A MATLAB Script for Solving 2D/3D Minimum Compliance Problems using Anisotropic Mesh Adaptation

Jensen, Kristian Ejlebjærg

Published in:
Procedia Engineering

Link to article, DOI:
10.1016/j.proeng.2017.09.792

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 203 (2017) 102–114

Procedia Engineering

www.elsevier.com/locate/procedia

26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain

A MATLAB Script for Solving 2D/3D Minimum Compliance Problems using Anisotropic Mesh Adaptation

Kristian Ejlebjerg Jensen[a,*]

[a]*Department of Micro- and Nanotechnology, Technical University of Denmark, Ørsteds Plads, DK-2800 Kgs. Lyngby, Denmark*

Abstract

We present a pure MATLAB implementation for solving 2D/3D compliance minimization problems using the density method. A filtered design variable with a minimum length is computed using a Helmholtz-type differential equation. The optimality criteria is used as optimizer and to avoid local minima we apply continuation of an exponent that controls the stiffness associated with intermediate design variables. We constrain the volume from above and use the implementation to show that optimizations with dynamic meshes can save significant amounts of computational time compared to fixed meshes without introducing mesh dependence for the mesh topology. This is despite the fact that the dynamic meshes cause oscillations of the objective function, particular for coarse meshes in 3D. The meshes are generated using anisotropic mesh adaptation based on local mesh modifications and we extent these modifications to preserve the information required for interpolating the design variables between meshes. We exploit symmetry boundaries in 3D, but not in 2D. Dirichlet boundary conditions are used to prevent non-zero filtered design variables on free boundaries. Mesh adaptation involves substantial book keeping, so the implementation totals some 5,000 lines of MATLAB code, but the functions associated with the forward analysis, geometry/mesh setup and optimization are concise and well documented, so the implementation can be used as a starting point for research on related topics.

Keywords: MATLAB ; adaptation; elasticity; compliance; optimization

1. Introduction

1.1. Topology optimization

Compliance minimization using the density method is a decades old technology for saving weight in structural members without sacrificing stiffness [1]. It is a free form optimization method that allows topologically different designs to arise, i.e. it is a topology optimization method. The density method has been widely used outside structural optimization [2,3] and it remains the most popular topology optimization method, but many other techniques exist, see [4] and references therein. Structured meshes remain the norm despite the prevalence of unstructured meshes within

* Corresponding author. Tel.: +45 4525 5770.
E-mail address: kristian.jensen@nanotech.dtu.dk

10.1016/j.proeng.2017.09.792

computationally aided engineering in industry, lately dynamic unstructured meshes are, however, starting to become more popular [5,6] [1].

Nomenclature

$\mathbf{u}$	displacement vector
$\underline{\underline{\sigma}}$	stress tensor
$\underline{\underline{\epsilon}}$	deformation tensor
E	young's modulus
P	SIMP exponent
$L_{\min}$	filter length
ρ	design variable
$\tilde{\rho}$	filtered design variable
G	shear modulus
$\underline{\underline{\mathbf{H}}}$	hessian
$\underline{\underline{\text{abs}}}$	absolute tensor in principal frame (i.e. take positive value of eigenvalues)
λ	Lamé's first parameter
Ω	domain
$\underline{\underline{\mathbf{I}}}$	identity tensor
$\underline{\underline{\mathcal{M}}}$	metric tensor
q	norm in which to minimize interpolation error
η	scaling factor
$\hat{\mathbf{n}}$	unit normal vector
V_{frac}	volume fraction
N_t	mesh complexity
$i_{\max}$	number of optimization iterations to use
L_{char}	characteristic length
L_1	length associated with load area
L_x	length in x-direction
L_z	length in z-direction
i	iteration number
κ	adjoint scalar (orthogonal to variations in ρ)
$\tilde{\mathbf{u}}$	adjoint vector (orthogonal to variations in $\tilde{\rho}$)
d	dimensionality
ν	Poisson ratio

1.2. Anisotropic mesh adaptation

Simulations where the solution has discontinuities are best accelerated using h-type mesh adaptation and when the solutions has anisotropic features, it can be advantageous to elongate the elements accordingly. The continuous mesh framework is the most popular setting for anisotropic mesh adaptation [8,9], because it is an excellent way to address the issue of which mesh to generate, without having to worry about how to generate it. Local mesh modification operations (as shown in figure 1) constitute the most robust and popular method for generating good discrete meshes [10], but more recently it has been shown that methods based on advancing fronts can be superior in terms of mesh quality [11]. One issue is the significant amount of book keeping associated with dynamic mesh adaptation software and the scarcity of open source implementations [12–14].

[1] The published MATLAB script shares implementation details with a WCSMO12 conference paper on heat optimization (ID 62) [7]

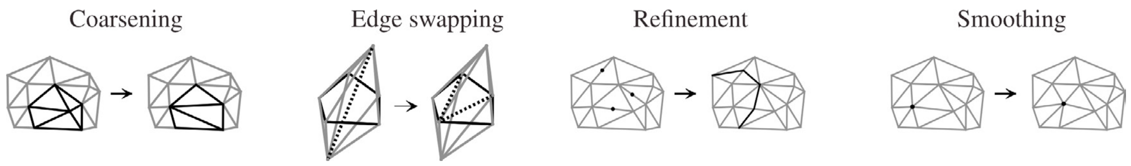

Fig. 1. Four local mesh modification operations are sketched. None of the operations are allowed to decrease worst local element quality associated with the operation, except for coarsening in 3D. The operations are implemented in the `adapt_rmnd`, `adapt_flipedg`, `adapt_add_nd` and `adapt_mvnd` functions. Note that we do not apply 3D face swapping (also called 2-to-3 swapping), because the applied implementation has been found [13] to generate equally good meshes without it.

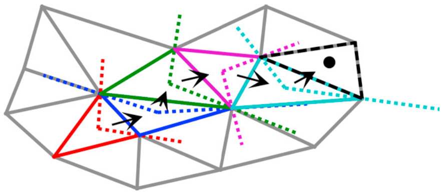

Fig. 2. The information required for interpolating a nodal field is preserved during the mesh adaptation by relying on the fact that we can guess an element (red) close to the element containing the old node. By dividing the domain in 3 (4 in 3D), we can make a decision as to which of the neighboring elements is the better candidate. We continue this process until the correct element is found (dashed black). This functionality is implemented in the `elem_interp` and `elem_find` functions.

It has been shown [15] that the interpolation error of a function, f, is minimized, if the mesh on which the interpolation is represented satisfies the metric

$$\underline{\underline{\mathcal{M}}} = \frac{1}{\eta}\left[\det(\underline{\underline{\mathbf{H}}})\right]^{-\frac{1}{2q+3}} \underline{\underline{\mathbf{H}}} \quad \text{where} \quad \underline{\underline{\mathbf{H}}} = \underline{\underline{\text{abs}}}\left(\underline{\underline{\mathbf{H}}}(f)\right). \tag{1}$$

Here q is the norm of the error to be minimized, det is the determinant, $\underline{\underline{\mathbf{H}}}$ is the Hessian and $\underline{\underline{\text{abs}}}$ is the absolute value in the principal frame. Finally, η is a scaling factor, which can be eliminated by introducing a desired mesh complexity [9], N_t, where

$$N_t = \int_\Omega \sqrt{\det(\underline{\underline{\mathcal{M}}})}d\Omega \tag{2}$$

It is important to note that the metric is symmetric and positive definite, which means that one can think of it is a field of ellipses (ellipsoids in 3D) that describe the optimal size and orientation of elements in a continuous way. The functionality for calculating the metric using equations (1) and (2) is available in the `metric_pnorm` and `metric_scale` functions. Note that the former takes a nodal scalar as input and applies Galerkin projections for derivative recovery, so that a nodal Hessian can be computed, i.e.

$$0 = \int_\Omega (\text{grad}(f) - \nabla f) \cdot \mathcal{V}_{\text{test}} d\Omega$$

$$0 = \int_\Omega (\underline{\underline{\mathbf{H}}}(f) - \nabla\text{grad}(f)) : \underline{\underline{\mathcal{V}}}_{\text{test}} d\Omega,$$

where $\mathcal{V}_{\text{test}}$ and $\underline{\underline{\mathcal{V}}}_{\text{test}}$ are 1st order nodal vector and tensor test functions, respectively.

