Structural and Multidisciplinary Optimization (2021) 63:2065–2097
https://doi.org/10.1007/s00158-020-02760-8

EDUCATIONAL PAPER

PolyStress: a Matlab implementation for local stress-constrained topology optimization using the augmented Lagrangian method

Oliver Giraldo-Londoño[1,2] · Glaucio H. Paulino[1]

Received: 28 December 2019 / Revised: 3 June 2020 / Accepted: 6 October 2020 / Published online: 10 February 2021
© Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
We present PolyStress, a Matlab implementation for topology optimization with local stress constraints considering linear and material nonlinear problems. The implementation of PolyStress is built upon PolyTop, an educational code for compliance minimization on unstructured polygonal finite elements. To solve the nonlinear elasticity problem, we implement a Newton-Raphson scheme, which can handle nonlinear material models with a given strain energy density function. To solve the stress-constrained problem, we adopt a scheme based on the augmented Lagrangian method, which treats the problem consistently with the local definition of stress without employing traditional constraint aggregation techniques. The paper discusses several theoretical aspects of the stress-constrained problem, including details of the augmented Lagrangian-based approach implemented herein. In addition, the paper presents details of the Matlab implementation of PolyStress, which is provided as electronic supplementary material. We present several numerical examples to demonstrate the capabilities of PolyStress to solve stress-constrained topology optimization problems and to illustrate its modularity to accommodate any nonlinear material model. Six appendices supplement the paper. In particular, the first appendix presents a library of benchmark examples, which are described in detail and can be explored beyond the scope of the present work.

Keywords Aggregation-free approach · Local stress constraints · Topology optimization · Augmented Lagrangian · Matlab

1 Introduction

Nineteen Eighty-Eight: that is when the landmark paper by Bendsøe and Kikuchi (1988) was published, which led to the creation of the modern framework for the field of topology optimization. Moreover, later on, Prof. Bendsøe wrote THE book (Bendsøe 1995), which served as an invitation to new researchers to the field by providing a unified presentation of methods for the design of topology, shape and material for continuum and discrete structures. As a result, THE book had a major impact in promoting and advancing the field. Inspired by Bendsøe's open scientific spirit, we hope that this paper and its associated software (PolyStress) will serve as an invitation for other researchers to investigate stress constrained topology optimization by means of a local approach, consistent with continuum mechanics.

This paper belongs to a series of educational computer codes written in Matlab for topology optimization on unstructured polygonal finite element meshes (Talischi et al. 2012a; 2012b; Pereira et al. 2016; Sanders et al. 2018). The first paper in the series is PolyTop (Talischi et al. 2012b), which was designed to solve compliance minimization problems on unstructured polygonal finite elements. The Matlab code developed for the polygonal mesh generation was presented in a companion code called PolyMesher (Talischi et al. 2012a). Owing to the modular

Responsible Editor: Ole Sigmund

Dedicated to Prof. Martin Philip Bendsøe (December 29, 1955 - present)

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00158-020-02760-8) contains supplementary material, which is available to authorized users.

✉ Glaucio H. Paulino
paulino@gatech.edu

[1] School of Civil and Environmental Engineering, Georgia Institute of Technology, 790 Atlantic Drive, Atlanta, GA 30332, USA

[2] Department of Civil and Environmental Engineering, University of Missouri, Columbia, MO, 65211, USA

structure of PolyTop, it is possible to modify either the topology optimization or the analysis routines to solve a variety of topology optimization problems. One example is PolyFluid (Pereira et al. 2016), in which a few modifications to the analysis routine led to a topology optimization code for power dissipation in Stokes flow. Another example is PolyMat (Sanders et al. 2018), which was conceived by modifying PolyTop to solve compliance minimization problems for multi-material structures with possibly many volume constraints.

In this paper, we use the general structure of PolyTop in order to develop PolyStress, a Matlab code for topology optimization problems with local stress constraints considering material nonlinearities. To solve the stress-constrained problem, we exploit the modular structure of PolyTop and modify the optimization routine in order to implement the augmented Lagrangian (AL) method (Bertsekas 1999; Nocedal and Wright 2006). This method allows us to solve the stress-constrained problem consistently with the local definition of stress with no need to use constraint aggregation techniques. Additionally, we replace the standard finite element (FE) analysis routine by a Newton-Raphson scheme, which we use to solve for the state equations considering material nonlinearities. We write the nonlinear FE code in a modular way, such that any material model can be incorporated into it.

This paper is motivated by the pioneering contributions of Prof. Martin P. Bendsøe to the field of computational design optimization and applied mathematics (Bendsøe 1989). For example, the paper by Duysinx and Bendsøe (1998) maintains "the local nature of constraints," which is the consistent approach followed in this work.

The remainder of this paper is organized as follows. Section 2 presents the stress-constrained topology optimization problem in a continuum setting and the discretization of the problem is discussed in Section 3. Details on the AL method and, in particular the AL-based formulation implemented in this study, are introduced in Section 4, followed by a discussion on the sensitivity analysis in Section 5. We discuss the Matlab implementation of PolyStress in Section 6, followed by several examples in Section 7. We finalize the paper with some concluding remarks in Section 8. Afterwards, we provide several appendices, such that the first provides a library of benchmark examples and the rest focus on the key parts of the Matlab code.

2 Topology optimization problem in a continuum setting

In this section, we introduce the formulation for topology optimization with local stress constraints in a continuum setting. First, we describe the classical continuum topology optimization problem for general objective and constraint functions. Afterwards, still in a continuum setting, we focus on the stress-constrained problem and introduce both the problem statement and the stress constraint definition that we discretize in the subsequent section to solve stress-constrained topology optimization problems. We conclude the section with a discussion on the definition for stress constraints that we adopt here, which we refer to as the *polynomial vanishing constraint*.

2.1 The general optimization problem

In a general (two-dimensional) topology optimization problem, we aim to find the shape, $\omega \subseteq \mathbb{R}^2$, of a structure, such that a given objective function, $f(\omega, \mathbf{u}_\omega)$, is minimized and some constraints, $g_j(\omega, \mathbf{u}_\omega) \leq 0, \ \ j = 1, \ldots, K$, are satisfied. The shape, ω, is usually defined on an extended domain, Ω, such that $\omega \subseteq \Omega \subseteq \mathbb{R}^2$ (see Fig. 1). In general, both the objective function and constraints depend on the shape of the structure, ω, and on the solution, $\mathbf{u}_\omega$, of a boundary value problem. Mathematically, such optimization statement is written as:

$$\begin{aligned} &\inf_{\omega \in \mathcal{O}} \ f(\omega, \mathbf{u}_\omega) \\ &\text{s.t.} \ \ g_j(\omega, \mathbf{u}_\omega) \leq 0, \quad j = 1, \ldots, K, \end{aligned} \tag{1}$$

where $\mathcal{O}$ represents the space of admissible shapes and, for the purpose of this discussion, $\mathbf{u}_\omega \in \mathcal{V}_\omega$ satisfies the variational problem of nonlinear elasticity:

$$\begin{aligned} &\mathbf{u}_\omega = \inf_{\mathbf{u}} \Pi_\omega(\mathbf{u}), \ \ \text{with} \\ &\Pi_\omega(\mathbf{u}) = \int_\omega W_0(\mathbf{u}, \mathbf{x}) d\mathbf{x} - \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{u} dS, \end{aligned} \tag{2}$$

where

$$\mathcal{V}_\omega = \left\{ \mathbf{u} \in H^1(\omega, \mathbb{R}^2) \colon \mathbf{u}|_{\partial\omega \cap \Gamma_D} = \mathbf{0} \right\} \tag{3}$$

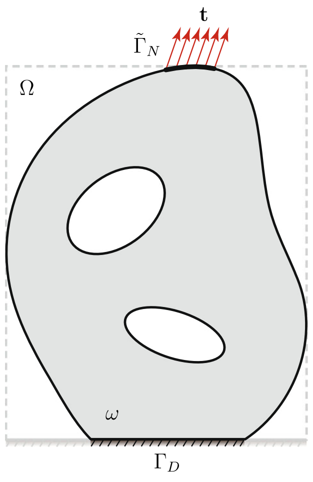


Fig. 1 Extended design domain and boundary conditions (adapted from Talischi et al. (2012b))

is the space of admissible displacements, $\Pi_\omega(\mathbf{u})$ is the total potential energy of the system, $W_0(\mathbf{u}, \mathbf{x})$ is the strain energy density of the solid material from which the shape ω is made, Γ_D and Γ_N form a partition of $\partial\Omega$, and $\mathbf{t}$ are the non-zero tractions applied on $\tilde{\Gamma}_N \subseteq \Gamma_N$.

In order to use typical discretization techniques and optimization algorithms to solve the optimization problem (1) together with the variational problem (2), it is convenient to recast (2) on Ω by introducing a characteristic function χ_ω associated with ω, as follows:[1]

$$\mathbf{u}_\omega = \inf_{\mathbf{u}} \Pi(\mathbf{u}), \quad \text{with}$$
$$\Pi(\mathbf{u}) = \int_\Omega \chi_\omega W_0(\mathbf{u}, \mathbf{x}) d\mathbf{x} - \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{u} dS, \quad \mathbf{u}_\omega \in \mathcal{V}, \tag{4}$$

where the set of admissible displacements, $\mathcal{V}$, is given by

$$\mathcal{V} = \left\{\mathbf{u} \in H^1(\Omega, \mathbb{R}^2) \colon \mathbf{u}|_{\Gamma_D} = \mathbf{0}\right\}, \tag{5}$$

which, unlike (3), is independent of ω. This way of writing the problem allows to define the shape ω on an admissible space, $\mathcal{A}_\mathcal{O} = \{\chi_\omega \colon \omega \in \mathcal{O}\}$.

Because the admissible space given by $\mathcal{A}_\mathcal{O}$ is not a vector space, the solution of the optimization statement (1) becomes an integer programming problem, which is computationally too expensive to solve in practice. As a means of solving the optimization problem, a continuous parametrization, $\rho \in [0,\ 1]$, of the shape is introduced. We interpret ρ as a density function that can take values in the interval $[0,\ 1]$. To recover the binary nature of the optimization problem, it is typical to use a penalty function such as the solid isotropic material with penalization (SIMP) (Bendsøe 1989; Zhou and Rozvany 1991; Rozvany et al. 1992). Using a continuous parametrization based on the SIMP method, the boundary value problem (4) takes the form:

$$\mathbf{u} = \inf_{\mathbf{u}} \Pi(\rho, \mathbf{u}), \quad \text{with}$$
$$\Pi(\rho, \mathbf{u}) = \int_\Omega [\epsilon + (1-\epsilon)\rho^p] W_0(\mathbf{u}, \mathbf{x}) d\mathbf{x} - \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{u} dS, \quad \mathbf{u} \in \mathcal{V}, \tag{6}$$

where $p > 1$ is a penalization factor.

The direct use of a continuous parametrization, ρ, to solve (1) does not render it a well-posed problem. To enforce its well-posedness, we indirectly impose regularity on the space of admissible designs by means of a regularization map $\mathcal{P}_F$, such that $\rho = \mathcal{P}_F(\eta)$, in which η is the design function. Using this technique allows ρ to inherit the smoothness characteristics of the kernel, F, used to define $\mathcal{P}_F$. In particular, the regularization map is defined by a convolution of the design function, η, with a smooth kernel, F, and it is obtained as follows (Bourdin 2001; Borrvall and Petersson 2001):

$$\mathcal{P}_F(\eta)(\mathbf{x}) = \int_\Omega F(\mathbf{x}, \bar{\mathbf{x}}) \eta(\bar{\mathbf{x}}) d\bar{\mathbf{x}}. \tag{7}$$

Here, we adopt the nonlinear kernel of radius R:

$$F(\mathbf{x}, \bar{\mathbf{x}}) = c(\mathbf{x}) \max\left(1 - \frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{R},\ 0\right)^q, \tag{8}$$

where $c(\mathbf{x})$ is a normalization coefficient that is chosen such that:

$$\int_\Omega F(\mathbf{x}, \bar{\mathbf{x}}) d\bar{\mathbf{x}} = 1. \tag{9}$$

In (8), $q \geq 1$ is a filter exponent that, if chosen equal to 1, leads to the traditional linear hat kernel.

As discussed by Talischi et al. (2012b), we can apply other layout or manufacturing constraints on the admissible shapes (e.g., extrusion, pattern repetition, or gradation) by introducing an operator $\mathcal{P}_s(\eta(\mathbf{x}))$ defined on Ω that maps function $\eta(\mathbf{x})$ according to the desired behavior. Here, we focus on using $\mathcal{P}_s(\eta(\mathbf{x}))$ to impose symmetries to the set of admissible shapes. For example, if we want to impose symmetry with respect to the x_1-axis, $\mathcal{P}_s(\eta(\mathbf{x}))$ takes the form:

$$\mathcal{P}_s(\eta(\mathbf{x})) = \eta(x_1, |x_2|) \tag{10}$$

and if we want to impose symmetry with respect to the x_2-axis, $\mathcal{P}_s(\eta(\mathbf{x}))$ takes the form:

$$\mathcal{P}_s(\eta(\mathbf{x})) = \eta(|x_1|, x_2). \tag{11}$$

When imposing symmetries, we need to combine both symmetry and filtering approaches and define the space of admissible density fields so that $\rho \in \mathcal{A}$, in which:

$$\mathcal{A} = \left\{\mathcal{P}(\eta) : \eta \in L^\infty(\Omega; [0, 1])\right\}, \quad \text{with} \quad \mathcal{P}(\eta) = (\mathcal{P}_F \circ \mathcal{P}_s)(\eta). \tag{12}$$

As we discuss in Appendix B, we modify the subroutine `PolyFilter` used in `PolyTop` to compute the filter operator and add the ability to impose symmetries about either the x_1-axis, the x_2-axis, or both.

