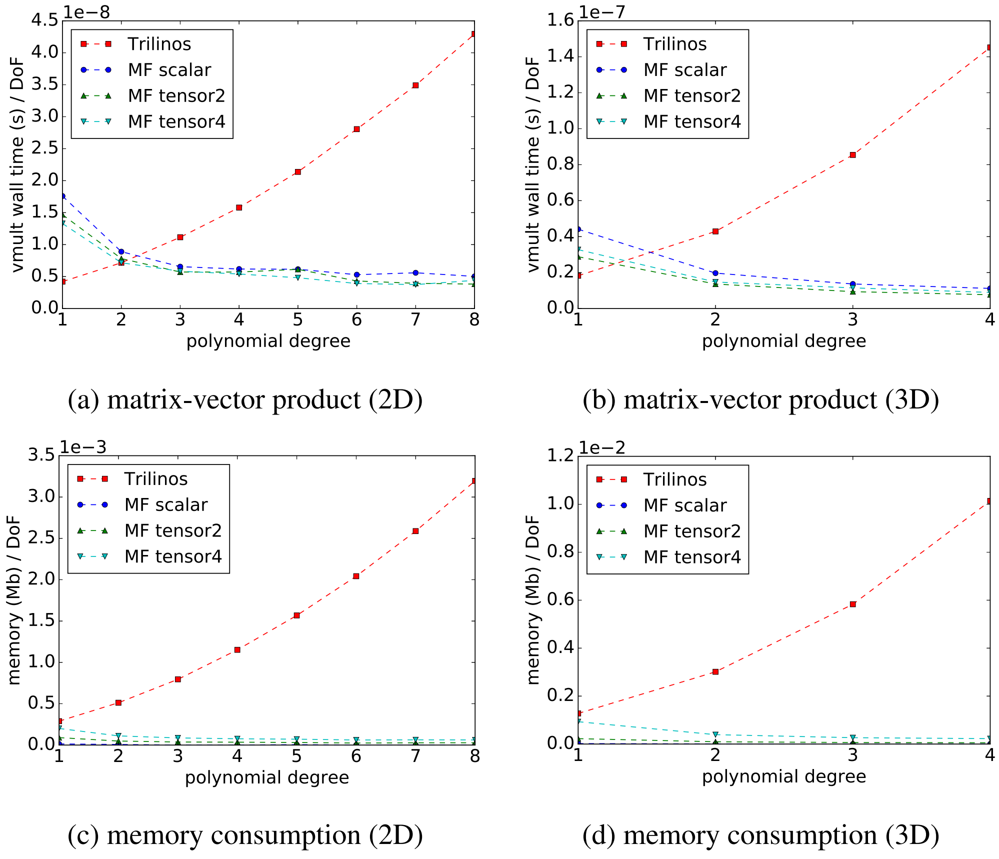
Figures 5 and 6 show the results of the matrix-vector multiplication for the considered finite-strain hyperelastic benchmark problem as performed on the two clusters. As expected, the matrix-vector multiplication becomes very expensive for higher order discretization for sparse matrix-based approaches. The matrix-free implementation is faster than the matrix-based counterpart already for cubic Lagrangian basis in 2D and quadratic in 3D. Figures 5(a), 5(b), 6(a) and 6(b) clearly show the influence of the different caching strategies. Two conclusions can be drawn from these results: The scalar caching implementation is the most time consuming of the three, as at each step we additionally evaluate the gradient



(a) matrix-vector product (2D)
(b) matrix-vector product (3D)
(c) memory consumption (2D)
(d) memory consumption (3D)

Figure 5: Emmy cluster, RRZE, matrix-vector multiplication.

of the displacement field in the referential configuration that is required to evaluate the Kirchhoff stress $\boldsymbol{\tau}$. There is little difference between the approach where the material part of the fourth-order spatial tangent stiffness tensor is cached ("tensor4") and the one where we only cache the second-order Kirchhoff tensor and utilize the chosen hyperelastic constitutive equation to efficiently implement the action of the material part of the fourth-order spatial tangent stiffness tensor on the second-order symmetric tensor. This indicates that the time savings we get from the latter approach are insignificant within the complete matrix-free operator implementation. Consequently, we can apply the matrix-free approach to any finite-strain material model by caching the material part of the fourth-order spatial tangent stiffness tensor in addition to the Kirchhoff stress and expect this to be faster than the matrix-based implementation. It is therefore feasible to define



(a) matrix-vector product (2D) (b) matrix-vector product (3D)

(c) memory consumption (2D) (d) memory consumption (3D)

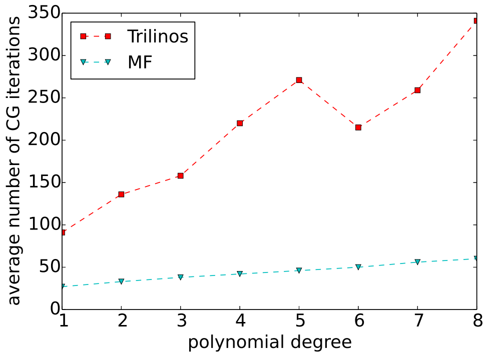
Figure 6: IWR cluster, matrix-vector multiplication

highly performant generic operators that are independent of any applied constitutive laws.