We use an open source MATLAB implementation of anisotropic mesh adaptation [13], which has been extended with this functionality for metric computation. Furthermore, we need interpolation of nodal fields between meshes and we handle this as part of the local operations, see figure 2.

2. Setup

The density method is a popular heuristic for solving integer optimization problems by replacing the discrete variables with continuous ones. The design variable, ρ, can thus take on any value between between zero and one, but the problem is stated such as to discourage intermediate design variables. Therefore the Young's modulus, E is interpolated in the design variable using the following relation [1]

$$E(\rho) = E_{\min} + (E_{\max} - E_{\min})\rho^P.$$

That is to say that we approximate void ($\rho = 0$) with a material having a low modulus $E_{\min}$ relative to solid ($\rho = 1$), where the modulus is $E_{\max}$. The SIMP exponent, P, controls the stiffness associated with intermediate design variables. It is well-known that $P = 1$ gives rise to a convex problem with a lot of intermediate design, but we want a 0-1 solution and thus apply an exponential continuation [16] in the exponent for the first half of the optimizations

$$P_i = \min\left((1+d)^{\frac{2i}{i_{\max}}}, 1+d\right),$$

where $i_{\max}$ is the total number of iterations, i is the iterations number and d is the dimensionality. To quantify the amount of non-discreteness the following functional can be computed [17]

$$\mathrm{ND} = \int_\Omega 4\rho(1-\rho)d\Omega \bigg/ \int_\Omega d\Omega. \tag{3}$$

The functional is computed in the `get_ND` function. For a fully non-dicrete design ($\rho = 0.5$) the functional evaluates to 100 %, while it evaluates to 0 % for a perfect 0-1 design. The stiffest design will always be $\rho = 1$, so we impose a volume constraint,

$$0 > \int_\Omega \rho d\Omega \bigg/ \int_\Omega d\Omega - V_{\mathrm{frac}}. \tag{4}$$

In order to form a well-posed problem we filter the design using a Helmholtz-type PDE [18], so that we get a filtered design $\tilde{\rho}$ with a minimum length scale, $L_{\min}$. We then use this to calculate the Young's modulus, $E(\tilde{\rho})$ for the linear elasticity analysis. The governing equations thus become

$$\begin{aligned}
&\mathbf{0} = \boldsymbol{\nabla}\cdot\underline{\underline{\sigma}} \quad \text{with} \quad \underline{\underline{\sigma}} = \underline{\underline{\sigma}}_{\mathrm{load}} \quad \text{on} \quad \partial\Omega_{\mathrm{load}} \quad \text{and} \quad \mathbf{u} = \mathbf{0} \quad \text{on} \quad \partial\Omega_{\mathrm{support}}, \quad \text{where}\\
&\underline{\underline{\sigma}} = 2G\underline{\underline{\epsilon}} + \lambda\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}}) \quad \text{and} \quad \underline{\underline{\epsilon}} = \tfrac{1}{2}(\boldsymbol{\nabla}\mathbf{u} + [\boldsymbol{\nabla}\mathbf{u}]^T)\\
&G = \frac{E(\tilde{\rho})}{2(1+\nu)}, \quad \lambda = \frac{E(\tilde{\rho})\nu}{(1+\nu)(1-2\nu)}\\
&\tilde{\rho} = L_{\min}^2\boldsymbol{\nabla}^2\tilde{\rho} + \rho \quad \text{and} \quad \boldsymbol{\nabla}\tilde{\rho}\cdot\hat{\mathbf{n}} = 0 \quad \text{on} \quad \partial\Omega_{\mathrm{unfree}} \quad \text{and} \quad \tilde{\rho} = 0 \quad \text{on} \quad \partial\Omega_{\mathrm{free}}
\end{aligned} \tag{5}$$

where $\underline{\underline{\sigma}}$, $\underline{\underline{\epsilon}}$ and $\underline{\underline{\mathbf{I}}}$ are the stress, deformation and identity tensor. $\mathbf{u}$ is the displacement vector, G is the shear modulus, λ is the Lamé's first parameter and ν is the Poisson ratio. We use the compliance as objective function

$$\phi = \int_\Omega \underline{\underline{\epsilon}} : \underline{\underline{\sigma}} d\Omega \tag{6}$$

It can be shown (see appendix) that this objective function makes the problem self-adjoint, so that the continuous sensitivity becomes

$$\begin{aligned}
&\frac{\partial\phi}{\partial\rho} = \kappa, \quad \text{where} \quad \kappa = L_{\min}^2\boldsymbol{\nabla}^2\kappa + \frac{\partial\phi}{\partial\tilde{\rho}}, \quad \boldsymbol{\nabla}\kappa\cdot\hat{\mathbf{n}} = 0 \quad \text{on} \quad \partial\Omega_{\mathrm{unfree}}, \quad \kappa = 0 \quad \text{on} \quad \Omega_{\mathrm{free}} \quad \text{and}\\
&\frac{\partial\phi}{\partial\tilde{\rho}} = -\underline{\underline{\epsilon}} : \left(2\frac{\partial G}{\partial\tilde{\rho}}\underline{\underline{\epsilon}} + \frac{\partial\lambda}{\partial\tilde{\rho}}\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}})\right)
\end{aligned} \tag{7}$$

In other words we can compute the sensitivity with respect to the filtered design variable explicitly, but in order to get the sensitivity with respect to the design variable itself, we have to apply the PDE filter again. By doing this we arrive at a consistent sensitivity, which sets this work apart from [6].

We use the sensitivity to drive the mesh adaptation by requiring that the mesh minimizes the 2-norm of the interpolation error, see equation (1). The discrete nodal sensitivity can be calculated by multiplying the nodal values of the continuous sensitivity with the design variable volumes associated with the nodes, see figure 3. This is done when the

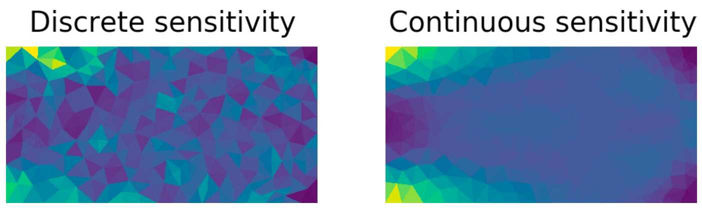

Fig. 3. The continuous sensitivity to the left is related to the discrete sensitivty to the right by way of the volumes associated with the elements. In the interest of clarity this figure shows element wise sensitivities, but we use nodal sensitivities.

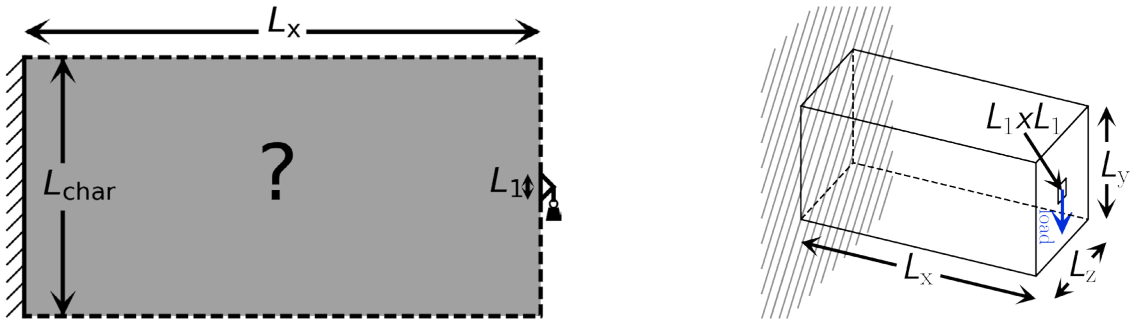

Fig. 4. The 2D and 3D cantilever problems are plotted with the support to the left and the load to the right. The initial volume and boundary mesh for these problems are generated in the `mesh_cant` and `mesh_cant3D` functions.

mesh has been adapted and the design variable as well as the continuous sensitivity have been interpolated on to the new mesh. That is to say that the mesh is adapted, before the design variables are updated.