In summary, a topology optimization problem written in terms of a continuous parametrization, ρ, (i.e., in terms of a density field), has the following general form:

$$\begin{aligned} &\inf_{\rho \in \mathcal{A}} \ f(\rho, \mathbf{u}) \\ &\text{s.t.} \ \ g_j(\rho, \mathbf{u}) \leq 0, \quad j = 1, \ldots, K, \end{aligned} \tag{13}$$

[1]The characteristic function, χ_ω, is defined such that $\chi_\omega = 1$ if $\mathbf{x} \in \omega$ and $\chi_\omega = 0$ otherwise. In practice, however, we use an Ersatz material model to guarantee the existence and uniqueness of the solution of the boundary value problem. That is, in regions where $\chi_\omega = 0$, we use $\epsilon + (1 - \epsilon)\chi_\omega$, in which $\epsilon \ll 1$ is the Ersatz parameter.

where the space of admissible density fields is defined in (12) and the displacement field, $\mathbf{u}$, solves the variational problem:

$$\mathbf{u} = \inf_{\mathbf{u}} \Pi(\rho, \mathbf{u}), \quad \text{with}$$
$$\Pi(\rho, \mathbf{u}) = \int_{\Omega} m_E(\rho) W_0(\mathbf{u}, \mathbf{x}) \mathrm{d}\mathbf{x} - \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{u} \mathrm{d}S, \quad \mathbf{u} \in \mathcal{V}, \quad (14)$$

where $m_E(\rho)$ is a material interpolation function used to compute the stiffness at a point based on the density at that point (e.g., $m_E(\rho) = \epsilon + (1 - \epsilon)\rho^p$ when SIMP is used). In a similar way as $m_E(\rho)$ is used to compute the stiffness at a point, different interpolation functions can be used for different physical quantities. For instance, if the physical quantity of interest is the volume fraction at a point, we can define a volume interpolation function of the form $m_V(\rho) = \rho$ to determine the volume fraction at a point based on the density at that point.[2] The volume interpolation function can be used, for instance, to determine the volume fraction of a structure defined on Ω:

$$f(\rho) = \frac{1}{|\Omega|} \int_{\Omega} m_V(\rho) \mathrm{d}\mathbf{x}. \quad (15)$$

Function $f(\rho)$ measures the ratio of the volume of a structure with respect to the volume, $|\Omega|$, of the entire domain in which the structure is defined. As we shall discuss later, that is precisely the objective function that we use in our mass minimization, topology optimization problem with local stress constraints.

2.2 The stress-constrained problem

A typical stress-constrained topology optimization problem aims to find the lightest structure which can withstand the applied loads, without material failure at any point of the domain. To limit the stress at points $\mathbf{x}_j \in \Omega$, we impose material failure constraints of the form $g_j(\rho, \mathbf{u}) \leq 0$, $j = 1, \ldots, K$. Thus, in a continuum setting, this type of optimization problem is expressed as:

$$\begin{aligned}
&\inf_{\rho \in \mathcal{A}} \ f(\rho) = \frac{1}{|\Omega|} \int_{\Omega} m_V(\rho) \mathrm{d}\mathbf{x} \\
&\text{s.t. } g_j(\rho, \mathbf{u}) \leq 0, \quad j = 1, \ldots, K \\
&\text{with: } \mathbf{u} = \inf_{\mathbf{u}} \Pi(\rho, \mathbf{u}), \quad \mathbf{u} \in \mathcal{V} \\
&\qquad \Pi(\rho, \mathbf{u}) = \int_{\Omega} m_E(\rho) W_0(\mathbf{u}, \mathbf{x}) \mathrm{d}\mathbf{x} - \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{u} \mathrm{d}S.
\end{aligned} \quad (16)$$

The objective function, $f(\rho)$, represents the mass ratio (volume fraction) of the structure, which is defined in terms of a volume interpolation function, $m_V(\rho)$. In order to promote black-and-white designs, we adopt a volume interpolation function based on the threshold projection function (Wang et al. 2011):

$$m_V(\rho) = \frac{\tanh(\beta\eta) + \tanh(\beta(\rho - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))}. \quad (17)$$

In addition, we adopt a material interpolation function of the form:[3]

$$m_E(\rho) = \begin{cases} \epsilon + (1 - \epsilon)[h(\rho)]^p & \text{(SIMP)} \\ \epsilon + (1 - \epsilon)\dfrac{h(\rho)}{1 + p_0[1 - h(\rho)]} & \text{(RAMP)}, \end{cases} \quad (18)$$

where $h(\rho) = m_V(\rho)$ is the volume fraction at a given point $\mathbf{x} \in \Omega$.

2.3 Polynomial vanishing constraints

In theory, to prevent material failure from occurring at any point $\mathbf{x}_j \in \Omega$, we should impose stress constraints pointwise (i.e., $g_j(\rho, \mathbf{u})$, $\forall \mathbf{x}_j \in \Omega$), which comes from the local definition of stress in classical continuum mechanics. In practice, however, we choose a finite number of evaluation points, K, such that the number of stress constraints becomes finite. For instance, in a discretized setting, we evaluate the stress constraints at the centroid of each finite element, and thus, we choose $K = N$, in which N is the number of elements in the finite element mesh. The stress constraints, $g_j(\rho, \mathbf{u})$, adopted here, which we refer to as *polynomial vanishing constraints* (Giraldo-Londoño and Paulino 2020), are a variation of the traditional vanishing constraints[4] (Cheng and Jiang 1992) and are defined as:

$$g_j(\rho, \mathbf{u}) = m_E(\rho)\Lambda_j(\Lambda_j^2 + 1), \quad \text{with } \Lambda_j = \sigma_j^v/\sigma_{\text{lim}} - 1, \quad (19)$$

where σ_j^v is the von Mises stress measured at evaluation point $\mathbf{x}_j \in \Omega$ and σ_{lim} is the stress limit of the material. The von Mises stress used to define $g_j(\rho, \mathbf{u})$ is based on the Cauchy stress tensor computed from the strain energy density of the solid material:

$$\boldsymbol{\sigma} = \frac{\partial W_0}{\partial \boldsymbol{\varepsilon}}, \quad (20)$$

where W_0 is the stored energy density function of the solid material and $\boldsymbol{\varepsilon}$ is the infinitesimal strain tensor.

As discussed by Giraldo-Londoño and Paulino (2020), the benefits of using the polynomial vanishing constraint are twofold. First, when $\sigma_j^v/\sigma_{\text{lim}} \gg 1$ (i.e., the constraint

[2]As we will discuss later, we adopt a volume interpolation function based on a threshold projection function (Wang et al. 2011), which allows us obtain clear black-and-white designs.

[3]A similar approach is employed by Talischi et al. (2012b) when considering the smooth Heaviside function by Guest et al. (2004).

[4]In the context of density-based topology optimization, the traditional vanishing constraints can be written as $g_j(\rho, \mathbf{u}) = \rho\Lambda_j$, with $\Lambda_j = \sigma_j^v/\sigma_{\text{lim}} - 1$

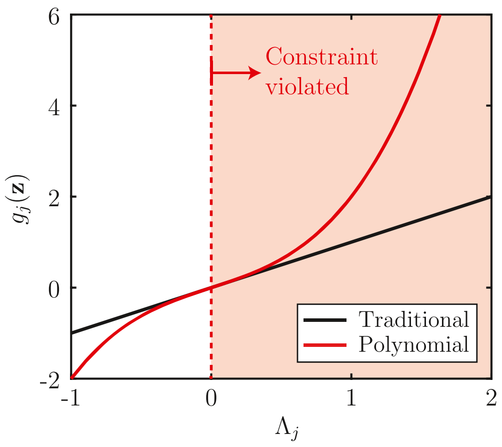


Fig. 2 Comparison between the traditional vanishing constraint (Cheng and Jiang 1992) and the polynomial vanishing constraint (Giraldo-Londoño and Paulino 2020) as a function of Λ_j. Values of $\Lambda_j > 0$ indicate stress constraint violation

is highly violated), constraint g_j scales with $(\sigma_j^v/\sigma_{\lim} - 1)^3$, thus driving the optimizer to a solution with lower overall stress. Similarly, when $\sigma_j^v/\sigma_{\lim} \to 1$ (i.e., the constraint is close to being satisfied), constraint g_j scales with $(\sigma_j^v/\sigma_{\lim} - 1)$, thus preserving the characteristics of the traditional vanishing constraint (Cheng and Jiang 1992). To emphasize these points, Fig. 2 compares the traditional vanishing constraint and the polynomial vanishing constraint as a function of Λ_j. The figure shows that the polynomial vanishing constraint penalizes constraint violation more severely than the traditional vanishing constraint, but behaves similarly to the traditional vanishing constraint when $\Lambda_j \approx 0$ (i.e., when $\sigma_j^v/\sigma_{\lim} \to 1$).

The main difference between the polynomial vanishing constraint (19) and the traditional vanishing constraint is the presence of the term Λ_j^3. Although this cubic term increases the stress constraint nonlinearity, we found no adverse effect related to this nonlinearity when solving all problems presented herein. In fact, we found that a nonlinear penalization of stress violation is effective in driving the solution toward a state of overall lower stress at a faster pace than that achieved using a linear penalization of stress violation, as it is the case for the traditional vanishing constraint. Although not pursued in the present work, other nonlinear penalization functions can be explored in lieu of the cubic function used here.[5]

[5]For a reader interested in exploring a different nonlinear term (e.g., using Λ_j^4 instead of Λ_j^3 in (19)), the function `PenalFnc` located inside the `PolyStress.m` file must be modified. An example of a similar modification is provided later.

3 Discretization

As a means to solve the stress-constrained topology optimization problem (16) numerically, we need to discretize the design domain, which allows for the discretization of the design space, $\mathcal{A}$, and of the displacement field, $\mathcal{V}$. Below, we provide specific details on the discretization of these fields and finalize with an explicit representation of the discretized topology optimization problem with local stress constraints that will lead to the implementation of `PolyStress`.

3.1 Discretized design space and filter operator

We discretize the design domain, Ω, using a fixed partition, $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$, corresponding to a characteristic mesh size h, such that $\Omega_k \cap \Omega_\ell = \emptyset, \quad \forall k \neq \ell$ and $\cup_{\ell=1}^N \overline{\Omega}_\ell = \overline{\Omega}$. We use partition $\mathcal{T}_h$ to define a piecewise constant discretization of the design space, $\mathcal{A}$:

$$\mathcal{A}_h = \left\{ \mathcal{P}(\eta_h) : 0 \le \eta_h \le 1, \ \eta|_{\Omega_\ell} = \text{const} \ \forall \ell \right\}. \tag{21}$$

The discretized design function, η_h, is written as:

$$\eta_h = \sum_{\ell=1}^{N} z_\ell \chi_{\Omega_\ell}(\mathbf{x}), \tag{22}$$

where χ_{Ω_ℓ} is the value of the characteristic function for element Ω_ℓ and $\mathbf{z} = \{z_\ell\}_{\ell=1}^N$ is the vector of design variables.

In our implementation, the density field ρ_h is replaced by another field $\tilde{\rho}_h$ that is also constant over each finite element:

$$\tilde{\rho}_h(\mathbf{x}) = \sum_{\ell=1}^{N} y_\ell \chi_{\Omega_\ell}(\mathbf{x}), \tag{23}$$

where y_ℓ are elemental values, which we define as the value of ρ_h at the centroid, $\mathbf{x}_\ell^*$, of element ℓ, i.e., $y_\ell = \rho_h(\mathbf{x}_\ell^*)$. As discussed by Talischi et al. (2012b), this definition leads to a discretization of the mapping $\mathcal{P}$ that we use to relate the vectors of elemental values, $\mathbf{y}$, and design variables, $\mathbf{z}$, as:

$$\mathbf{y} = \mathbf{P}\mathbf{z}, \tag{24}$$

where $\mathbf{P}$ is the filter matrix, which is interpreted as a discrete counterpart of the mapping, $\mathcal{P}$. For the case in which no symmetries are imposed,

$$P_{\ell k} = \int_{\Omega_k} F(\mathbf{x}_\ell^*, \bar{\mathbf{x}}) \mathrm{d}\bar{\mathbf{x}} \tag{25}$$

provides the components of the filter matrix, $\mathbf{P}$. Using the nonlinear kernel (8), we obtain the filter matrix

$$P_{\ell k} = \frac{w_{\ell k} v_k}{\sum\limits_{j=1}^{N} w_{\ell j} v_j}, \tag{26}$$

with

$$w_{\ell k} = \max\left(0, 1 - \frac{\|\mathbf{x}_\ell - \mathbf{x}_k\|_2}{R}\right)^q, \tag{27}$$

where R is the filter radius, $\|\mathbf{x}_\ell - \mathbf{x}_k\|_2$ is the distance between the centroids, $\mathbf{x}_\ell$ and $\mathbf{x}_k$, of elements ℓ and k, respectively, and q the filter exponent.