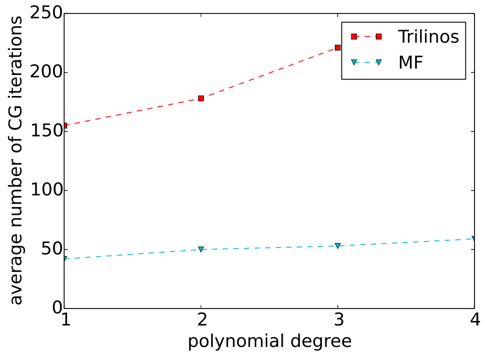
As outlined in the introduction, it is not only the performance of the matrix-free vector multiplication, but also the memory requirement to store a sparse matrix which is the driving force behind matrix-free methods. Figures 5(c), 5(d), 6(c) and 6(d) demonstrate that even with caching one fourth-order symmetric tensor and one second-order tensor at each quadrature point, the matrix-free approach takes much less memory than its matrix-based counterpart.

6.2. *Preconditioned iterative solver*

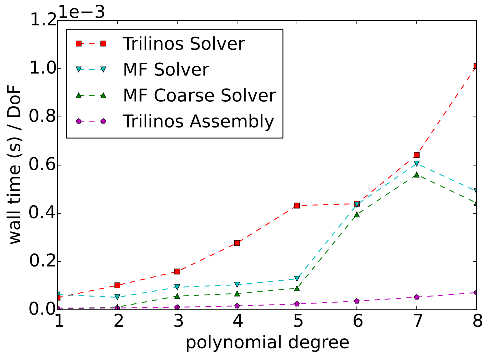
Next we evaluate the performance of the proposed geometric multigrid preconditioner. First we consider the number of iterations in the conjugate gradient



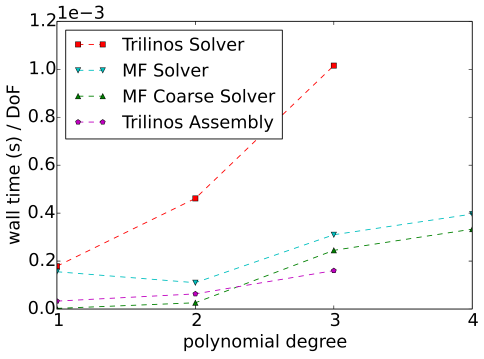
(a) CG iterations (2D)



(b) CG iterations (3D)



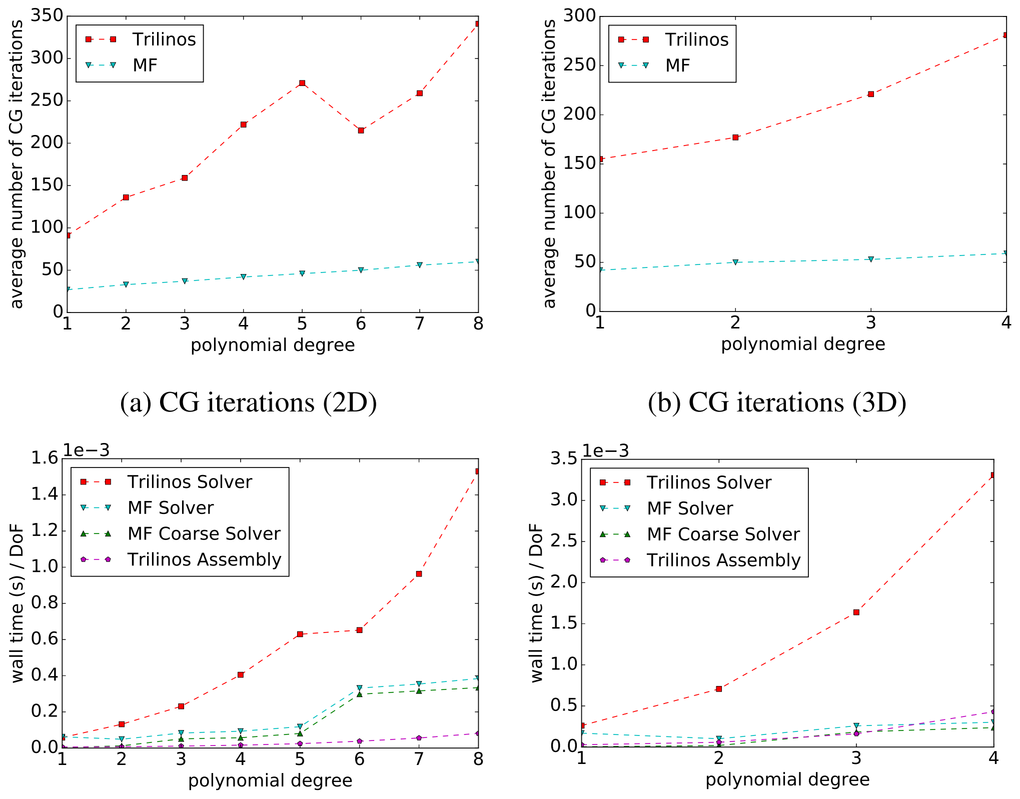
(c) CG solution time (2D)