We find that it is advantageous to use nodal design variables and we discretize the forward problem as well as the filters with linear simplex finite elements. To avoid problems with negative filtered design variables and to simplify assembly of the linear system associated with the forward problem, we compute a corrected filtered design variable, which is defined element-wise as the interpolated value in the center of the element with values smaller than 0 set to 0 and values larger than 1 set to 1. In other words we solve equation (5) using the a finite element method, which is not monotonous and therefore we choose to correct the extreme values explicitly.

The code uses the optimality criteria (`optC`) as optimizer and the snippet is taken directly from [19], which also describes the simple principle on which it relies.

The complete MATLAB code is released as an open source script[2]. We use a direct solver for all linear system, including the Galerkin projections associated with the derivative recovery requried for metric computations. The linear systems are assembled and solved in the `fem_tri2xy`, `fem_filter` and `fem_hooke` functions.

We consider the 2D and 3D cantilever benchmark problems as sketched in figure 4. In all cases we use the maximum Young's modulus as characteristic stress. We use the length of the domain in the y-dimension as characteristic length scale, and set $L_x = 2L_{\mathrm{char}}$, $L_1 = 0.1L_{\mathrm{char}}$ and $\nu = 0.3$. For the 2D problem we consider plane stress only, so that

$$\underline{\underline{\sigma}} = 2G\underline{\underline{\epsilon}} + \lambda\underline{\underline{\mathbf{I}}}\left(\mathrm{Tr}(\underline{\underline{\epsilon}}) + \partial_z u_z\right), \quad \text{where} \quad \partial_z u_z = -\frac{\nu}{1-\nu}\nabla\cdot\mathbf{u} \quad \text{in} \quad 2\mathrm{D}$$

Note that the correction $\partial_z u_z$ is also incorporated into equation (7).

For the 3D problem we fix $L_z = 0.5L_{\mathrm{char}}$ and make use of symmetry so that only a quarter of the computational domain has to be simulated. We consider the case of a single as well as two load cases. The two load cases are combined by taking the mean of their respective objective functions.

$$\underline{\underline{\sigma}}_{\mathrm{load}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{in} \quad 2\mathrm{D}$$

[2] `https://github.com/kristianE86/trullekrul`

$$\underline{\underline{\sigma}}_{\text{load}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{or} \quad \underline{\underline{\sigma}}_{\text{load}} = \begin{bmatrix} 0 & 0 & 0.5 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{bmatrix} \quad \text{in} \quad 3\text{D}$$

We also vary, the target node number, the total number of iterations, the filter length and the volume fraction as listed in table 1.

Table 1. The table shows tested parameters, where N_t is the desired mesh complexity (NA for fixed meshes), i_{max} is the iterations count, $L_{\min}$ is the filter length and V_{frac} is the volume fraction, d is the dimension and h_i is the initial element size. Symmetry is exploited in 3D, but not in 2D. The tests can be reproduced using the instructions in appendix Appendix C. The objective function value, node count, computational time and non-discreteness measure is written in the titles of the figures in section 3.

No	N_t	$i_{\max}$	$L_{\min}$	V_{frac}	$\tilde{E}_{\min}$	d	h_i	load cases
1	NA	566	0.02	0.5	0.001	2	0.01	1
2	750	400	0.02	0.5	0.001	2	0.05	1
3	1500	566	0.02	0.5	0.001	2	0.05	1
4	3000	800	0.02	0.5	0.001	2	0.05	1
5	NA	566	0.01	0.2	0.001	3	0.02	1
6	1500	400	0.01	0.2	0.001	3	0.1	1
7	4243	566	0.01	0.2	0.001	3	0.1	1
8	12000	800	0.01	0.2	0.001	3	0.1	1
9	NA	566	0.02	0.1	0.001	3	0.02	2
10	2000	400	0.02	0.1	0.001	3	0.1	2
11	5657	566	0.02	0.1	0.001	3	0.1	2
12	16000	800	0.02	0.1	0.001	3	0.1	2
13	40000	800	0.002	0.01	0.0001	3	0.1	2

Dirichlet boundary conditions are imposed for $\tilde{\rho}$ and κ on boundaries, where there is neither load, support or symmetry conditions. Finally, it is important to note that we chose to reduce the relative computational cost of mesh adaptation by only adapting the mesh every 5th optimization iteration.

3. Results and discussion

The iteration count, objective value, number of nodes, computational time[3] and non-discreteness measure (see equation (3)) is written in the title of figures 5, 7, 8 and 9.

The 2D designs are shown in figure 5 and their objective functions are plotted in figure 6. The design topology is independent of the mesh, but the optimizations with dynamic meshes tend to produce similar objective functions with less computational cost. The objective function decreases with mesh refinement, which could be caused by the areas between solid-void transitions where the design variable has to take on intermediate values. This hypothesis is consistent with the tendency of the non-discreteness measure to decrease with mesh refinement.

The results of the 3D optimization for one and two load case are shown in figure 7 and 8, respectively, while the corresponding optimizations with fixed meshes are shown in figure 9. Besides iso-surfaces and slices, we also show a wireframe mesh on the boundary of the computational domain. Finally, the objective functions are plotted in figure 10. We also ran optimizations with $V_{\text{frac}} = 0.1$ and $L_{\min} = 0.005$ (see appendix Appendix B), but the fixed meshes ($h_i = 0.02$) cannot resolve this filter length.

The optimizations with a single load case all give an I-beam design, which is in agreement with previous work [6,20,21] and although the objective function oscillates significantly in 3D, the oscillations still decrease with mesh refinement and do not prevent the dynamic meshes from providing significant computational savings compared to the fixed mesh. The results for the problem with two load cases confirm these observations. Figure 9(right) shows the result of optimization No 13 with $L_{\min} = 2 \cdot 10^{-3}$ and a 1 % volume fraction. It is thus possible to get designs with

[3] on a Intel Xeon E5-2680 (2.80 GHz).

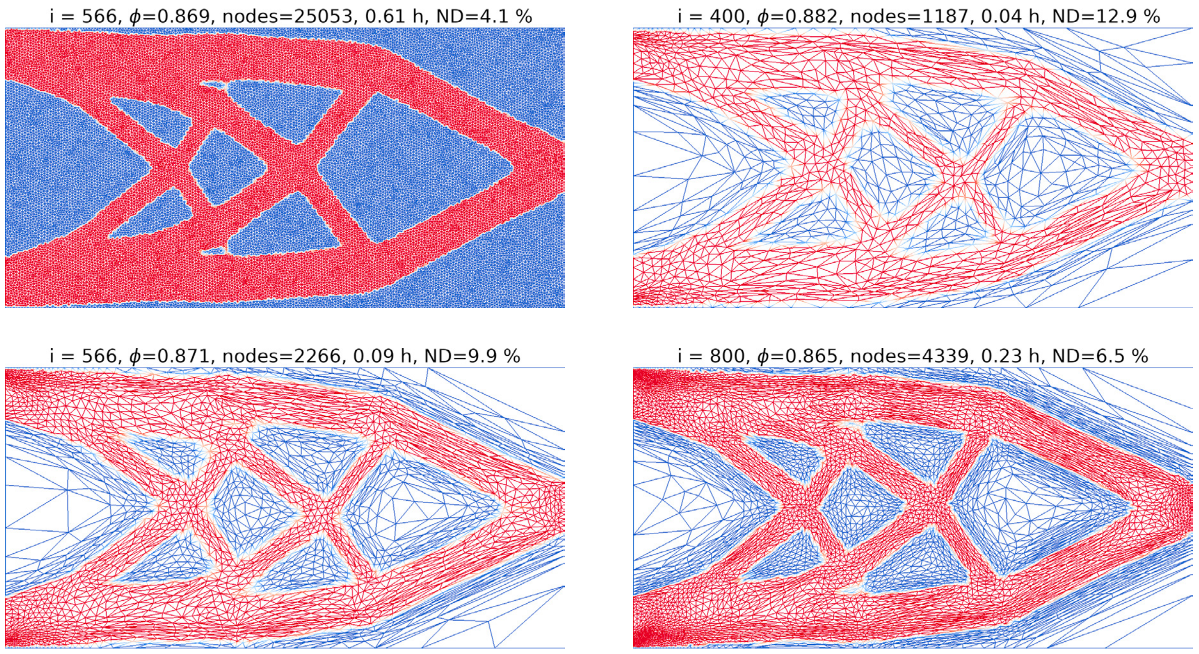
i = 566, ϕ=0.869, nodes=25053, 0.61 h, ND=4.1 %
i = 400, ϕ=0.882, nodes=1187, 0.04 h, ND=12.9 %
i = 566, ϕ=0.871, nodes=2266, 0.09 h, ND=9.9 %
i = 800, ϕ=0.865, nodes=4339, 0.23 h, ND=6.5 %