3.2 Discretized displacement field and solution of state equations

Now, to solve the variational problem (14), we discretize the displacement field, $\mathcal{V}$, on the same partition $\mathcal{T}_h$ defined previously. The discretized variational problem aims to find $\mathbf{u}_h \in \mathcal{V}_h = \text{span}\{\mathbf{N}_i\}_{i=1}^M$ such that:

$$\mathbf{u}_h = \inf_{\mathbf{u}} \Pi\,(\rho, \mathbf{u}), \text{ with}$$

$$\Pi(\rho, \mathbf{u}) = \int_\Omega m_E(\rho_h) W_0(\mathbf{u}, \mathbf{x})\mathrm{d}\mathbf{x} - \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{u}\mathrm{d}S, \tag{28}$$

where $\{\mathbf{N}_i\}_{i=1}^M$ forms the basis of $\mathcal{V}_h$ and M is the number of displacement degrees of freedom. Using a standard finite element procedure, we approximate $\mathbf{u}_h$ using shape functions $\mathbf{N}_i$:

$$\mathbf{u}_h = \sum_{i=1}^M U_i \mathbf{N}_i(\mathbf{x}). \tag{29}$$

This way, the discretized variational problem (28) takes the form:

$$\mathbf{U} = \arg\min_{\mathbf{U}} \Pi\,(\mathbf{z}, \mathbf{U}), \quad \text{with}$$

$$\Pi(\mathbf{z}, \mathbf{U}) = \sum_{\ell=1}^N \int_{\Omega_\ell} m_E(y_\ell) W_0(\mathbf{u}_\ell, \mathbf{x})\mathrm{d}\mathbf{x} - \mathbf{F}_{ext}\cdot\mathbf{U}, \tag{30}$$

where $\mathbf{U} = \{U_i\}_{i=1}^M$ is the vector of nodal displacements,

$$(\mathbf{F}_{ext})_i = \int_{\tilde{\Gamma}_N} \mathbf{t}\cdot\mathbf{N}_i\mathrm{d}S \tag{31}$$

are the design independent external nodal forces, and $\mathbf{u}_\ell$ is the vector of nodal displacements for element Ω_ℓ. The equilibrium condition satisfying the discrete minimization problem (30) is:

$$\mathbf{R} = \frac{\partial\Pi}{\partial\mathbf{U}} = \mathbf{F}_{int} - \mathbf{F}_{ext} = \mathbf{0}, \tag{32}$$

where

$$(\mathbf{F}_{int})_i = \int_\Omega m_E(\rho_h)\boldsymbol{\sigma}\cdot\nabla\mathbf{N}_i\mathrm{d}\mathbf{x} \tag{33}$$

$$= \sum_{\ell=1}^N \int_{\Omega_\ell} m_E(y_\ell)\boldsymbol{\sigma}\cdot\nabla\mathbf{N}_i\mathrm{d}\mathbf{x} \tag{34}$$

are the internal nodal forces, which are given in terms of the Cauchy stress vector $\boldsymbol{\sigma}$. The system of nonlinear equations given by the equilibrium condition (32) is solved iteratively using a Newton-Raphson scheme, for which we obtain the consistent tangent matrix:

$$(\mathbf{K}_T)_{ij} = \int_\Omega m_E(\rho_h)\mathbf{C}_T\nabla\mathbf{N}_i : \nabla\mathbf{N}_j\mathrm{d}\mathbf{x} \tag{35}$$

$$= \sum_{\ell=1}^N \int_{\Omega_\ell} m_E(y_\ell)\mathbf{C}_T\nabla\mathbf{N}_i : \nabla\mathbf{N}_j\mathrm{d}\mathbf{x}, \tag{36}$$

defined in terms of the material tangent moduli matrix, $\mathbf{C}_T$.

3.3 Discretized topology optimization problem with stress constraints

Having discretized the design space, $\mathcal{A}$, and the displacement field, $\mathcal{V}$, on Ω, the final discrete problem for topology optimization with local stress constraints becomes:

$$\begin{aligned}
\min_{\mathbf{z}\in[0,1]^N} \quad & f(\mathbf{z}) = \frac{\mathbf{A}^T m_V(\mathbf{y})}{\mathbf{A}^T\mathbf{1}} \\
\text{s.t.} \quad & g_j(\mathbf{z}, \mathbf{U}) = m_E(y_j)\Lambda_j(\Lambda_j^2 + 1) \leq 0, \\
& \qquad j = 1, \dots, N \\
\text{with:} \quad & \Lambda_j = \sigma_j^v/\sigma_{\lim} - 1 \\
& \mathbf{U} = \arg\min_{\mathbf{U}} \Pi(\mathbf{z}, \mathbf{U}) = \\
& \sum_{\ell=1}^N \int_{\Omega_\ell} m_E(y_\ell) W_0(\mathbf{u}_\ell, \mathbf{x})\mathrm{d}\mathbf{x} - \mathbf{F}_{ext}\cdot\mathbf{U}
\end{aligned} \tag{37}$$

where $\mathbf{A} = \{|\Omega_\ell|\}_{\ell=1}^N$ is the vector of element areas and σ_j^v is the von Mises stress evaluated at the centroid, $\mathbf{x}_j^*$, of element Ω_j. The volume interpolation function, $m_V(\cdot)$, and the material interpolation function, $m_E(\cdot)$, are evaluated according to (17) and (18), respectively. The state variables (i.e., the nodal displacements), $\mathbf{U}$, are found via the finite element method, as discussed previously. Note that the topology optimization problem stated above imposes one stress constraint per element, which is adequate when we consider only one load case.[6]

4 Augmented Lagrangian framework

One of the main challenges in stress-constrained topology optimization is the large number of stress constraints that must be imposed in order to prevent material failure at every evaluation point. In the discretized optimization statement (37), the number of stress constraints, N, is

[6]When multiple load cases are considered, the number of stress constraints becomes $N_c = mN$, in which m is the number of load cases. The current implementation of `PolyStress` considers one load case only, yet the interested reader can modify the code accordingly to account for multiple load cases (details on the implementation for multiple load cases can be found in a study by Senhora et al. 2020).

as large as the number of elements in the FE mesh. To reduce the computational cost associated with handling so many constraints, the most commonly used approach is to aggregate the local stress constraints into a single global constraint or a few clustered constraints (e.g., see Yang and Chen 1996; Luo et al. 2013; De Leon et al. 2015; Kiyono et al. 2016; Lee et al. 2016; Lian et al. 2017; Liu et al. 2018; Xia et al. 2018; Fan et al. 2019; Le et al. 2010; Lee et al. 2012; Holmberg et al. 2013b). The final designs obtained using these clustering techniques tend to depend on the number of clusters, the number of elements (stress evaluation points) used to define each cluster, and the norm function used to estimate the maximum stress in each cluster—e.g., the p-norm function (Park 1995) or the KS function (Kreisselmeier and Steinhauser 1979). Not only does the design depend on these choices, but depending on the norm function, the stress limit is not satisfied everywhere. Based on these observations, one could argue that clustering (or aggregation) approaches are not suitable for handling realistic design problems.

Instead of the aforementioned aggregation techniques, an attractive approach to solve the discretized optimization statement (37) while satisfying the stress constraints locally is the AL method (Bertsekas 1999; Nocedal and Wright 2006). Several studies have used the AL method to solve stress-constrained topology optimization problems (Pereira et al. 2004; Fancello 2006; Emmendoerfer and Fancello 2014; 2016; Emmendoerfer et al. 2019; da Silva et al. 2018). In the AL method, the solution of a constrained optimization problem such as (37) is obtained as the solution of a series of unconstrained optimization problems, each aiming to minimize the AL function, $\mathcal{L}_\mu(\mathbf{z}, \boldsymbol{\lambda})$, of the original optimization problem. Specifically, at the k-th step of the AL method, one solves the unconstrained optimization problem:[7]

$$\min_{\mathbf{z}\in[0,1]^N} \ \mathcal{L}_{\mu^{(k)}}(\mathbf{z}, \boldsymbol{\lambda}^{(k)}) = f(\mathbf{z}) + P^{(k)}(\mathbf{z}, \mathbf{U}), \tag{38}$$

where

$$P^{(k)}(\mathbf{z}, \mathbf{U}) = \sum_{j=1}^{N} \left[\lambda_j^{(k)} h_j(\mathbf{z}, \mathbf{U}) + \frac{\mu^{(k)}}{2} h_j(\mathbf{z}, \mathbf{U})^2 \right] \tag{39}$$

is the penalization term. Moreover

$$h_j(\mathbf{z}, \mathbf{U}) = \max \left[g_j(\mathbf{z}, \mathbf{U}), -\frac{\lambda_j^{(k)}}{\mu^{(k)}} \right] \tag{40}$$

are equality constraints used to define $P^{(k)}(\mathbf{z}, \mathbf{U})$, $\boldsymbol{\lambda}^{(k)} = \{\lambda_j^{(k)}\}_{j=1}^N$ is a vector of Lagrange multiplier estimators, and $\mu^{(k)} > 0$ is a quadratic penalty factor. The penalty factor is typically updated as:

$$\mu^{(k+1)} = \min \left[\alpha\mu^{(k)}, \mu_{\max} \right], \tag{41}$$

where $\alpha > 1$ is an update parameter and $\mu_{\max}$ an upper limit used to prevent numerical instabilities. The Lagrange multiplier estimators are updated as:

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z}^{(k)}, \mathbf{U}). \tag{42}$$

Without any sort of normalization, the AL method discussed above is not suitable to solve large-scale stress-constrained topology optimization problems. The reason is that, as N becomes large, the penalization term, $P^{(k)}(\mathbf{z}, \mathbf{U})$, dominates over the objective function term, $f(\mathbf{z})$, which negatively impacts the convergence of the method. As a means of alleviating this issue, Senhora et al. (2020) introduced a normalization term to $P^{(k)}(\mathbf{z}, \mathbf{U})$, so that the normalized AL sub-problem becomes:

$$\min_{\mathbf{z}\in[0,1]^N} \ J^{(k)}(\mathbf{z}, \mathbf{U}) = f(\mathbf{z}) + \frac{1}{N} P^{(k)}(\mathbf{z}, \mathbf{U}). \tag{43}$$

The only difference between the original AL sub-problem (38) and sub-problem (43) is that, in the latter, the penalization term is normalized with respect to the number of constraints, N. This normalization allows solving problems with a large number of constraints without experiencing numerical instabilities, which is why we adopt this approach to solve the stress-constrained problem (37). A similar strategy to normalize the AL function is presented by da Silva et al. (2019a), in which the objective function term is multiplied by the number of elements in the FE mesh. Similarly, to improve the robustness of the AL method, da Silva et al. (2019b) normalized both the initial and the maximum penalty terms (i.e., $\mu^{(0)}$ and $\mu_{\max}$) with respect to the number of elements, N. These studies highlight the importance of a normalization term in order to improve the robustness of the AL method.

A schematic flowchart of the AL-based topology optimization framework that we implement in this work is depicted in Fig. 3. We first use input data related to the finite element problem and the optimizer and then initialize the Lagrange multiplier estimators, $\lambda_j^{(k)}$, and the penalty term, $\mu^{(k)}$ for $k = 0$. With that information, we use the method of moving asymptotes (MMA) (Svanberg 1987) to find an approximate minimizer of the normalized AL function; that is, we find an approximate solution of the minimization problem (43). Using the solution obtained from the approximate solution of the AL problem, we update $\lambda_j^{(k)}$ and $\mu^{(k)}$ and repeat the process until convergence is achieved. In particular, we consider that the problem has converged when $\frac{1}{N}\text{sum}(|\mathbf{z}_{i+1}^{(k)} - \mathbf{z}_i^{(k)}|) < \texttt{Tol}$

[7] Although we refer to the AL sub-problem (38) as unconstrained, it is in fact an optimization problem with box constraints.

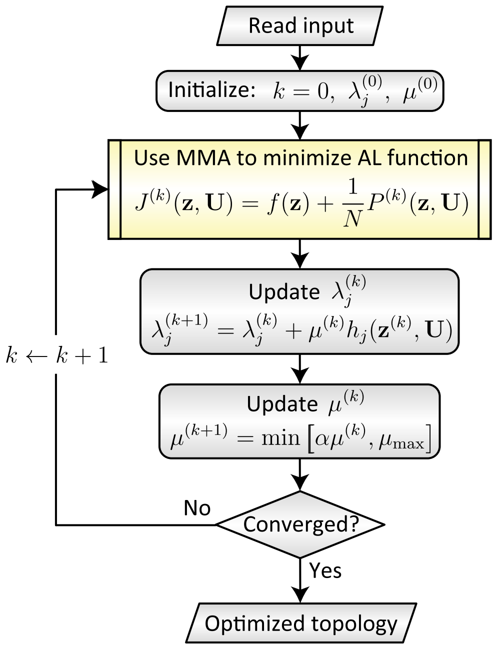

Read input
Initialize: $k=0,\ \lambda_j^{(0)},\ \mu^{(0)}$
Use MMA to minimize AL function
$J^{(k)}(\mathbf{z},\mathbf{U}) = f(\mathbf{z}) + \frac{1}{N}P^{(k)}(\mathbf{z},\mathbf{U})$
Update $\lambda_j^{(k)}$
$\lambda_j^{(k+1)} = \lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z}^{(k)},\mathbf{U})$
Update $\mu^{(k)}$
$\mu^{(k+1)} = \min\left[\alpha\mu^{(k)}, \mu_{\max}\right]$
$k \leftarrow k+1$
No
Converged?
Yes
Optimized topology

Fig. 3 Schematic flowchart of the AL-based topology optimization framework implemented in this study to solve topology optimization problems with local stress constraints

and $\max(\sigma_j^v/\sigma_{\lim}) - 1 < \texttt{TolS}$, where `Tol` and `TolS` are prescribed tolerance values for the design variable change and the stress constraints, respectively, and $\mathbf{z}_{i+1}^{(k)}$ and $\mathbf{z}_i^{(k)}$ are design variable vectors at two consecutive MMA iterations of a given AL sub-problem, k.