(d) CG solution time (3D)

Figure 7: Emmy cluster, RRZE, iterative solver.

(CG) solver. Figures 7(a), 7(b), 8(a) and 7(b) show a very mild increase in the average number of CG iterations for different polynomial degrees. On the other hand, the black-box AMG preconditioner requires many more iterations for convergence. Figures 7(c), 7(d), 8(c) and 8(d) confirm that the efficiency of the proposed GMG preconditioner translates into faster solution times for polynomial degrees up to $p = 5$. For higher orders in 2D the wall-clock time is comparable to the AMG preconditioner, see Figure 7(c). The reason for this is the suboptimal coarse level solver within the GMG preconditioner. Evidently, the combination of the Chebyshev smoother with the CG solver at the coarsest mesh level has unsatisfactory performance for higher polynomial degrees. One possible solution to this issues would be to perform further coarsening combined with the homogenization-based transfer operators [22]. Although untested in this work, we hypothesize that



(a) CG iterations (2D) (b) CG iterations (3D)

(c) CG solution time (2D) (d) CG solution time (3D)

Figure 8: IWR cluster.

this should also improve the performance of the preconditioner for lower degrees. A valid alternative would be to use matrix-based AMG at the coarsest level.

7. Summary and Conclusions

In this contribution, we proposed and numerically investigated several matrix-free implementations of tangent operators for finite-strain elasticity with heterogeneous materials. The implementation that caches the material part of the fourth-order spatial tangent stiffness tensor together with the second-order Kirchhoff stress was shown to be faster than the matrix-based approach for quadratic elements in 3D and cubic in 3D. This gives hope that the matrix-free implementation of the tangent operator can be applied to any constitutive model which operates with the material part of the fourth-order spatial tangent stiffness tensor on the quadrature point level.

We also studied the performance of the GMG preconditioner with standard geometric transfer operations between each level. The GMG was applied to heterogeneous material assuming that the coarsest level can provide an adequate discretization of the heterogeneity. The numerical studies indicate that the proposed preconditioner leads to a solution approach that is faster than the matrix-based AMG for moderately high polynomial degrees (approximately up to order five). However, a considerable amount of time of GMG preconditioner is spent in the coarse level solver. Our future work will, therefore, be focused on extending the proposed matrix-free multigrid solution approach to homogenization based transfer operators [22]. We are also interested in extending these methods to near incompressible three-field formulations.

Acknowledgements

D. Davydov acknowledges the financial support of the German Research Foundation (DFG), grant DA 1664/2-1.

D. Davydov, D. Arndt and J-P. Pelteret are grateful to Jed Brown (CU Boulder) and Veselin Dobrev (Lawrence Livermore National Laboratory) for a fruitful discussion on matrix-free operator evaluation approaches.

D. Arndt was supported by the German Research Foundation (DFG) under the project “High-order discontinuous Galerkin for the exa-scale” (ExaDG) within the priority program “Software for Exascale Computing” (SPPEXA).

P. Steinmann acknowledges the support of the Cluster of Excellence Engineering of Advanced Materials (EAM) which made this collaboration possible, as well as funding by the EPSRC Strategic Support Package “Engineering of Active Materials by Multiscale/Multiphysics Computational Mechanics”.

Bibliography

[1] M. Kronbichler, K. Kormann, A generic interface for parallel cell-based finite element operator application, Computers & Fluids 63 (2012) 135–147.

[2] D. A. May, J. Brown, L. Le Pourhiet, A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow, Computer methods in applied mechanics and engineering 290 (2015) 496–523.