Fig. 5. The results of optimizations No 1-4 (see table 1) indicate that the dynamic meshes give rise to the same topology with less computational effort.

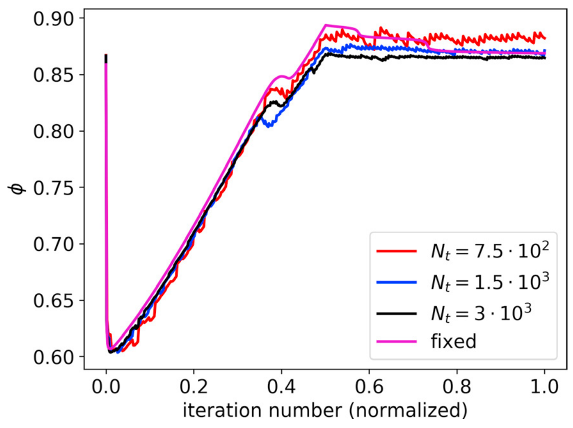

Fig. 6. The objective functions for the 2D optimizations are plotted versus the normalized iteration number. The dynamic meshes tend to produce some oscillations, particularly for the coarse meshes, but the dynamic meshes consistently gives better objective functions than the fixed mesh.

low volume fractions using this method, but we have been unable to show mesh independence for such optimizations and the topology seems suboptimal in the sense that connections are curved and misaligned. The oscillations are significantly smaller in [6] and [7], which might be due to the use of coarse meshes.

Finally, figure 11 shows that the computation of the displacement field take up more than half of the computational time for problem No 12, while the mesh adaptation and metric calculation take up a combined 30 %. If the mesh was adapted every iteration instead of every 5th, one would thus expect it to dominate the cost. This is partly due to the bandwidth limited nature of the vectorized MATLAB implementation [13], but on the other hand the computation of the forward problem could also be optimized by making use of previous iterations [22] and for the fixed meshes, one could use a structured mesh taking advantage of the speed-up that this facilitates [23]. In terms of absolute speed, the volume fractions are important, because smaller values indicate larger weight savings. The 3D optimizations with a single load case in figure 9 and appendix Appendix B take at least 0.6 h and 1.2 h for $V_{\text{frac}} = 0.2$ and $V_{\text{frac}} = 0.1$,

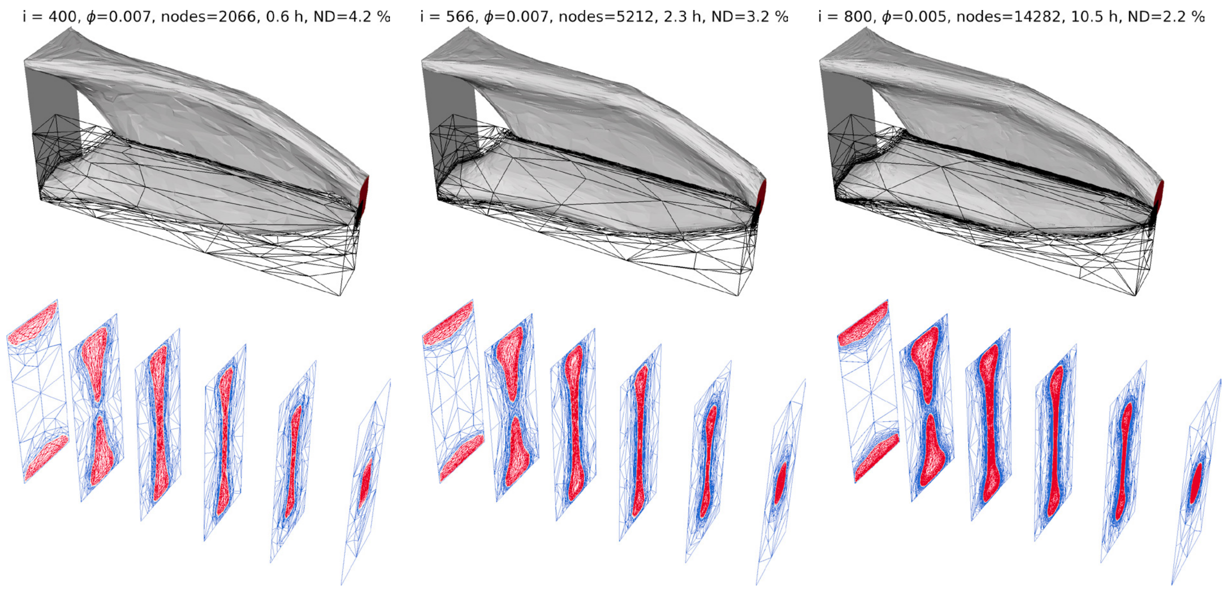
i = 400, ϕ=0.007, nodes=2066, 0.6 h, ND=4.2 %
i = 566, ϕ=0.007, nodes=5212, 2.3 h, ND=3.2 %
i = 800, ϕ=0.005, nodes=14282, 10.5 h, ND=2.2 %

Fig. 7. The result of optimizing for a single load case is shown by means of an iso-surface and slices orthogonal to the *x*-direction. All optimizations give an I-beam design, but there are large variatios in the objective functions. The corresponding optimization with a fixed mesh is shown in figure 9 (left).

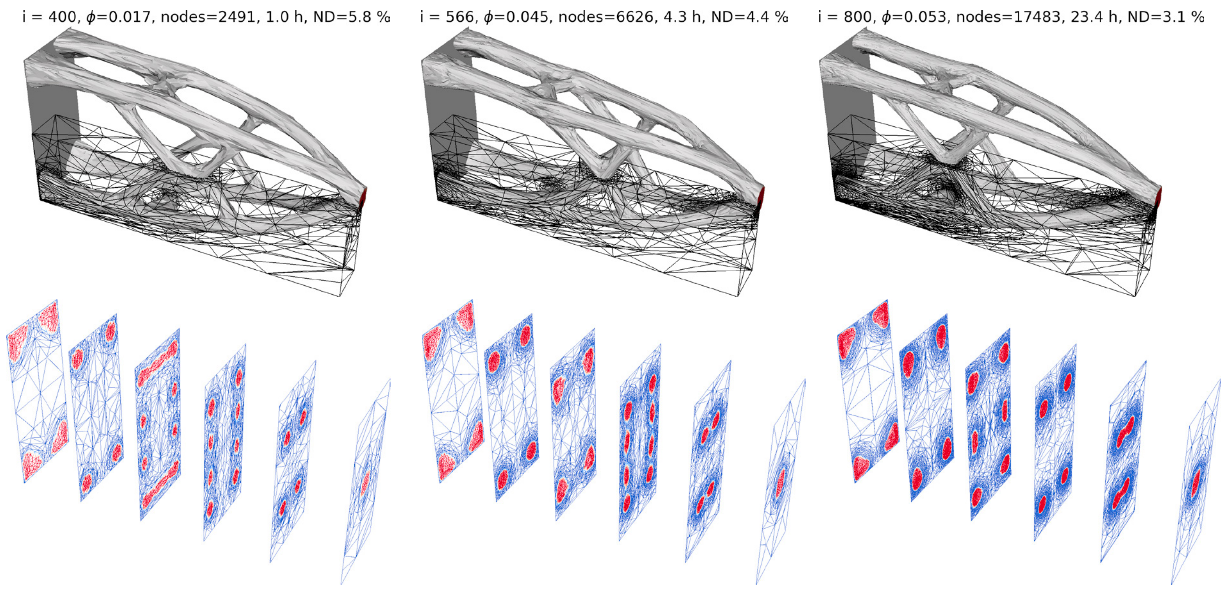
i = 400, ϕ=0.017, nodes=2491, 1.0 h, ND=5.8 %
i = 566, ϕ=0.045, nodes=6626, 4.3 h, ND=4.4 %
i = 800, ϕ=0.053, nodes=17483, 23.4 h, ND=3.1 %