The values of the initial penalty term, $\mu^{(0)}$, maximum penalty term, $\mu_{\max}$, penalty term update parameter, α, and initial Lagrange multiplier estimators, $\boldsymbol{\lambda}^{(0)}$, influence the performance of the AL method. From all these terms, the latter has the least influence in the performance of the method and can be chosen as $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$. However, the parameters used to define the update of the penalty term must be obtained through an initial calibration process. First, $\mu^{(0)}$ should not be too small or else the convergence of the method becomes too slow. Second, the value of $\mu_{\max}$ should not be too large to prevent ill-conditioning. Third, the value of α is recommended to be small (e.g., $1 < \alpha < 2$), so that the $\mu^{(k)}$ grows at a moderate pace between consecutive AL sub-problems. In our experience, once these parameters have been found for a given problem (e.g., for the traditional L-bracket problem), they tend to work for a variety of other problems, as we show later in the numerical examples.

5 Sensitivity analysis

We solve the stress-constrained topology optimization problem (37) using a gradient-based optimization algorithm; thus, we require the sensitivity information of the normalized AL function in (43), for which we use the chain rule:

$$\frac{\mathrm{d}J^{(k)}}{\mathrm{d}z_e} = \sum_{\ell=1}^{N}\left(\frac{\partial E_\ell}{\partial z_e}\frac{\mathrm{d}J^{(k)}}{\mathrm{d}E_\ell} + \frac{\partial V_\ell}{\partial z_e}\frac{\mathrm{d}J^{(k)}}{\mathrm{d}V_\ell}\right) \tag{44}$$

or in vector form:

$$\frac{\mathrm{d}J^{(k)}}{\mathrm{d}\mathbf{z}} = \frac{\partial \mathbf{E}}{\partial \mathbf{z}}\frac{\mathrm{d}J^{(k)}}{\mathrm{d}\mathbf{E}} + \frac{\partial \mathbf{V}}{\partial \mathbf{z}}\frac{\mathrm{d}J^{(k)}}{\mathrm{d}\mathbf{V}}, \tag{45}$$

where $\mathbf{E} = m_E(\mathbf{y})$ and $\mathbf{V} = m_V(\mathbf{y})$ are vectors containing design-related information. Because the normalized AL function is divided into the objective function term and the penalty term, we rewrite (45) in a more convenient way as:

$$\frac{\mathrm{d}J^{(k)}}{\mathrm{d}\mathbf{z}} = \frac{\partial \mathbf{E}}{\partial \mathbf{z}}\left(\frac{\partial f}{\partial \mathbf{E}} + \frac{1}{N}\frac{\partial P^{(k)}}{\partial \mathbf{E}}\right) + \frac{\partial \mathbf{V}}{\partial \mathbf{z}}\left(\frac{\partial f}{\partial \mathbf{V}} + \frac{1}{N}\frac{\partial P^{(k)}}{\partial \mathbf{V}}\right). \tag{46}$$

Given that $\mathbf{E} = m_E(\mathbf{Pz})$ and $\mathbf{V} = m_V(\mathbf{Pz})$, the sensitivity of the material stiffness parameters, $\mathbf{E}$, and volume fractions, $\mathbf{V}$, with respect to the design variables, $\mathbf{z}$, are given by:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_E}(\mathbf{Pz}) \ \text{ and } \ \frac{\partial \mathbf{V}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_V}(\mathbf{Pz}), \tag{47}$$

respectively, where $J_{m_E} = \operatorname{diag}(m'_E(y_1), \ldots, m'_E(y_N))$ and $J_{m_V} = \operatorname{diag}(m'_V(y_1), \ldots, m'_V(y_N))$ (Talischi et al. 2012b).

Based on the optimization statement (37), the sensitivity of the objective function term is:

$$\frac{\partial f}{\partial E_\ell} = 0 \ \text{ and } \ \frac{\partial f}{\partial V_\ell} = \frac{A_\ell}{\mathbf{A}^T\mathbf{1}}. \tag{48}$$

According to (39), $P^{(k)}$ has no direct dependence on V_ℓ, and thus, its sensitivity with respect to V_ℓ is:

$$\frac{\partial P^{(k)}}{\partial V_\ell} = 0. \tag{49}$$

However, $P^{(k)}$ has a direct dependence on the element stiffness parameter, E_ℓ, through the stress constraints (cf. $(37)_2$). Thus, its sensitivity with respect to E_ℓ, is expressed as:

$$\frac{\partial P^{(k)}}{\partial E_\ell} = \sum_{j=1}^{N}[\lambda_j^{(k)} + \mu^{(k)} h_j(\mathbf{z},\mathbf{U})]\left[\frac{\partial h_j(\mathbf{z},\mathbf{U})}{\partial E_\ell} + \frac{\partial h_j(\mathbf{z},\mathbf{U})}{\partial \mathbf{U}}\cdot\frac{\partial \mathbf{U}}{\partial E_\ell}\right]. \tag{50}$$

We use the adjoint method (Bendsøe and Sigmund 2003) to avoid the expensive computation of $\partial\mathbf{U}/\partial E_\ell$, for which we use the equilibrium condition (32). The sensitivity of the residual vector, $\mathbf{R}$, at equilibrium can be written as:

$$\begin{aligned}\frac{\partial \mathbf{R}}{\partial E_\ell} &= \frac{\partial \mathbf{F}_{int}}{\partial \mathbf{U}}\cdot\frac{\partial \mathbf{U}}{\partial E_\ell} + \frac{\partial \mathbf{F}_{int}}{\partial E_\ell} - \frac{\partial \mathbf{F}_{ext}}{\partial E_\ell}\\ &= \mathbf{K}_T\frac{\partial \mathbf{U}}{\partial E_\ell} + \frac{\partial \mathbf{F}_{int}}{\partial E_\ell} = \mathbf{0},\end{aligned} \tag{51}$$

where we have assumed that the external force vector, $\mathbf{F}_{ext}$, is independent of the design variables. Now, we add the term

$\boldsymbol{\xi}^T \partial \mathbf{R}/\partial E_\ell$ to (50), in which $\boldsymbol{\xi}$ is an adjoint vector. That is, we write (50) as:

$$\frac{\partial P^{(k)}}{\partial E_\ell} = \sum_{j=1}^{N}[\lambda_j^{(k)}+\mu^{(k)}h_j(\mathbf{z},\mathbf{U})]\left[\frac{\partial h_j(\mathbf{z},\mathbf{U})}{\partial E_\ell}+\frac{\partial h_j(\mathbf{z},\mathbf{U})}{\partial \mathbf{U}}\cdot\frac{\partial \mathbf{U}}{\partial E_\ell}\right] \\ +\boldsymbol{\xi}^T\left(\mathbf{K}_T\frac{\partial \mathbf{U}}{\partial E_\ell}+\frac{\partial \mathbf{F}_{int}}{\partial E_\ell}\right) \quad (52)$$

Finally, we choose $\boldsymbol{\xi}$ such that all terms containing $\partial \mathbf{U}/\partial E_\ell$ vanish from (52), which leads to:

$$\frac{\partial P^{(k)}}{\partial E_\ell} = \sum_{j=1}^{N}[\lambda_j^{(k)}+\mu^{(k)}h_j(\mathbf{z},\mathbf{U})]\frac{\partial h_j(\mathbf{z},\mathbf{U})}{\partial E_\ell}+\boldsymbol{\xi}^T\frac{\partial \mathbf{F}_{int}}{\partial E_\ell}, \quad (53)$$

where $\boldsymbol{\xi}$ solves the adjoint problem:

$$\mathbf{K}_T\boldsymbol{\xi} = -\sum_{j=1}^{N}\left[\lambda_j^{(k)}+\mu^{(k)}h_j(\mathbf{z},\mathbf{U})\right]\frac{\partial h_j(\mathbf{z},\mathbf{U})}{\partial \mathbf{U}}. \quad (54)$$

Based on the functional form of $h_j(\mathbf{z},\mathbf{U})$ in (40), we obtain that $\partial h_j(\mathbf{z},\mathbf{U})/\partial \mathbf{U} = \mathbf{0}$ whenever $g_j(\mathbf{z}) < -\lambda_j^{(k)}/\mu^{(k)}$ and:

$$\frac{\partial h_j}{\partial \mathbf{U}} = \frac{\partial g_j}{\partial \mathbf{U}} = \frac{\partial g_j}{\partial \sigma_j^v}\frac{\partial \sigma_j^v}{\partial \boldsymbol{\sigma}}\cdot\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{U}} \quad (55)$$

otherwise. The term $\partial g_j/\partial \sigma_j^v$ is obtained directly from $(37)_2$. Now, to obtain $\partial \sigma_j^v/\partial \boldsymbol{\sigma}$, we recall that the von Mises stress is computed as:

$$\sigma_j^v = \sqrt{\boldsymbol{\sigma}^T\mathbf{V}_0\boldsymbol{\sigma}}, \quad (56)$$

where $\boldsymbol{\sigma} = [\sigma_{11}\ \sigma_{22}\ \sigma_{12}]^T$ is the vector of Cauchy stresses, in Voigt notation, obtained as $\boldsymbol{\sigma} = \partial W_0/\partial \boldsymbol{\varepsilon}$, and:

$$\mathbf{V}_0 = \begin{bmatrix} 1 & -1/2 & 0 \\ -1/2 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

Therefore, the sensitivity of the von Mises stress with respect to the Cauchy stress vector is given by:

$$\frac{\partial \sigma_j^v}{\partial \boldsymbol{\sigma}} = \frac{\mathbf{V}_0\boldsymbol{\sigma}}{\sigma_j^v}. \quad (57)$$

Finally, the term $\partial \boldsymbol{\sigma}/\partial \mathbf{U}$ is found explicitly as:

$$\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{U}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\cdot\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{U}} = \mathbf{DB} \quad (58)$$

where $\boldsymbol{\varepsilon}$ is the infinitesimal strain vector (in Voigt notation), $\mathbf{D}$ is the material tangent matrix, and $\mathbf{B}$ is the strain-displacement matrix.

6 `PolyStress` implementation in Matlab

We adopt the general structure established in `PolyTop` to implement the discrete topology optimization problem (37) and apply appropriate modifications to the FE analysis routine to solve the discrete variational problem (30) and to the optimization routine to implement the AL method. As part of the modification to the optimization routine, we implement a version of the method of moving asymptotes (MMA) (Svanberg 1987), tailored to solve unconstrained minimization problems. We provide specific details on these modifications in the subsequent sections.

6.1 Input data and `PolyScript`

As in `PolyTop`, we use a Matlab script called `PolyScript` to collect all the parameters necessary to run `PolyStress`. The input parameters required to run `PolyStress` are all placed in two `struct` arrays, `fem` and `opt`. The `fem` structure contains the information required for the FE analysis routine (e.g., mesh, loading, supports, and material properties) while the `opt` structure contains information related to the topology optimization (e.g., filter matrix, material interpolation functions, and parameters of the optimizer). Table 1 displays the fields stored in the `fem` structure array and Table 2 those stored in the `opt` structure array. The main differences between the current `fem` fields and those in `PolyTop` are related to the nonlinear FE analysis routine (e.g., `fem.MatModel`, `fem.MatParam`, `fem.tolR`, and `fem.MaxIter`) and to the evaluation of the von Mises stresses (e.g., `fem.SLim` `fem.B0`, `fem.rowD`, `fem.colD`, `fem.eDof`, and `fem.VM_Stress0`). Regarding the `opt` structure, the differences between the current `opt` fields and those in `PolyTop` are related to the optimizer (e.g., `opt.Move`, `opt.Osc`, `opt.AsymInit`, `opt.AsymInc`, and `opt.AsymDecr`, which we use for the MMA subroutine) and to the continuation of the penalization parameter of the threshold projection function (17), which are embedded in the `opt.contB` field. The `opt.contB` field contains four entries, `[BFreq,B0,Binc,Bmax]`, corresponding to the update frequency, initial value, increment value, and maximum value, respectively, of parameter β in (17). `PolyStress` uses these four entries to update parameter β as:

```
if mod(Iter,BFreq)==0; B = min(B+Binc,Bmax); end
```

Similarly to the `PolyTop` implementation, `PolyScript` calls the polygonal mesh generator `PolyMesher` (Talischi et al. 2012a) to obtain the finite element mesh, boundary conditions, and applied loads. `PolyScript` also calls the subroutine `PolyFilter` to compute the filter matrix, $\mathbf{P}$. We modify the `PolyFilter`

Table 1 List of fields in `fem` structure

`fem` field	
`fem.NNode`	Number of nodes
`fem.NElem`	Number of elements
`fem.Node`	[`NNode` $\times$ 2] array of nodes
`fem.Element`	[`NElem` $\times$ `Var`] cell array of elements
`fem.Supp`	[`NSupp` $\times$ 3] support array
`fem.Load`	[`NLoad` $\times$ 3] load array
`fem.Passive`	Array of passive elements
`fem.Thickness`	Element thickness
`fem.MatModel`	String array with the name of the material model
`fem.MatParam`	[1 $\times$ `Var`] Array of material model parameters
`fem.SLim`	Material yield stress
`fem.TolR`	Tolerance for the norm of the force residual
`fem.MaxIter`	Maximum number of Newton-Raphson iterations
`fem.MEX`	String array to use of MEX functions (`'Yes'` or `'No'`)
`fem.ElemArea`†	Array of element areas
`fem.W`†	Weights at Gauss points
`fem.dNdxi`†	Derivatives of shape functions at Gauss points
`fem.ElemNDof`†	Array showing number of DOFs of elements
`fem.k0`†	Array of local stiffness matrix entries
`fem.i`†	Index array for sparse assembly of `fem.k0`
`fem.j`†	Index array for sparse assembly of `fem.k0`
`fem.e`†	Array of element IDs corresponding to `fem.k0`
`fem.eDof`†	Index array for sparse assembly of global load vectors
`fem.DofE`†	Array of element IDs corresponding to global load vectors
`fem.iK0`†	Index array for sparse assembly of element stiffness matrices
`fem.jK0`†	Index array for sparse assembly of element stiffness matrices
`fem.Fext`†	Global external load vector
`fem.FreeDofs`†	Array of free degrees of freedom
`fem.B0`†	Strain-displacement matrices at element centroids
`fem.rowD`†	Index array for sparse assembly of element tangent matrices
`fem.colD`†	Index array for sparse assembly of element tangent matrices
`fem.U`†	Converged displacement vector at each optimization iteration
`fem.L`†	Lower triangular matrix from Cholesky factorization of stiffness matrix
`fem.s`†	Permutation vector from Cholesky factorization of stiffness matrix
`fem.f_NL`†	Force vector assuming solid elements
`fem.VM_Stress0`†	Array of von Mises stress at element centroids

All fields marked with the superscript †, if empty, are populated inside `PolyStress`

routine in `PolyTop` in order to implement the nonlinear filter (26)–(27) used in the current implementation and add a new capability to the function to impose symmetry to the density field about either the x_1-axis, the x_2-axis, or both.[8] We provide the modified `PolyFilter` routine as supplementary material. Before populating the `opt` structure, `PolyScript` calls an auxiliary function called `MatIntFnc`, which given an input vector $\mathbf{y}$ and a set of parameters, `params`,[9] it outputs the vector of stiffness, $\mathbf{E} = m_E(\mathbf{y})$, the vector of volume fractions, $\mathbf{V} = m_V(\mathbf{y})$, and their corresponding sensitivity vectors, $\partial\mathbf{E}/\partial\mathbf{y}$ and $\partial\mathbf{V}/\partial\mathbf{y}$, which as in `PolyTop`, are understood as the diagonal entries of the Jacobian matrices of $m_E(\mathbf{y})$ and $m_V(\mathbf{y})$, respectively. Because the threshold projection function (17) is not available in `PolyTop`, we modify the

[8]Appendix B provides additional details explaining our approach to impose symmetry.