[3] B. Krank, N. Fehn, W. A. Wall, M. Kronbichler, A high-order semi-explicit discontinuous Galerkin solver for 3D incompressible flow with application to DNS and LES of turbulent channel flow, Journal of Computational Physics 348 (2017) 634–659.

[4] J. Brown, Efficient nonlinear solvers for nodal high-order finite elements in 3D, Journal of Scientific Computing 45 (1-3) (2010) 48–63.

[5] B. Gmeiner, M. Huber, L. John, U. Rüde, B. Wohlmuth, A quantitative performance study for Stokes solvers at the extreme scale, Journal of Computational Science 17 (2016) 509–521.

[6] A. Abdelfattah, M. Baboulin, V. Dobrev, J. Dongarra, C. Earl, J. Falcou, A. Haidar, I. Karlin, T. Kolev, I. Masliah, et al., High-performance tensor contractions for GPUs, Procedia Computer Science 80 (2016) 108–118.

[7] K. Ljungkvist, M. Kronbichler, Multigrid for matrix-free finite element computations on graphics processors, Tech. rep., Technical Report 2017-006, Department of Information Technology, Uppsala University (2017).

[8] T. C. Clevenger, T. Heister, G. Kanschat, M. Kronbichler, A Flexible, Parallel, Adaptive Geometric Multigrid method for FEM, in preparation.

[9] M. Mentler, High performance implementation of one-field elasticity using the deal.II Finite Element library, Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (2017).

[10] P. Wriggers, Nonlinear finite element methods, Springer Science & Business Media, 2008.

[11] P. E. Vos, S. J. Sherwin, R. M. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations, Journal of Computational Physics 229 (13) (2010) 5161–5181.

[12] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells, The `deal.II` Library, Version 9.0, Journal of Numerical Mathematics.

[13] J. H. Bramble, J. E. Pasciak, J. Xu, Parallel multilevel preconditioners, Mathematics of Computation 55 (191) (1990) 1–22.

[14] W. L. Briggs, V. E. Henson, S. F. McCormick, A Multigrid Tutorial: Second Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[15] B. Janssen, G. Kanschat, Adaptive Multilevel Methods with Local Smoothing for H^1- and H^{curl}-Conforming High Order Finite Element Methods, SIAM Journal on Scientific Computing 33 (4) (2011) 2095–2114. doi:10.1137/090778523.

[16] W. Hackbusch, Multi-grid methods and applications, Springer Series in Computational Mathematics, Springer-Verlag Berlin Heidelberg, 1985. doi:10.1007/978-3-662-02427-0.

[17] P. Wesseling, An introduction to multigrid methods, Pure and applied mathematics, John Wiley & Sons, 1992.

[18] P. Suquet, Elements of homogenization for inelastic solid mechanics, in: E. Sanchez-Palencia, A. Zaoui (Eds.), Homogenization techniques for composite media, Vol. 272 of Lecture Notes in Physics, Springer-Verlag Berlin Heidelberg, 1987, pp. 193–278.

[19] R. Hill, On constitutive macro-variables for heterogeneous solids at finite strain, Proc. R. Soc. Lond. A 326 (1565) (1972) 131–147.

[20] Z. Hashin, Analysis of composite materials–a survey, Journal of Applied Mechanics 50 (3) (1983) 481–505.

[21] P. P. Castaneda, P. Suquet, Nonlinear composites, in: Advances in applied mechanics, Vol. 34, Elsevier, 1997, pp. 171–302.

[22] C. Miehe, C. Bayreuther, On multiscale FE analyses of heterogeneous structures: from homogenization to multigrid solvers, International Journal for Numerical Methods in Engineering 71 (10) (2007) 1135–1180.

[23] C. G. Bayreuther, C. Miehe, Coupling of homogenization techniques with multigrid solvers for unstructured meshes, in: Analysis and Simulation of Multifield Problems, Springer, 2003, pp. 67–72.

[24] J. Fish, V. Belsky, Multi-grid method for periodic heterogeneous media part 2: Multiscale modeling and quality control in multidimensional case, Computer Methods in Applied Mechanics and Engineering 126 (1) (1995) 17–38.

[25] L. Kaczmarczyk, C. Pearce, N. Bicanic, E. de Souza Neto, Numerical multiscale solution strategy for fracturing heterogeneous materials, Computer Methods in Applied Mechanics and Engineering 199 (17-20) (2010) 1100–1113. doi:10.1016/j.cma.2009.11.018.
URL `http://eprints.gla.ac.uk/38423/`

[26] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, K. S. Stanley, An overview of the Trilinos project, ACM Trans. Math. Softw. 31 (3) (2005) 397–423. doi:http://doi.acm.org/10.1145/1089014.1089021.