Fig. 8. Optimizing for two load cases leads to truss-structure, which appers to be mesh independent, but there are some variation in slice 3, 4 and 5, which we attribute to minor differences between the designs. Figure 9(right) shows the corresponding optimization with a fixed mesh.

respectively, while [6] take at least 2.8 h for $V_{\text{frac}} = 0.1$ even though it uses the same mesh adaptation code and only half the number of iterations, so the fact that the mesh is only adapted every 5th iteration is critical for achieving efficiency. A much older work [2] used 16 CPUs to solve a similar problem with $V_{\text{frac}} = 0.5$ in 3.9 h. on a structured mesh.

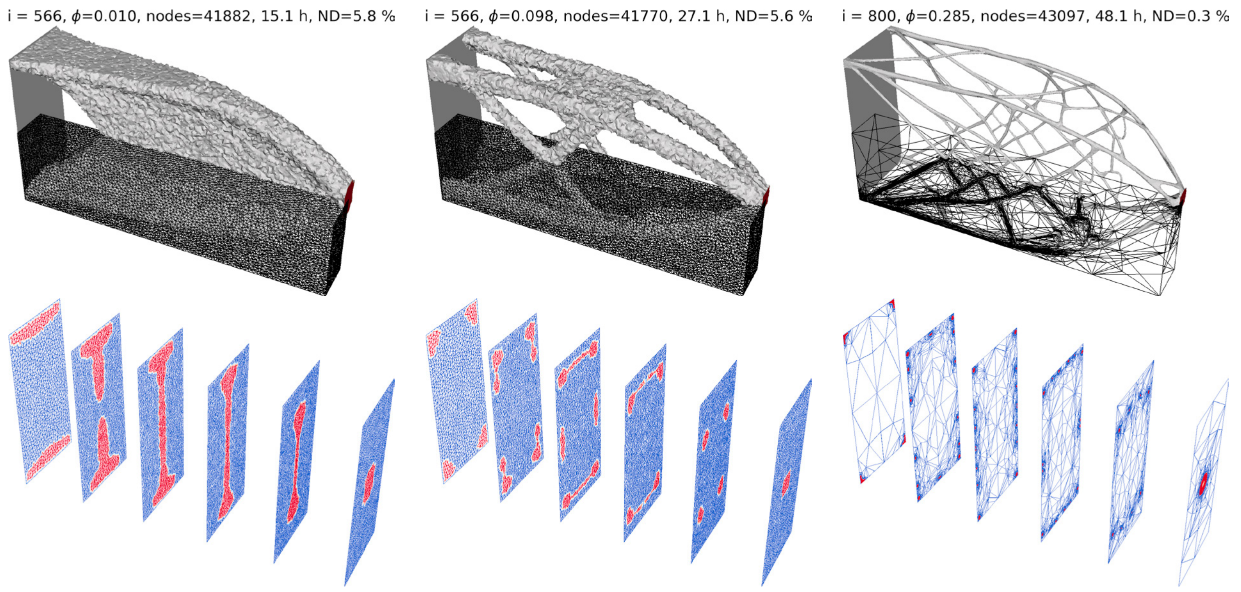
i = 566, ϕ=0.010, nodes=41882, 15.1 h, ND=5.8 %
i = 566, ϕ=0.098, nodes=41770, 27.1 h, ND=5.6 %
i = 800, ϕ=0.285, nodes=43097, 48.1 h, ND=0.3 %

Fig. 9. The results of the 3D optimizations with fixed meshes (No 5 and 9) are shown for one (left) and two load cases (center). The corresponding optimization with fixed meshes are shown in figures 7 and 8. Note that the large linear systems associated with these optimizations require significant amounts of memory. Finally, the result of optimization No 13 is shown to the right.

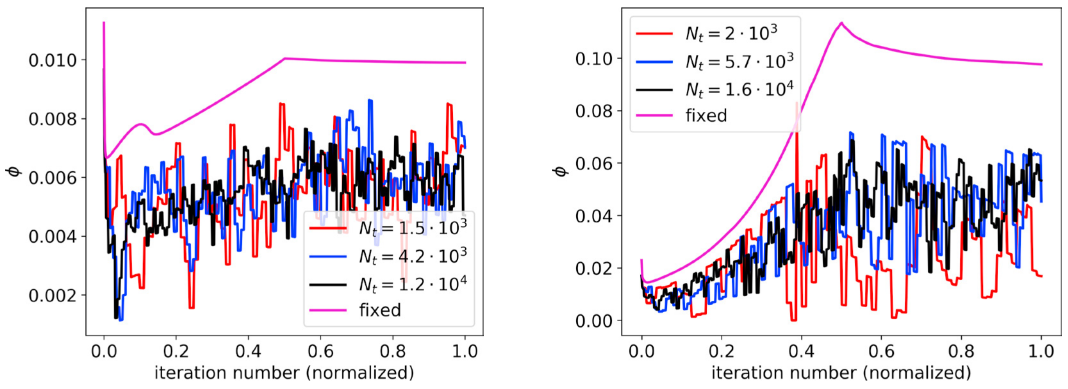

Fig. 10. The dynamic meshes give rise to large oscillations in 3D, but they still outperform the fixed meshes.

4. Conclusion

We have demonstrated a pure MATLAB implementation for solving volume constrained minimum compliance problems using the density method and anisotropic mesh adaptation. The mesh adaptation relies on local mesh modifications and in performing these we make sure to preserve the information required for interpolating the nodal design variables and sensitivities between meshes. We are able to demonstrate mesh independence for a 2D/3D cantilever problems and in 3D we even achieve this for two load cases. There are issues with oscillating objective functions in 3D, but these appear to decrease with mesh refinement and so does the degree of non-discreteness.

Despite the tendency of the MATLAB implementation mesh adaptation to require substantial computational effort, the dynamic meshes still tend to provide better objective functions than the fixed meshes – and for a fraction of the computational effort.

The published code facilitates reproducibility, but we also hope that it can serve to increase the use of dynamic meshes within the field of topology optimization and research on computationally aided engineering in general.

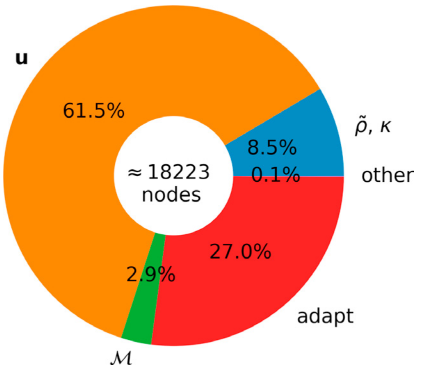

Fig. 11. Optimization No 12 with an average node count of 18576 is analyzed with respect to the relative amount of computational time spent in various segments of the code. Calculation of the displacement field dominates, but the mesh adaptation and the related metric computation also take up a significant proportion of the computational time.