[9]For the case in which we compute $m_V(\mathbf{y})$ using the threshold projection function (17) and $m_E(\mathbf{y})$ using SIMP in (18)), the set of parameters in the field `params` becomes `[p,B,eta0]`, which correspond to parameters p, β, and η, respectively.

MatIntFnc implementation accordingly, as can be seen in the supplementary material.

6.2 Computation of the AL function in PolyStress

We use the AL method to solve the topology optimization problem (37), for which we require to solve the AL sub-problem (43) at each step. The computation of the normalized AL function is divided in two parts: an objective function term, ObjectiveFnc, and a penalty term, PenalFnc. To compute the objective function term, we use a slightly modified version of the constraint function, ConstraintFnc, implemented in PolyTop. The only difference between the function in PolyTop and that in PolyStress is that the latter needs no volume fraction limit. To compute the penalty term of the normalized AL function, we require aseparate implementation, which calls two auxiliary functions: the nonlinear FEM routine, NLFEM, and the von Mises stress computation routine, von_Mises_Stress. The former is used to obtain the displacement vector, $\mathbf{U}$, while the latter is used to compute the von Mises stress at the centroid of all elements as well as their sensitivity with respect to the displacement vector. The function used to evaluate the normalized AL function and its sensitivity is called AL_Function and is provided with the PolyStress code, as shown in Appendix E.

6.3 Nonlinear FE analysis routine

We use a nonlinear FE analysis routine based on the Newton-Raphson method with a weak line search algorithm[10] to solve the discretized variational problem (30). The main portion of the nonlinear FE analysis routine is summarized in the following few lines of code:

```
U = fem.U; % Use previously converged U as initial guess
[K,~,Res,~,fem] = GlobalK(fem,U,E); %Initial stiffness mtrx. & Res vector
nRes0 = norm(fem.Fext); nRes = nRes0; % Initial norm of force residual
% NEWTON-RAPHSON ITERATIONS
Iter = 0; % Initialize Newton-Raphson iteration counter
while (nRes>fem.TolR*nRes0 && Iter<=fem.MaxIter)
  [Delta_U, L, s] = SolveLinSys(K,Res);
  [K,~,Res,nRes,fem,U] = LineSearch(U,Delta_U,nRes,-K*Res./nRes,fem,E);
  Iter = Iter+1;
end
fem.L = L; fem.s = s; % Store Cholesky decomposition information
fem.U = U; % Store converged displacement vector
```

In the routine above, the incremental displacement vector, Delta_U, is obtained using a subroutine called SolveLinSys, which uses the Cholesky decomposition of the tangent stiffness matrix, $\mathbf{K}_T$. Then, the displacement vector, U, is updated by means of a weak line search algorithm. Upon convergence, the code stores the converged displacement vector, U, in addition to the lower triangular matrix, L, and the permutation vector, s, coming from the Cholesky decomposition of K. Matrix L and vector s are stored so that we can compute the adjoint vector efficiently. The nonlinear FE analysis routine is called only when the material model is nonlinear. When dealing with a linear material, we obtain the displacement vector directly by solving the linear system, $\mathbf{K}_T\mathbf{U} = \mathbf{F}_{ext}$, where the stiffness matrix is computed based on the stiffness matrix for the solid elements, which is stored in a pre-processing stage.

As widely known, the bottleneck in any topology optimization is related to the finite element analysis, especially when the structure is made of a nonlinear material. In an attempt to improve the efficiency of the nonlinear analysis routine, the subroutine in charge of computing the global stiffness matrix is converted to a MEX function. When a user first runs PolyStress to optimize a nonlinear structure, the MEX files are compiled automatically in the subroutine called GlobalK, but they are compiled only if the user wants to use them. For example, if the user types fem.MEX='Yes' (see PolyScript file in Appendix D), the MEX functions will be generated, but if the user types fem.MEX='No', PolyStress will use traditional Matlab functions. The complete nonlinear analysis routine is provided in Appendix F.

The Newton-Raphson routine implemented in PolyStress uses only one load step to solve the variational problem of nonlinear elasticity. The implementation with a single load step and a weak line search algorithm has

[10]Our weak line search algorithm was obtained from the following reference: Ascher and Greif (2011) A first course in numerical methods. SIAM (Society for Industrial and Applied Mathematics).

Table 2 List of fields in opt structure

opt field	
opt.zMin	Lower bound for design variables
opt.zMax	Upper bound for design variables
opt.zIni	Initial array of design variables
opt.MatIntFnc	Handle to material interpolation function
opt.contB	Threshold projection continuation params.
opt.P	Matrix that maps design to element variables
opt.Tol	Convergence tolerance on design variables
opt.TolS	Convergence tolerance on stress constraints
opt.MaxIter	Maximum number of AL steps
opt.MMA_Iter	Number of MMA iterations per AL step
opt.lambda0	Initial Lagrange multiplier estimators
opt.mu0	Initial value AL penalty factor
opt.mu_max	Maximum value AL penalty factor
opt.alpha	Penalty factor update parameter
opt.Move	Allowable move step in the MMA update scheme
opt.Osc	Oscillation parameter in the MMA update scheme
opt.AsymInit	Initial asymptote parameter in the MMA update scheme
opt.AsymInc	Asymptote parameter increment in the MMA update scheme
opt.AsymDecr	Asymptote parameter decrement in the MMA update scheme

proven effective to solve all the problems presented in this manuscript.[11]

6.4 Bilinear material model

The nonlinear FE analysis routine discussed previously can accommodate any nonlinear material model, given that its stored energy function, W_0, is defined. Each nonlinear material model is called by the nonlinear FE analysis routine through a function called material_model. One of the material models implemented in the material_model routine belongs to the category of *conewise linear elastic* materials, which are a generalization of the bimodular material models (Curnier et al. 1994). To define the strain energy density, W_0, of a conewise linear elastic material, the space of infinitesimal strains, $\mathcal{E}$, is divided into a compression sub-domain, $\mathcal{E}_c$ and a tension sub-domain, $\mathcal{E}_t$, such that $\mathcal{E}_c = \{\boldsymbol{\varepsilon} \in \mathcal{E} \mid \chi(\boldsymbol{\varepsilon}) < 0\}$ and $\mathcal{E}_t = \{\boldsymbol{\varepsilon} \in \mathcal{E} \mid \chi(\boldsymbol{\varepsilon}) > 0\}$, where $\chi(\boldsymbol{\varepsilon}) = \operatorname{tr}(\boldsymbol{\varepsilon})$ is a scalar function used to ensure continuous differentiability of W_0. The stored energy density of a conewise linear elastic material with continuous Cauchy stress tensor across $\mathcal{E}$ is given by:

$$W_0(\boldsymbol{\varepsilon}) = \frac{1}{2}\lambda(\boldsymbol{\varepsilon})\operatorname{tr}^2(\boldsymbol{\varepsilon}) + \mu\operatorname{tr}(\boldsymbol{\varepsilon}^2), \tag{59}$$

[11]However, an implementation with several load steps can also be used. To do so, one would need to add an additional for loop to the nonlinear FE analysis routine including all the load steps.

where μ is the shear modulus and $\lambda(\boldsymbol{\varepsilon})$ is a Lamé parameter given by:

$$\lambda(\boldsymbol{\varepsilon}) = \begin{cases} \lambda_c & \text{if } \operatorname{tr}(\boldsymbol{\varepsilon}) < 0 \\ \lambda_t & \text{if } \operatorname{tr}(\boldsymbol{\varepsilon}) > 0. \end{cases} \tag{60}$$

The Cauchy stress and material moduli tensors are computed as:

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = \frac{\partial W_0}{\partial \boldsymbol{\varepsilon}} = \lambda(\boldsymbol{\varepsilon})\operatorname{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon} \quad \text{and} \tag{61}$$

$$\mathbf{C}_T(\boldsymbol{\varepsilon}) = \frac{\partial^2 W_0}{\partial \boldsymbol{\varepsilon}^2} = \lambda(\boldsymbol{\varepsilon})\mathbf{I}\otimes\mathbf{I} + 2\mu\mathbf{I}\,\underline{\overline{\otimes}}\,\mathbf{I},$$

respectively, where the operators $\otimes$ and $\underline{\overline{\otimes}}$ are defined such that, for any two second-order tensors, $\boldsymbol{a}$ and $\boldsymbol{b}$, $(a \otimes b)_{ijkl} = a_{ij}b_{kl}$ and $(a \,\underline{\overline{\otimes}}\, b)_{ijkl} = \frac{1}{2}(a_{ik}b_{jl} + a_{il}b_{jk})$. It is convenient to express the material model above in terms of the Young modulus in tension, E_t, and the Young modulus in compression, E_c, for which we rewrite λ_c and λ_t, as follows:

$$\lambda_c = \frac{\mu(E_c - 2\mu)}{3\mu - E_c} \quad \text{and} \quad \lambda_t = \frac{\mu(E_t - 2\mu)}{3\mu - E_t}. \tag{62}$$

According to (59)–(60), the material behaves as a linear material with properties (E_t, μ) when $\chi(\boldsymbol{\varepsilon}) = \operatorname{tr}(\boldsymbol{\varepsilon}) > 0$ (tension subdomain, $\mathcal{E}_t$) and as a linear material with properties (E_c, μ) when $\chi(\boldsymbol{\varepsilon}) = \operatorname{tr}(\boldsymbol{\varepsilon}) < 0$ (compression subdomain, $\mathcal{E}_c$). Even when $E_t \neq E_c$, both the strain energy density and stress tensor are continuous in the entire space of strains, $\mathcal{E}$, including the interface between $\mathcal{E}_t$ and $\mathcal{E}_c$.

We provide a detailed proof of continuity of both the strain energy density function (59) and the stress tensor $(61)_1$ in Appendix C.

To implement the conewise linear elastic material in PolyStress, we evaluate the stress tensor and material tangent matrix using the elastic properties in compression (E_c and μ) when $\mathrm{tr}(\boldsymbol{\varepsilon}) < 0$, or the elastic properties in tension (E_t and μ) when $\mathrm{tr}(\boldsymbol{\varepsilon}) > 0$. The implementation of this material model is provided in the material_model routine, which is provided as a supplementary material. To use the bilinear model, the following fem fields should be used in the PolyScript file (see Appendix D):

```
fem.MatModel='Bilinear', fem.MatParam=[Et,Ec,G]
```

where Et, Ec, and G are the Young's modulus in tension, the Young's modulus in compression, and the shear modulus, respectively.

6.5 Design variable update scheme

In PolyTop, the design variables are updated using a routine called UpdateScheme, which is based on the Optimality Criteria (OC) method (Bendsøe and Sigmund 2003). However, the solution of the stress-constrained problem requires the implementation of a different update scheme. Specifically, we implement a version of MMA (Svanberg 1987) tailored to solve unconstrained optimization problems such as problem (43). At each AL step, instead of solving the AL sub-problem (43), we solve an approximate convex sub-problem of the form:

$$\begin{aligned}\min_{\mathbf{z}\in[0,1]^N} \quad & \tilde{J}^{(k)}(\mathbf{z}) = r^{(k)} + \sum_{\ell=1}^{N}\left(\frac{p_\ell^{(k)}}{U_\ell^{(k)} - z_\ell} + \frac{q_\ell^{(k)}}{z_\ell - L_\ell^{(k)}}\right) \\ \text{s.t:} \quad & \bar{\alpha}_\ell^{(k)} \le z_\ell \le \bar{\beta}_\ell^{(k)}, \quad \ell = 1,\dots,N,\end{aligned} \tag{63}$$

where $\bar{\alpha}_\ell^{(k)} = \max[\underline{z}_\ell, \alpha_\ell^{(k)}]$ and $\bar{\beta}_\ell^{(k)} = \min[\bar{z}_\ell, \beta_\ell^{(k)}]$, in which $\alpha_\ell^{(k)}$ and $\beta_\ell^{(k)}$ are chosen such that $L_\ell^{(k)} < \alpha_\ell^{(k)} < z_\ell^{(k)} < \beta_\ell^{(k)} < U_\ell^{(k)}$ and, as suggested by (Svanberg 1987), we evaluate them as $\alpha_\ell^{(k)} = 0.9L_\ell^{(k)} + 0.1z_\ell^{(k)}$ and $\beta_\ell^{(k)} = 0.9U_\ell^{(k)} + 0.1z_\ell^{(k)}$. Parameters $\underline{z}_\ell$ and $\bar{z}_\ell$ are obtained as $\underline{z}_\ell = \max[0, z^{(k)} - \texttt{move}]$ and $\bar{z}_\ell = \min[1, z^{(k)} + \texttt{move}]$, where move is a prescribed move limit. In addition,[12]

$$p_\ell^{(k)} = (U_\ell^{(k)} - z_\ell^{(k)})^2\left[\max\left(\frac{\partial J}{\partial z_\ell}, 0\right) + \tau\left|\frac{\partial J}{\partial z_\ell}\right| + \frac{\theta}{U_\ell^{(k)} - L_\ell^{(k)}}\right], \tag{64}$$

$$q_\ell^{(k)} = (z_\ell^{(k)} - L_\ell^{(k)})^2\left[-\min\left(\frac{\partial J}{\partial z_\ell}, 0\right) + \tau\left|\frac{\partial J}{\partial z_\ell}\right| + \frac{\theta}{U_\ell^{(k)} - L_\ell^{(k)}}\right], \tag{65}$$

and

$$r^{(k)} = J(\mathbf{z}^{(k)}) - \sum_{\ell=1}^{N}\left(\frac{p_\ell^{(k)}}{U_\ell^{(k)} - z_\ell^{(k)}} + \frac{q_\ell^{(k)}}{z_\ell^{(k)} - L_\ell^{(k)}}\right), \tag{66}$$

where $\tau = 10^{-3}$, $\theta = 10^{-6}$, and the terms $\partial J/\partial z_\ell, \ell = 1\dots,N$ are evaluated at $\mathbf{z} = \mathbf{z}^{(k)}$.

The last terms required to define the approximate convex sub-problem (63) are the lower and upper asymptotes, $L_\ell^{(k)}$ and $U_\ell^{(k)}$, respectively, which we compute as discussed in Svanberg (1987). That is, for the first two iterations (i.e., when $k = 1$ and $k = 2$):

$$L_\ell^{(k)} = z_\ell^{(k)} - s_\ell^{(k)}(\bar{z}_\ell - \underline{z}_\ell) \quad \text{and} \quad U_\ell^{(k)} = z_\ell^{(k)} + s_\ell^{(k)}(\bar{z}_\ell - \underline{z}_\ell) \tag{67}$$

and for later iterations (i.e., when $k \ge 3$),

$$\begin{aligned} L_\ell^{(k)} &= z_\ell^{(k)} - s_\ell^{(k)}\left(z_\ell^{(k-1)} - L_\ell^{(k-1)}\right) \quad \text{and} \\ U_\ell^{(k)} &= z_\ell^{(k)} + s_\ell^{(k)}\left(U_\ell^{(k-1)} - z_\ell^{(k-1)}\right).\end{aligned} \tag{68}$$

In the current Matlab implementation, $s_\ell^{(k)} = \texttt{AsymInit}$ for $k = 1$ and $k = 2$ and

$$s_\ell^{(k)} = \begin{cases} \texttt{AsymInc} & \text{if } (z_\ell^{(k)} - z_\ell^{(k-1)})(z_\ell^{(k-1)} - z_\ell^{(k-2)}) > 0 \\ \texttt{AsymDecr} & \text{if } (z_\ell^{(k)} - z_\ell^{(k-1)})(z_\ell^{(k-1)} - z_\ell^{(k-2)}) < 0 \\ 1 & \text{otherwise,}\end{cases} \tag{69}$$

where AsymInit, AsymInc, and AsymDecr are fields prescribed in the opt structure.

The minimizer of (63) can be found explicitly, which improves the computational efficiency considerably, and it is given by (Senhora 2019):

$$z_\ell^* = \max\{\bar{\alpha}_\ell^{(k)}, \min[\bar{\beta}_\ell^{(k)}, B_\ell]\}, \tag{70}$$

where

$$B_\ell = \frac{L_\ell^{(k)}p_\ell^{(k)} - U_\ell^{(k)}q_\ell^{(k)} + (U_\ell^{(k)} - L_\ell^{(k)})\sqrt{p_\ell^{(k)}q_\ell^{(k)}}}{p_\ell^{(k)} - q_\ell^{(k)}}. \tag{71}$$

The update scheme discussed above is implemented in the MMA_unconst subroutine, which is located inside PolyStress, as can be seen in Appendix E.

7 Numerical results

In this section, we present several numerical examples to demonstrate various features of the educational PolyStress code. We obtain all results reported in this

[12]Although the expressions for $p_\ell^{(k)}$ and $q_\ell^{(k)}$ used here differ from those in Svanberg (1987), they correspond to those used in Svanberg's implementation of MMA in Matlab.

section based on the single set of parameters displayed in Table 3, which are kept constant in this paper.

Unless otherwise specified, all the problems discussed next use SIMP for the material interpolation function (see (18)). Moreover, for plotting purposes, the von Mises stress maps displayed in this section are expressed in normalized, such that the normalized von Mises stress at the centroid of each element is given by:

$$\widetilde{\sigma}_\ell^v = E_\ell \sigma_\ell^v / \sigma_{\lim}, \tag{72}$$

where $E_\ell = m_E(y_\ell)$ and σ_ℓ^v is given by (56).

7.1 Benchmark problems

Here, we solve several benchmark problems that are typically found in the stress constrained literature. With the benchmark problems, we demonstrate the ability of `PolyStress` to solve problems with an increasing number of elements/constraints, to enforce symmetry to the designs through the regularization filter, to handle problems with strong geometric singularities, among others. For all the benchmark problems, we consider a linear material model, for which we set `fem.MatModel='Bilinear'` and `fem.MatParam=`$[E_0, E_0, G]$, in which E_0 is the Young's modulus, $G = \frac{E_0}{2(1+\nu_0)}$ is the shear modulus, and ν_0 is the Poisson's ratio of the solid material.

Table 3 Input parameters used to solve all examples

Parameter	Value
Initial Lagrange multiplier estimators, $\lambda_j^{(0)}$	0
Initial penalty factor, $\mu^{(0)}$	10
Maximum penalty factor, $\mu_{\max}$	10,000
Penalty factor update parameter, α	1.10
SIMP penalization factor, p	3.5
Nonlinear filter exponent, q	3
Ersatz parameter, ϵ	10^{-8}
MMA iterations per AL step, `MMA_Iter`	5
Initial threshold projection penalization factor, $\beta^\dagger$	1
Maximum threshold projection penalization factor, $\beta_{\max}^\dagger$	10
Threshold projection density, η	0.5
Initial guess, $\mathbf{z}^{(0)}$	0.5
Convergence tolerance on design variables, `Tol`	0.002
Convergence tolerance on stress constraints, `TolS`	0.003
Maximum number of AL steps, `MaxIter`	150

†Parameter β starts at 1 and increases by 1 every five AL steps and up to the maximum value, $\beta_{\max}$, i.e., `[BFreq,B0,Binc,Bmax]=[5,1,1,10]`.

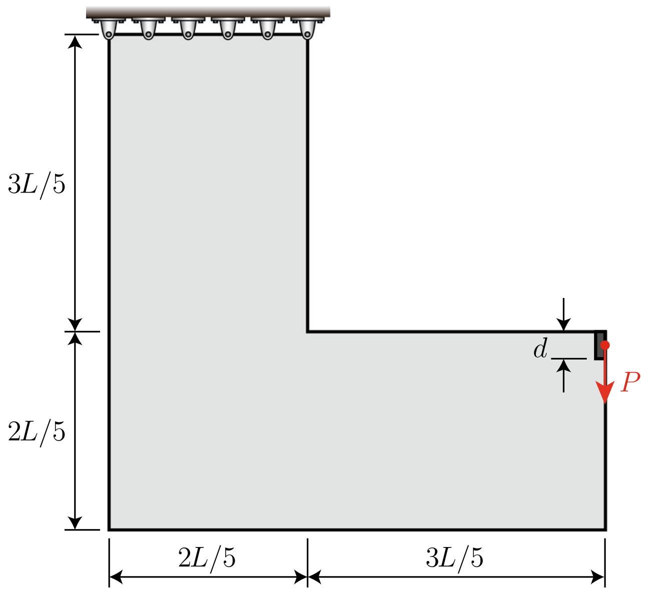


Fig. 4 L-bracket domain and boundary conditions

7.1.1 L-bracket

The first example is the *L-bracket* problem, which is ubiquitous in the stress constraints literature (e.g., see Pereira et al. 2004; Bruggi 2008; Paris et al. 2009; Paris et al. 2010; Le et al. 2010; Guo et al. 2011; Bruggi and Duysinx 2012; Xia et al. 2012; Luo et al. 2013; Zhang et al. 2013; Holmberg et al. 2013b, 2013a; Emmendoerfer and Fancello 2014; Emmendoerfer and Fancello 2016; Lee et al. 2016; Verbart et al. 2016; da Silva et al. 2018, 2019a). The domain and boundary conditions of the L-bracket are provided in Fig. 4. For this problem, we consider a linear material with Young's modulus, $E_0 = 70$ GPa, Poisson's ratio, $\nu_0 = 0.25$, and stress limit $\sigma_{\lim} = 100$ MPa.

We use `PolyMesher` (Talischi et al. 2012a) to generate four regular quadrilateral meshes with increasing levels of refinement (from ∼50,000 to ∼500,000 elements) and, for each of the four meshes, we obtain the results shown in Fig. 5 considering a filter radius of $R = 0.05$ m. As observed in the results, the optimized topologies are consistent across all levels of mesh refinement.

We analyze the efficiency of `PolyStress` using the results from the L-bracket problem. To do so, we let `PolyStress` run for 200 iterations (i.e., we run it for 40 AL steps with 5 MMA iterations per AL step) and report the computational times obtained from the Matlab profiler.[13] The code runtime breakdown for 200 iterations (40 AL steps) of the L-bracket problem is reported in Table 4.

[13]The optimization iterations are run using Matlab 2017a on desktop computer with a Xeon(R) CPU E5-1660 v3 @ 3.00 GHz processor and 256 GB of RAM.

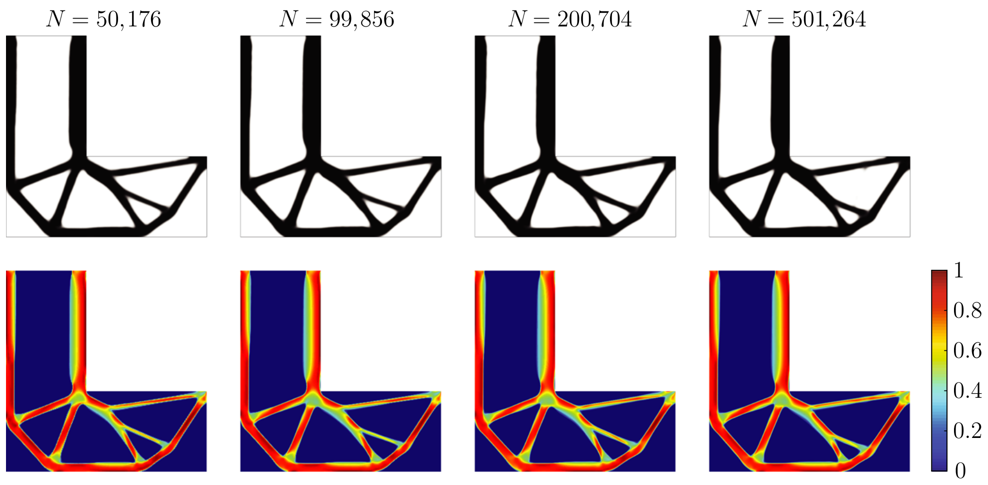


Fig. 5 L-bracket topologies (top) and von Mises stress maps (bottom) for various levels of mesh refinement

Table 4 Code runtime breakdown for 200 optimization iterations of the L-bracket problem[†]

Mesh size	50,176	99,856	200,704	501,264
Precomputations	33.7 (11.6%)	58.2 (9.4%)	116.8 (7.1%)	291.4 (4.7%)
Computing **P**	13.5 (4.7%)	36.0 (5.8%)	366.7 (22.2%)	2480.2 (40.0%)
Assembling **K**	99.4 (34.3%)	203.9 (32.8%)	419.8 (25.4%)	1065.0 (17.2%)
Solving $\mathbf{KU} = \mathbf{F}$	58.8 (20.3%)	138.7 (22.3%)	308.0 (18.6%)	886.9 (14.3%)
Evaluating von Mises stress	14.2 (4.9%)	28.0 (4.5%)	59.3 (3.6%)	149.8 (2.4%)
AL function and sensitivity	45.1 (15.5%)	97.3 (15.7%)	213.0 (12.9%)	618.8 (10.0%)
Mapping **z**, **E**, and **V**	5.0 (1.7%)	18.7 (3.0%)	80.3 (4.9%)	492.9 (7.9%)
Design variable update	0.8 (0.3%)	1.5 (0.2%)	7.8 (0.5%)	19.1 (0.3%)
Plotting the solutions	14.1 (4.9%)	27.3 (4.4%)	54.9 (3.3%)	136.2 (2.2%)
Other	5.2 (1.8%)	11.7 (1.9%)	25.3 (1.5%)	67.3 (1.1%)
Total time of PolyScript[‡]	**290**	**621**	**1652**	**6208**

[†] Times are in seconds with percentage of total runtime of PolyScript in parenthesis

[‡] The bold font is used to emphasize total runtimes

Table 5 Code runtime comparison of PolyStress and PolyTop (Talischi et al. 2012b)[†]

Mesh size	PolyStress			PolyTop		
	Total time	Iterations	Time/200 iter.[‡]	Total time	Iterations	Time/200 iter.[‡]
50,176	500	333	290	335	222	261
99,856	1063	327	621	727	223	616
200,704	2883	362	1652	1891	230	1462
501,264	9439	309	6208	8164	230	4826

[†] All runtimes are reported in seconds

[‡] These times come from Table 4 and include times for both precomputations and computation of **P**

The results show that, for smaller mesh sizes (e.g., less than 100,000 elements), most of the computational time is used to assemble the stiffness matrix and to solve the linear systems, while the time spent during precomputations (i.e., generating the FE mesh, computing the stiffness matrices of all solid elements, among others) and that spent computing the filter matrix, **P**, are relatively small. As the mesh size increases (e.g., more than 200,000 elements), the precomputation time remains small in comparison to the total time, yet the time spent computing the filter matrix increases considerably. For example, for the 200,704 element mesh, the time to compute **P** is 22% of the total time and for the 500,264 element mesh, it is 40% of the total time.