Acknowledgements

This work is supported by the Villum Foundation (Grant No. 9301) and the Danish Council for Independent Research (DNRF122).

Appendix A. Continuous sensitivity analysis

The objective function and governing equations are

$$\begin{aligned}
\phi &= \int_\Omega \underline{\underline{\epsilon}} : \underline{\underline{\sigma}} d\Omega \\
\mathbf{0} &= \boldsymbol{\nabla} \cdot \underline{\underline{\sigma}} \quad \text{with} \quad \underline{\underline{\sigma}} = \underline{\underline{\sigma}}_{\text{load}} \quad \text{on} \quad \partial\Omega_{\text{load}} \quad \text{and} \quad \mathbf{u} = \mathbf{0} \quad \text{on} \quad \partial\Omega_{\text{support}} \\
\underline{\underline{\sigma}} &= 2G\underline{\underline{\epsilon}} + \lambda \underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}}) \quad \text{and} \quad \underline{\underline{\epsilon}} = \tfrac{1}{2}(\boldsymbol{\nabla}\mathbf{u} + [\boldsymbol{\nabla}\mathbf{u}]^T) \\
G &= \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad E = E_{\min} + (E_{\max} - E_{\min})\tilde{\rho}^P, \\
\tilde{\rho} &= L_{\min}^2 \boldsymbol{\nabla}^2 \tilde{\rho} + \rho \quad \text{and} \quad \tilde{\rho} = 0 \quad \text{on} \quad \Omega_{\text{free}}, \quad \boldsymbol{\nabla}\tilde{\rho} \cdot \hat{\mathbf{n}} = 0 \quad \text{on} \quad \Omega_{\text{unfree}}
\end{aligned} \tag{A.1}$$

We now introduce the adjoint displacement field, $\tilde{\mathbf{u}}$, which is invariant with respect to perturbations in the filtered design variable field.

$$\begin{aligned}
0 &= \int_\Omega \tilde{\mathbf{u}} \cdot \boldsymbol{\nabla} \cdot \underline{\underline{\sigma}} d\Omega \\
&= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \underline{\underline{\sigma}} \cdot \hat{\mathbf{n}} ds - \int_\Omega \boldsymbol{\nabla}\tilde{\mathbf{u}} : \underline{\underline{\sigma}} d\Omega \\
&= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \underline{\underline{\sigma}} \cdot \hat{\mathbf{n}} ds - \int_\Omega \tilde{\epsilon} : \underline{\underline{\sigma}} d\Omega \quad \text{where} \quad \tilde{\epsilon} = \tfrac{1}{2}(\boldsymbol{\nabla}\tilde{\mathbf{u}} + [\boldsymbol{\nabla}\tilde{\mathbf{u}}]^T)
\end{aligned}$$

The variation with respect to the filtered design variable, $\tilde{\rho}$ becomes

$$\begin{aligned}
0 &= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \frac{\partial \underline{\underline{\sigma}}}{\partial\tilde{\rho}} \cdot \hat{\mathbf{n}} \delta\tilde{\rho} ds - \int_\Omega \left[\frac{\partial \tilde{\epsilon}}{\partial \tilde{\rho}} : \underline{\underline{\sigma}} + \underline{\underline{\tilde{\epsilon}}} : \frac{\partial \underline{\underline{\sigma}}}{\partial\tilde{\rho}} \right] \delta\tilde{\rho} d\Omega \\
&= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \frac{\partial \underline{\underline{\sigma}}}{\partial\tilde{\rho}} \cdot \hat{\mathbf{n}} \delta\tilde{\rho} ds - \int_\Omega \left[\underline{\underline{\tilde{\epsilon}}} : \left(2\frac{\partial G}{\partial\tilde{\rho}} \underline{\underline{\epsilon}} + 2G\frac{\partial \underline{\underline{\epsilon}}}{\partial\tilde{\rho}} + \frac{\partial\lambda}{\partial\tilde{\rho}} \underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}}) + \lambda\underline{\underline{\mathbf{I}}} \left[\underline{\underline{\mathbf{I}}} : \frac{\partial \underline{\underline{\epsilon}}}{\partial\tilde{\rho}} \right] \right) \right] \delta\tilde{\rho} d\Omega \\
&= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \frac{\partial \underline{\underline{\sigma}}}{\partial\tilde{\rho}} \cdot \hat{\mathbf{n}} \delta\tilde{\rho} ds - \int_\Omega \left[\underline{\underline{\tilde{\epsilon}}} : \left(2\frac{\partial G}{\partial\tilde{\rho}} \underline{\underline{\epsilon}} + 2G\frac{\partial \underline{\underline{\epsilon}}}{\partial\tilde{\rho}} \right) + \frac{\partial\lambda}{\partial\tilde{\rho}} \underline{\underline{\epsilon}} : \underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\tilde{\epsilon}}}) + \frac{\partial\epsilon}{\partial\tilde{\rho}} : \lambda\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\tilde{\epsilon}}}) \right] \delta\tilde{\rho} d\Omega
\end{aligned} \tag{A.2}$$

where we have used that

$$\underline{\underline{\tilde{\epsilon}}} : \lambda\underline{\underline{\mathbf{I}}}\left[\underline{\underline{\mathbf{I}}} : \frac{\partial\epsilon}{\partial\tilde{\rho}}\right] = \frac{\partial\epsilon}{\partial\tilde{\rho}} : \lambda\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\tilde{\epsilon}}}) \tag{A.3}$$

The variation of the objective function with respect to $\tilde{\rho}$ becomes

$$\begin{aligned}
\delta\phi &= \int_\Omega \left(\frac{\partial\underline{\underline{\epsilon}}}{\partial\tilde{\rho}} : \underline{\underline{\sigma}} + \underline{\underline{\epsilon}} : \frac{\partial\underline{\underline{\sigma}}}{\partial\tilde{\rho}}\right)\delta\tilde{\rho}d\Omega \\
&= \int_\Omega \left(\frac{\partial\underline{\underline{\epsilon}}}{\partial\tilde{\rho}} : \left[2G\underline{\underline{\epsilon}} + \lambda\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}})\right] + \underline{\underline{\epsilon}} : \left[2\frac{\partial G}{\partial\tilde{\rho}}\underline{\underline{\epsilon}} + 2G\frac{\partial\underline{\underline{\epsilon}}}{\partial\tilde{\rho}} + \frac{\partial\lambda}{\partial\tilde{\rho}}\lambda\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}}) + \lambda\underline{\underline{\mathbf{I}}}\left(\underline{\underline{\mathbf{I}}} : \frac{\partial\underline{\underline{\epsilon}}}{\partial\tilde{\rho}}\right)\right]\right)\delta\tilde{\rho}d\Omega \\
&= \int_\Omega \left(\frac{\partial\underline{\underline{\epsilon}}}{\partial\tilde{\rho}} : \left[4G\underline{\underline{\epsilon}} + 2\lambda\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}})\right] + \underline{\underline{\epsilon}} : \left[2\frac{\partial G}{\partial\tilde{\rho}}\underline{\underline{\epsilon}} + \frac{\partial\lambda}{\partial\tilde{\rho}}\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}})\right]\right)\delta\tilde{\rho}d\Omega,
\end{aligned} \tag{A.4}$$