For the sake of comparison, we also solve the L-bracket problem using `PolyTop` and compare the CPU time with that of `PolyStress` in Table 5. In contrast to the results reported in Table 4, here we compare the total runtime of both `PolyTop` and `PolyStress` up until each of them reaches convergence. In addition, we compare the computational times for the first 200 iterations to assess the efficiency of `PolyStress` relative to that of `PolyTop` for the same number of iterations. Table 5 shows that the computational time for the first 200 iterations in `PolyStress` is larger than that in `PolyTop` for all mesh sizes (as expected), yet the increase in CPU time is smaller than 13% for the first three meshes and about 29% for the 501,264 element mesh. The increase in CPU time is expected because `PolyStress` requires additional computations (e.g., von Mises stress computation, update of AL parameters, among others), which are not needed in `PolyTop`. Table 5 also shows the total CPU time for both `PolyStress` and `PolyTop` up until convergence is achieved.[14] As expected, the number of iterations required to achieve convergence is larger in `PolyStress` than in `PolyPolyTop`. That is because of the local nature of the stress-constrained problem, which uses a different methodology and is more demanding computationally in the sense that it is highly nonlinear and requires a large number of iterations to satisfy all stress constraints.

`PolyStress` can also be used with other stress constraint definitions such as the traditional vanishing constraint (Cheng and Jiang 1992). For instance, to implement a traditional vanishing constraint of the form:

$$g_j(\mathbf{z}, \mathbf{U}) = m_E(y_j)\Lambda_j \leq 0, \quad j = 1, \ldots, N,$$

we replace lines 73, 77, and 79 of the `PolyStress.m` code by

```
73 g = E.*s;
77 dhdVM(a1) = E(a1).*1/fem.SLim;
79 dPenaldE(a1) = (lambda(a1)+mu. *h(a1)).*s(a1);
```

We solve the L-bracket problem discretized with 50,176 elements, considering the traditional vanishing constraint shown above, to compare the results with those obtained using the polynomial vanishing constraint. To solve this problem with `PolyStress`, we first calibrate the value of $\mu^{(0)}$ (i.e., the initial penalty factor) to work appropriately with the new constraint definition. For this problem, we find that $\mu^{(0)} = 200$ yields satisfactory results.

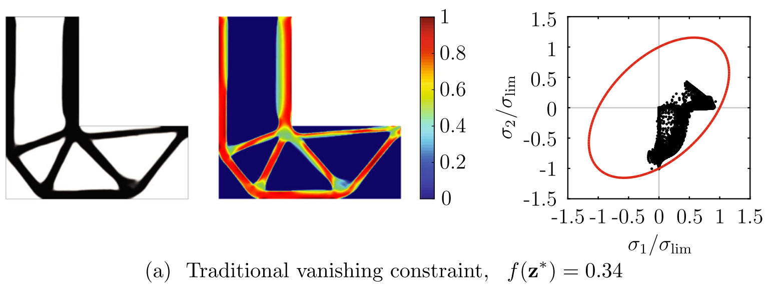
Figure 6 displays both the results we obtain using the traditional vanishing constraint (Fig. 6a) and those obtained using the polynomial vanishing constraint (Fig. 6b). The results obtained with either the traditional constraint or the polynomial one satisfy the stress constraints locally, as observed by the von Mises stress maps (center plots) or by the von Mises yield surface (right plots), which show that all stress evaluation points are inside the von Mises envelope. Although `PolyStress` is able to obtain a solution that satisfies the stress constraints locally, the results show that the topology obtained with the traditional constraint differs from that obtained with the polynomial constraint. Moreover, the results show that the optimized volume fraction, $f(\mathbf{z}^*)$, obtained with the traditional constraint is larger than that obtained with the polynomial one. Both the difference in topology and optimized volume fractions are expected outcomes because the stress-constrained problem is non-convex, and thus, the optimizer is likely to converge to a different local minima when the parameters of the model change or when different stress constraint definitions are used.

7.1.2 Portal frame

The next benchmark problem is the portal frame (Le et al. 2010; Lian et al. 2017; da Silva et al. 2019a), whose geometry and boundary conditions are shown in Fig. 7. The domain is discretized with 100,000 polygonal finite elements using `PolyMesher`.[15] For this problem, we use a linear material with Young's modulus, $E_0 = 100$ GPa, Poisson's ratio, $\nu_0 = 0.25$, and stress limit, $\sigma_{\text{lim}} = 1000$ MPa.

[14] For the case of `PolyTop`, we use a convergence tolerance, `Tol` $= 10^{-4}$

[15] To generate the polygonal mesh to run this problem, line 11 of the provided PolyScript.m file should be replaced by `[Node,Element,Supp,Load,~] = PolyMesher(@PortalDomain,NElem,100)`, using `NElem=100000`.



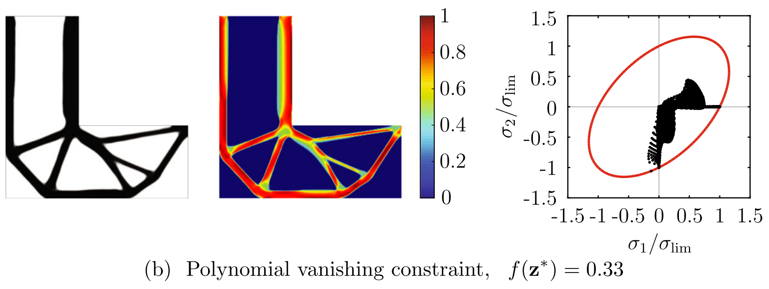


Fig. 6 Optimized topologies (left), von Mises stress maps (center), and von Mises yield surfaces (right) for the L-bracket meshed with 50,176 elements. The results displayed here are obtained using **a** the traditional vanishing constraint and **b** the polynomial vanishing constraint. The right figures display the principal stresses at the centroid of all elements together with the von Mises yield surface to show that all stress evaluation points are inside the envelope

For this problem, we compare the results obtained when using either SIMP or RAMP as the material interpolation function. To obtain the results using RAMP, we use $p_0 = 3.5$ in $(18)_2$. The solutions, depicted in Fig. 8, are obtained using a filter radius, $R = 0.25$ m, and imposing symmetry about the y-axis, which we achieve through the filter operator. To impose symmetry, we compute the filter matrix, $\mathbf{P}$, using `PolyFilter`, as `P = PolyFilter(fem,R,q,'Y')`, where $q = 3$ are the nonlinear filter exponent (see Table 3). Although a fairly symmetric solution can be obtained without imposing symmetry, we impose it because the polygonal mesh that we used is not symmetric about the y-axis. As observed from the results, `PolyStress` successfully removed material from the region of stress concentration located at the

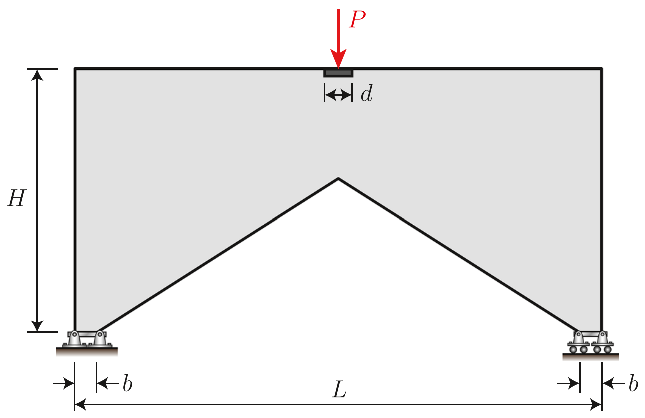


Fig. 7 Portal frame domain

re-entrant corner on the lower center part of the portal frame for both the SIMP and RAMP designs. Due to the highly nonlinear behavior of the stress-constrained problem, the solution obtained for SIMP differs from that obtained for RAMP, as expected. Finally, the results also show that the stresses are satisfied locally, which can be observed from the results on the center or on the right of the figure.

7.1.3 Eyebar

Here, we solve a problem usually referred to as the eyebar. This practical design problem was introduced by Pereira et al. (2004) to find the optimized topology of an eyebar that is part of an eyebar chain of a suspension bridge. The geometry and boundary conditions of the eyebar problem are shown in Fig. 9. The hole is subjected to a horizontal load of magnitude P, distributed according to the function $t(x, y) = r^2 - y^2$, in which $(x, y) = (0, 0)$ is the center of the circle and r is the radius of the circle.

For this problem, we consider a linear material with Young's modulus, $E_0 = 200$ GPa, Poisson's ratio, $\nu_0 = 0.3$, and a stress limit, $\sigma_{\text{lim}} = 450$ MPa. To obtain the optimized topology of the eyebar reported in Fig. 10, we discretize the design domain using 100,000 polygonal finite elements (Talischi et al. 2012a) and use a filter radius, $R = 0.04$ m.[16]

[16]To generate the polygonal mesh to run this problem, line 11 of the provided PolyScript.m file should be replaced by `[Node,Element, Supp,Load,~] = PolyMesher(@EyeBarDomain,NElem, 100)` with `NElem=100000`.

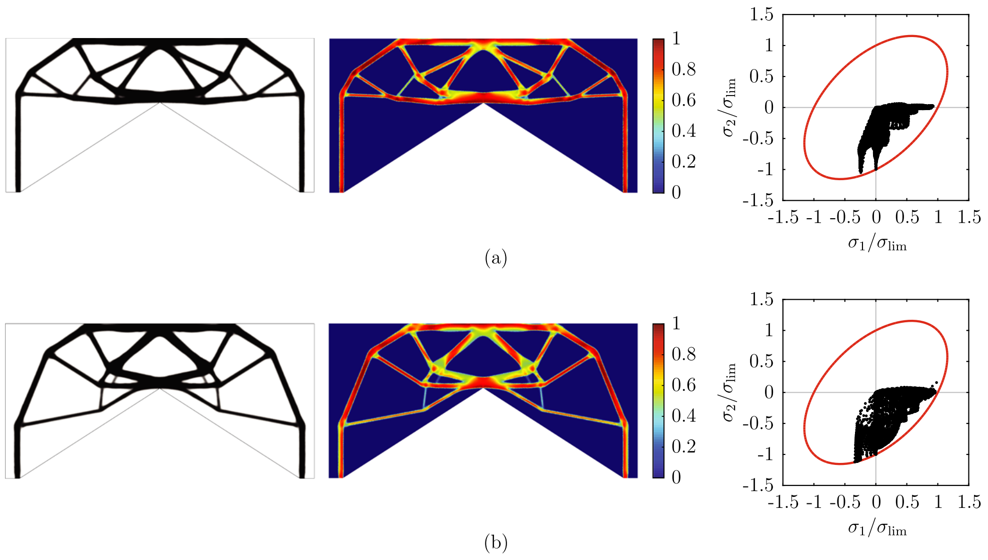


Fig. 8 Portal frame topologies (left), von Mises stress maps (center), and von Mises yield surfaces (right) obtained using **a** SIMP and **b** RAMP

As observed in Fig. 10, small holes appear at two locations along the loaded portion of the domain boundary. Such features are unlikely to be obtained using traditional design methodologies, which highlights the importance of stress-based topology optimization to achieve non-intuitive designs.

7.1.4 Crack

The following problem, denoted to as the *crack problem*, is found in studies by Emmendoerfer and Fancello (2014) and Emmendoerfer and Fancello (2016), and most recently in a study by Chu et al. (2018). Here, we solve a slightly modified version of the crack problem by Emmendoerfer and Fancello (2014), as seen in Fig. 11. Due to symmetry, we only model half of the design domain.[17] We use this problem to test the ability of PolyStress to solve stress-constrained problems in domains with strong singularities such as those found at the tip of a crack.

For this problem, we consider a linear material with Young's modulus, $E_0 = 70$ GPa, Poisson's ratio, $\nu_0 = 0.25$, and stress limit, $\sigma_{\rm lim} = 100$ MPa. We obtain the optimized topology shown in Fig. 12 using 100,352 regular quadrilateral elements and a filter radius, $R = 0.045$ m. As seen in the results, PolyStress is able to round the crack tip in order to remove the geometric singularity that was causing the stress concentration.