Where we once again made use of equation (A.3)[4]. Adding two times equation (A.2) to (A.4) yields

$$\begin{aligned}
\delta\phi &= 2\int_{\partial\Omega}\tilde{\mathbf{u}}\cdot\frac{\partial\underline{\underline{\sigma}}}{\partial\tilde{\rho}}\cdot\hat{\mathbf{n}}\delta\tilde{\rho}ds \\
&+ \int_\Omega \frac{\partial\underline{\underline{\epsilon}}}{\partial\tilde{\rho}} : \left(4G(\underline{\underline{\epsilon}} - \underline{\underline{\tilde{\epsilon}}}) + 2\lambda\underline{\underline{\mathbf{I}}}\left[\mathrm{Tr}(\underline{\underline{\epsilon}}) - \mathrm{Tr}(\underline{\underline{\tilde{\epsilon}}})\right]\right)\delta\tilde{\rho}d\Omega \\
&+ \int_\Omega \underline{\underline{\epsilon}} : \left(2\frac{\partial G}{\partial\tilde{\rho}}(\underline{\underline{\epsilon}} - 2\underline{\underline{\tilde{\epsilon}}}) + \frac{\partial\lambda}{\partial\tilde{\rho}}\underline{\underline{\mathbf{I}}}\left[\mathrm{Tr}(\underline{\underline{\epsilon}}) - 2\mathrm{Tr}(\underline{\underline{\tilde{\epsilon}}})\right]\right)\delta\tilde{\rho}d\Omega,
\end{aligned} \tag{A.5}$$

so the derivative of $\underline{\underline{\epsilon}}$ with respect to $\tilde{\rho}$ drops out, when $\underline{\underline{\epsilon}} = \underline{\underline{\tilde{\epsilon}}}$, i.e. the problem is self-adjoint. Furthermore the boundary term drops out, if only fixed load and support ($\mathbf{u} = \tilde{\mathbf{u}} = \mathbf{0}$) boundary conditions are used. Thus the sensitivity becomes

$$\frac{\partial\phi}{\partial\tilde{\rho}} = -\underline{\underline{\epsilon}} : \left(2\frac{\partial G}{\partial\tilde{\rho}}\underline{\underline{\epsilon}} + \frac{\partial\lambda}{\partial\tilde{\rho}}\underline{\underline{\mathbf{I}}}\mathrm{Tr}(\underline{\underline{\epsilon}})\right)$$

We now introduce an adjoint field, κ, which is invariant with respect to perturbations in the design variable.

$$\begin{aligned}
0 &= \int_\Omega \kappa\left(\rho - \tilde{\rho} + L_{\min}^2\nabla^2\tilde{\rho}\right)d\Omega \\
&= \int_{\partial\Omega}\kappa L_{\min}^2\overbrace{\boldsymbol{\nabla}\tilde{\rho}\cdot\hat{\mathbf{n}}}\,0ds + \int_\Omega\left(\kappa(\rho - \tilde{\rho}) - L_{\min}^2\boldsymbol{\nabla}r\tilde{h}o\cdot\boldsymbol{\nabla}\kappa\right)d\Omega \\
&= \int_{\partial\Omega}L_{\min}^2\left(\kappa\boldsymbol{\nabla}\tilde{\rho} - \tilde{\rho}\boldsymbol{\nabla}\kappa\right)\cdot\hat{\mathbf{n}}ds + \int_\Omega\left(\kappa(\rho - \tilde{\rho}) + \tilde{\rho}L_{\min}^2\nabla^2\kappa\right)d\Omega
\end{aligned}$$

The variation with respect to the design variable thus becomes

$$0 = \int_{\partial\Omega}L_{\min}^2\left(\kappa\boldsymbol{\nabla}\frac{\partial\tilde{\rho}}{\partial\rho} - \frac{\partial\tilde{\rho}}{\partial\rho}\boldsymbol{\nabla}\kappa\right)\cdot\hat{\mathbf{n}}\delta\rho ds + \int_\Omega\left(\kappa\left(1 - \frac{\partial\tilde{\rho}}{\partial\rho}\right) + \frac{\partial\tilde{\rho}}{\partial\rho}L_{\min}^2\nabla^2\kappa\right)\delta\rho d\Omega \tag{A.6}$$

The variation of the objective function can be expressed as

$$\delta\phi = \int_\Omega\frac{\partial\phi}{\partial\tilde{\rho}}\frac{\partial\tilde{\rho}}{\partial\rho}\delta\rho d\Omega$$

Adding equation (A.6) yields

$$\delta\phi = \int_\Omega\left(\frac{\partial\phi}{\partial\tilde{\rho}}\frac{\partial\tilde{\rho}}{\partial\rho} + \kappa\left(1 - \frac{\partial\tilde{\rho}}{\partial\rho}\right) + \frac{\partial\tilde{\rho}}{\partial\rho}L_{\min}^2\nabla^2\kappa\right)\delta\rho d\Omega$$

[4] with $\underline{\underline{\tilde{\epsilon}}} = \underline{\underline{\epsilon}}$

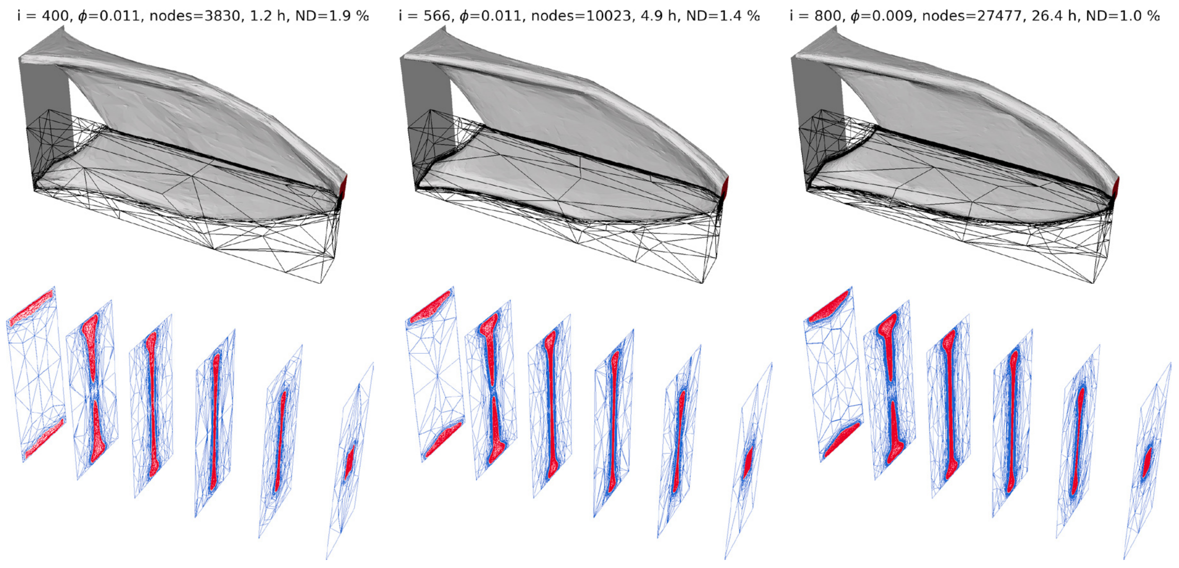
i = 400, ϕ=0.011, nodes=3830, 1.2 h, ND=1.9 %
i = 566, ϕ=0.011, nodes=10023, 4.9 h, ND=1.4 %
i = 800, ϕ=0.009, nodes=27477, 26.4 h, ND=1.0 %

Fig. B.12. The result of optimizing for a single load case with V_{frac}=0.1 is shown. The design is topologically independent, but there is a slight variation in the gap size for slice # 2.

$$- \int_{\partial\Omega} L^2_{\min} (\kappa\boldsymbol{\nabla}\tilde{\rho} - \tilde{\rho}\boldsymbol{\nabla}\kappa) \cdot \hat{\mathbf{n}} ds + \int_{\Omega} \left(\kappa(\rho - \tilde{\rho}) + \tilde{\rho} L^2_{\min} \nabla^2\kappa\right) \cdot \hat{\mathbf{n}} \delta\rho ds$$
$$= \int_{\Omega} \left(\frac{\partial\tilde{\rho}}{\partial\rho} \left[\frac{\partial\phi}{\partial\tilde{\rho}} - \kappa + L^2_{\min}\nabla^2\kappa \right] + \kappa \right) \delta\rho d\Omega$$
$$+ \int_{\partial\Omega} L^2_{\min} (\kappa\boldsymbol{\nabla}\tilde{\rho} - \tilde{\rho}\boldsymbol{\nabla}\kappa) \cdot \hat{\mathbf{n}} ds + \int_{\Omega} \left(\kappa(\rho - \tilde{\rho}) + \tilde{\rho} L^2_{\min} \nabla^2\kappa\right) \cdot \hat{\mathbf{n}} \delta\rho ds$$

I.e.

$$\frac{\partial\phi}{\partial\rho} = \kappa, \quad \text{where} \quad \kappa = L^2_{\min}\boldsymbol{\nabla}^2\kappa + \frac{\partial\phi}{\partial\tilde{\rho}}, \quad \boldsymbol{\nabla}\kappa \cdot \hat{\mathbf{n}} = 0 \quad \text{on} \quad \partial\Omega_{\text{unfree}} \quad \text{and} \quad \kappa = 0 \quad \text{on} \quad \Omega_{\text{free}}$$

Appendix B. One load case with $V_{\text{frac}} = 0.1$

Setting $V_{\text{frac}} = 0.1$ and $L_{\min} = 0.005$ with $N_t = 3000$, $N_t = 8485$ and $N_t = 24000$ for $it_{\max} = 400$, $it_{\max} = 566$ and $it_{\max} = 800$ yields the results in figure B.12

Appendix C. Reproduction

The results can be reproduced using the `top5001.m` script (available at
`https://github.com/kristianE86/trullekrul`):

```
top5001(nan,2e-2,0.5,0.3,false,2,1, [],1/50,400,1e-2,'fig5a',1.025,5,false);
top5001(7.5e2,2e-2,0.5,0.3,false,2,1, [],1/20,400,1e-2,'fig5b',1.025,5,false);
top5001(1.5e3,2e-2,0.5,0.3,false,2,1, [],1/20,566,1e-2,'fig5c',1.025,5,false);
top5001(3e3  ,2e-2,0.5,0.3,false,2,1, [],1/20,800,1e-2,'fig5d',1.025,5,false);
top5001(nan ,2e-2 ,0.2,0.3,true ,2,0.5,0.25,1/50,566,1e-3,'fig9a',1.025,false,2);
top5001(1.5e3 ,2e-2 ,0.2,0.3,true ,2,0.5,0.25,1/10,400,1e-3,'fig7a',1.025,5,2);
top5001(round(3e3*sqrt(2)) ,2e-2 ,0.2,0.3,true ,2,0.5,0.25,1/10,566,1e-3,'fig7b',1.025,5,2);
top5001(1.2e4 ,2e-2 ,0.2,0.3,true ,2,0.5,0.25,1/10,800,1e-3,'fig7c',1.025,5,2);
top5001(nan ,2e-2 ,0.1,0.3,2    ,2,0.5,0.25,1/50,566,1e-3,'fig9b',1.025,false,2);
top5001(2e3 ,2e-2 ,0.1,0.3,2    ,2,0.5,0.25,1/10,400,1e-3,'fig8a',1.025,5,2);
top5001(round(4e3*sqrt(2)) ,2e-2,0.1,0.3,2    ,2,0.5,0.25,1/10,566,1e-3,'fig8b',1.025,5,2);
```

```
top5001(1.6e4 ,2e-2 ,0.1,0.3,2  ,2,0.5,0.25,1/10,800,1e-3,'fig8c',1.025,5,2);
top5001(4e4 ,2e-3 ,0.01,0.3,2   ,2,0.5,0.25,1/10,800,1e-4,'fig9c',1.025,5,2);
top5001(3e3 ,5e-3 ,0.1,0.3,true ,2,0.5,0.25,1/10,400,1e-3,'fig12a',1.025,5,2);
top5001(round(6e3*sqrt(2)) ,5e-3 ,0.1,0.3,true ,2,0.5,0.25,1/10,566,1e-3,'fig12b',1.025,5,2);
top5001(2.4e3 ,5e-3 ,0.1,0.3,true ,2,0.5,0.25,1/10,800,1e-3,'fig12c',1.025,5,2);
```

References

[1] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Computer methods in applied mechanics and engineering 71 (1988) 197–224.

[2] T. Borrvall, J. Petersson, Topology optimization of fluids in stokes flow, International journal for numerical methods in fluids 41 (2003) 77–107.

[3] A. Gersborg-Hansen, M. P. Bendsøe, O. Sigmund, Topology optimization of heat conduction problems using the finite volume method, Structural and multidisciplinary optimization 31 (2006) 251–259.

[4] O. Sigmund, K. Maute, Topology optimization approaches, Structural and Multidisciplinary Optimization 48 (2013) 1031–1055.

[5] A. N. Christiansen, J. A. Bærentzen, M. Nobel-Jørgensen, N. Aage, O. Sigmund, Combined shape and topology optimization of 3d structures, Computers & Graphics 46 (2015) 25–35.

[6] K. E. Jensen, Anisotropic mesh adaptation and topology optimization in three dimensions, Journal of Mechanical Design 138 (2016) 061401.

[7] K. E. Jensen, Combining anisotropic mesh adaptation, heat conduction and topology optimization, in: 12th World Congress on Structural and Multidisciplinary Optimization, Springer, 2017.

[8] C. Pain, A. Umpleby, C. De Oliveira, A. Goddard, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Computer Methods in Applied Mechanics and Engineering 190 (2001) 3771–3796.

[9] A. Loseille, F. Alauzet, Continuous mesh framework part i: well-posed continuous interpolation error, SIAM Journal on Numerical Analysis 49 (2011) 38–60.

[10] L. A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, International Journal for Numerical Methods in Engineering 40 (1997) 3979–4002.

[11] A. Loseille, Metric-orthogonal anisotropic mesh generation, Procedia Engineering 82 (2014) 403–415.

[12] Y. Vassilevski, Advanced numerical methods in mesh generation and mesh adaptation a. agouzal [1] a. danilov [2] k. lipnikov [3], Computational Mathematics and Mathematical Physics 50 (2010) 139–156.

[13] K. E. Jensen, G. Gorman, Details of tetrahedral anisotropic mesh adaptation, Computer Physics Communications 201 (2016) 135–143.

[14] G. Rokos, G. J. Gorman, K. E. Jensen, P. H. Kelly, Thread parallelism for highly irregular computation in anisotropic mesh adaptation, in: Proceedings of the 3rd International Conference on Exascale Applications and Software, University of Edinburgh, 2015, pp. 103–108.

[15] L. Chen, P. Sun, J. Xu, Optimal anisotropic meshes for minimizing interpolation errors in $\mathcal{L}^p$-norm, Mathematics of Computation 76 (2007) 179–204.

[16] A. A. Groenwold, L. Etman, A quadratic approximation for structural topology optimization, International Journal for Numerical Methods in Engineering 82 (2010) 505–524.

[17] O. Sigmund, Morphology-based black and white filters for topology optimization, Structural and Multidisciplinary Optimization 33 (2007) 401–424.

[18] B. S. Lazarov, O. Sigmund, Filters in topology optimization based on helmholtz-type differential equations, International Journal for Numerical Methods in Engineering 86 (2011) 765–781.

[19] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, O. Sigmund, Efficient topology optimization in matlab using 88 lines of code, Structural and Multidisciplinary Optimization 43 (2011) 1–16.

[20] S. Wang, E. d. Sturler, G. H. Paulino, Large-scale topology optimization using preconditioned krylov subspace methods with recycling, International Journal for Numerical Methods in Engineering 69 (2007) 2441–2468.

[21] T. Borrvall, J. Petersson, Large-scale topology optimization in 3d using parallel computing, Computer methods in applied mechanics and engineering 190 (2001) 6201–6229.

[22] O. Amir, M. Stolpe, O. Sigmund, Efficient use of iterative solvers in nested topology optimization, Structural and Multidisciplinary Optimization 42 (2010) 55–72.

[23] G.-T. Bercea, A. T. McRae, D. A. Ham, L. Mitchell, F. Rathgeber, L. Nardi, F. Luporini, P. H. Kelly, A structure-exploiting numbering algorithm for finite elements on extruded meshes, and its performance evaluation in firedrake, arXiv preprint arXiv:1604.05937 (2016).