7.2 Corbel design

Here, we use PolyStress to obtain optimized topologies for a corbel structure whose geometry and boundary conditions are shown in Fig. 13 (left). We obtain three designs, one for a linear material, one for a tension-dominated bilinear material, and one for the same tension-dominated bilinear material with symmetry about the x-axis. The uniaxial stress-strain curves for the linear and the bilinear material are depicted in Fig. 13 (right). To impose symmetry, we compute the filter matrix as `P = PolyFilter(fem,R,q,'X')`. The linear material has

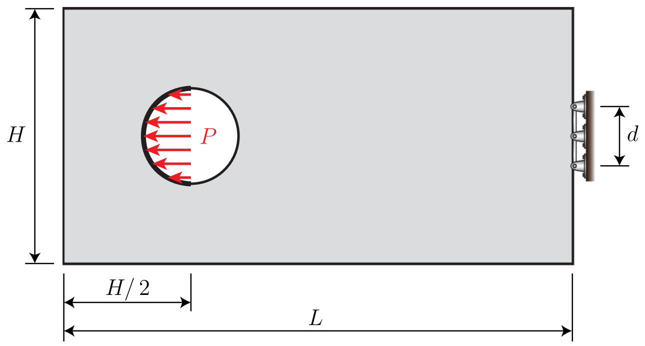


Fig. 9 Eye bar domain and boundary conditions

[17]To generate the mesh to run this problem, line 11 of the provided PolyScript.m file should be replaced by `[Node,Element,Supp,Load] = Mesh_Crack_Prob(Ne_ap); NElem=size(Element,1)`, using `Ne_ap=100000`.

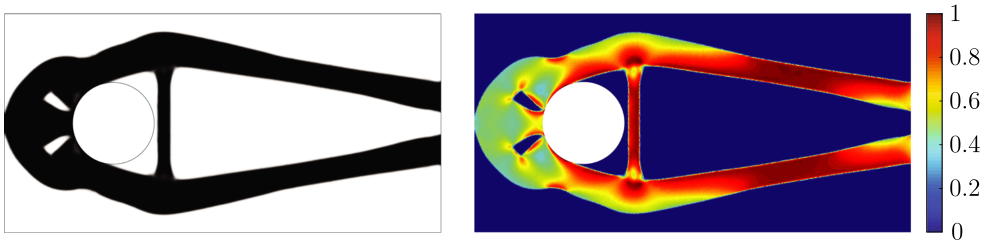


Fig. 10 Eye bar topology (left) and von Mises stress map (right)

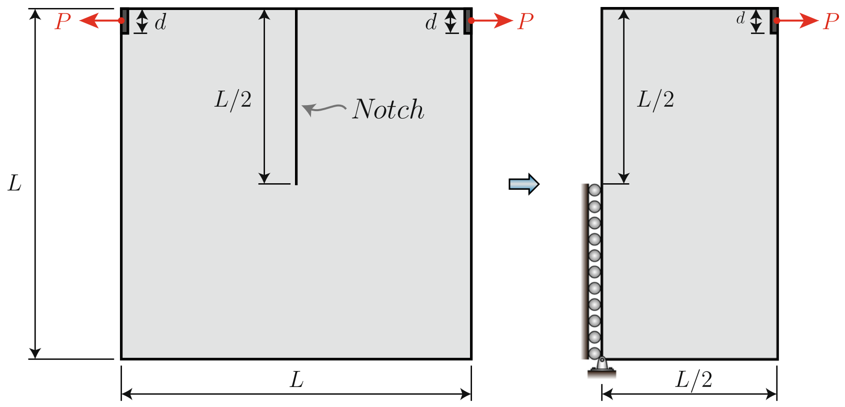


Fig. 11 Crack domain and boundary conditions

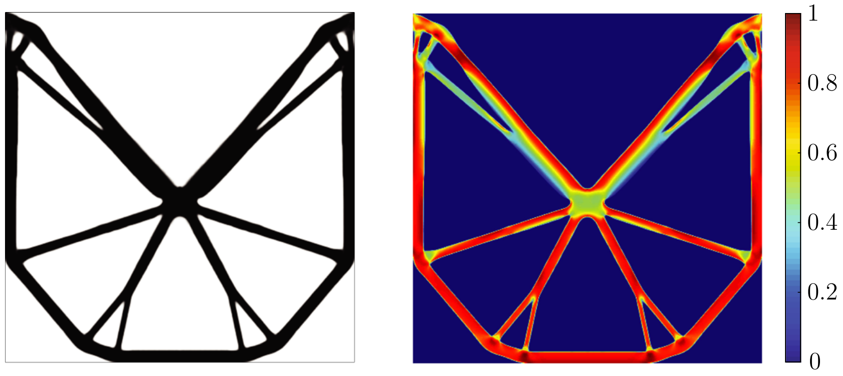


Fig. 12 Crack topology (left) and von Mises stress map (right)

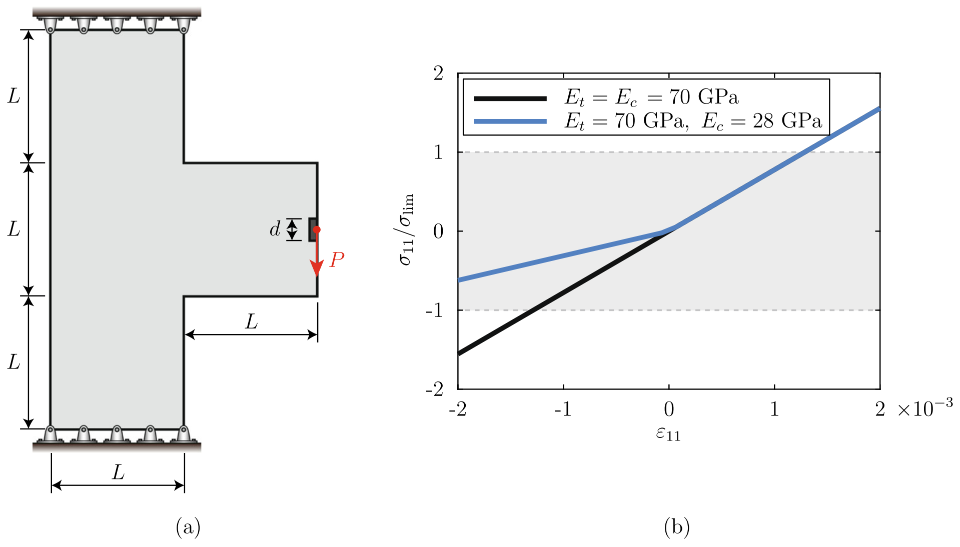


Fig. 13 Corbel design: **a** corbel domain and boundary conditions and **b** uniaxial stress-strain curves for two material models (i.e., for linear and a bilinear material) used for the design

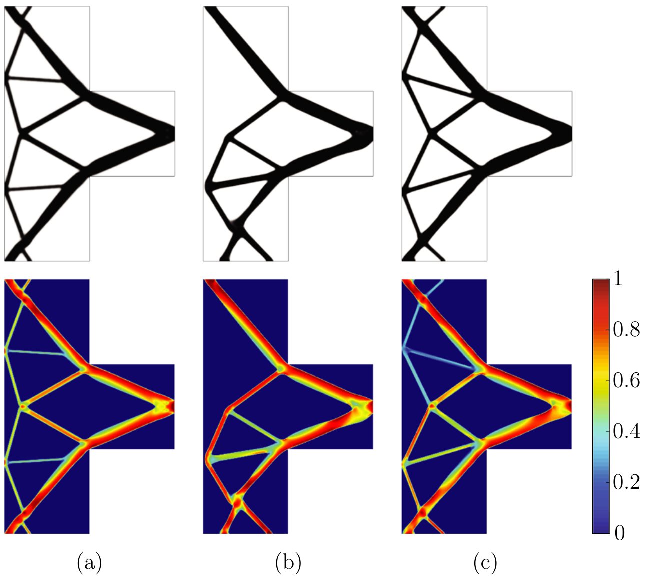


Fig. 14 Corbel topologies(top) and von Mises stress maps (bottom) for **a** linear material, **b** tension-dominated bilinear material, and **c** tension-dominated bilinear material with symmetry imposed through the filter operator

Young's modulus, $E_0 = 70$ GPa, and Poisson's ratio, $\nu_0 = 0.25$, and the bilinear material has Young's modulus in tension, $E_t = 70$ GPa, Young's modulus in compression, $E_c = 28$ GPa, and shear modulus, $G = 28$ GPa. To obtain the optimized topologies, we discretize the design domain using 79,524 regular quadrilateral elements and consider a filter radius, $R = 0.15$ m.[18]

The results obtained for the linear material are shown in Fig. 14a and, as expected, the results are symmetric with respect to the x-axis. For the linear case, we obtain an optimized volume fraction of 0.23. The non-symmetric results for the bilinear material are shown in Fig. 14b. In this case, part of the compression-dominated region that appears in Fig. 14a is removed. The optimized structure of Fig. 14b has a volume fraction of 0.22. Now, when symmetry is imposed, we obtain the results shown in in Fig. 14c. Because imposing symmetry adds additional constraints to the formulation (which are implicitly added via the filter operator), we expect the volume fraction of the symmetric structure of Fig. 14c to be larger than that of Fig. 14b. That is indeed the case because the symmetric results led to an optimized volume fraction of 0.25.

7.3 Antenna support bracket design

This example aims to find the optimized topology of the antenna support bracket with geometry and boundary conditions illustrated in Fig. 15a. We obtain three designs, one in which we use a linear material with Young's modulus, $E_0 = 120$ GPa, Poisson's ratio, $\nu_0 = 0.3$, and stress limit, $\sigma_{\lim} = 1000$ MPa, and two in which we use a nonlinear material with strain energy density function given by the compressible Ogden model (Ogden 1972; Feng et al. 2006):

$$W_0(\varepsilon_1, \varepsilon_2, \varepsilon_3) = \sum_{p=1}^{n} \frac{\mu_p}{\alpha_p}(\lambda_1^{\alpha_p} + \lambda_2^{\alpha_3} + \lambda_1^{\alpha_p} - 3) + \sum_{p=1}^{n} \frac{\mu_p}{\alpha_p \beta_p}\left[(\lambda_1\lambda_2\lambda_3)^{-\alpha_p\beta_p} - 1\right], \quad (73)$$

where α_p, μ_p, β_p, and n are material parameters and $\lambda_i = \varepsilon_i + 1$, $i = 1, 2, 3$ are the principal stretches under small deformations. To incorporate the Ogden model into `PolyStress`, we only need to add the stress vector, $\boldsymbol{\sigma}$, and the material tangent matrix, $\mathbf{D}$, to the `material_model` subroutine (see electronic supplementary material). Analytical expressions for the stress vector and material tangent matrix can be found in a study by Chi et al. (2019). To obtain designs using the Ogden model, we use the same Young's modulus and Poisson's ratio considered for the linear material (i.e., $E_0 = 120$ GPa and $\nu_0 = 0.3$) and take only one term of the Ogden model (i.e., we use $n = 1$) with two different values of α_1 (e.g., we use $\alpha_1 = 100$ and $\alpha_1 = -100$, which leads to a tension-dominated and a compression-dominated material, respectively). To use the Ogden material model in `PolyStress`, the following fields should be modified in the `fem` structure defined in the `PolyScript.m` file:

`fem.MatModel='Ogden'` and

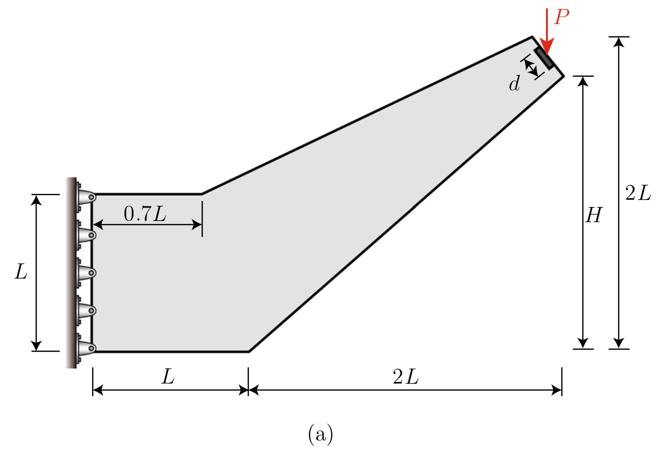
`fem.MatParam` $= [\mu_1, \alpha_1, \beta_1]$,

where $\mu_1 = \frac{E_0}{\alpha_1(1+\nu_0)}$ is the initial shear modulus and $\beta_1 = \frac{\nu_0}{1-2\nu_0}$ (Chi et al. 2019).[19]

Figure 15b depicts the uniaxial stress-strain curves for the three materials considered in the analysis. The figure shows the uniaxial stress, σ_{11}, normalized with respect to the stress limit, $\sigma_{\lim}$, which is useful to determine the level of strains that we expect in our optimized designs. To determine the expected level of strains for each design, we find the values of ε_{11} at the intersection between each of the stress-strain curves with the two horizontal dashed lines corresponding to $\sigma_{11}/\sigma_{\lim} = \pm 1$. Using this procedure, we obtain that the expected level of strains for the linear design ranges approximately between -8×10^{-3} and 8×10^{-3}, for the tension-dominated material (Ogden 1) between -12×10^{-3} and 6×10^{-3}, and for the compression-dominated material (Ogden 2) between -6×10^{-3} and 12×10^{-3}.

[18]To generate the mesh to run this problem, line 11 of the provided PolyScript.m file should be replaced by `[Node,Element, Supp,Load] = Mesh_Corbel(Ne_ap); NElem=size(Element,1)`, using `Ne_ap=80000`.

[19]When $n > 1$ in (73), `fem.MatParam` should be populated as follows: `fem.MatParam` $= [\mu_1, \alpha_1, \beta_1, \ldots, \mu_n, \alpha_n, \beta_n]$.



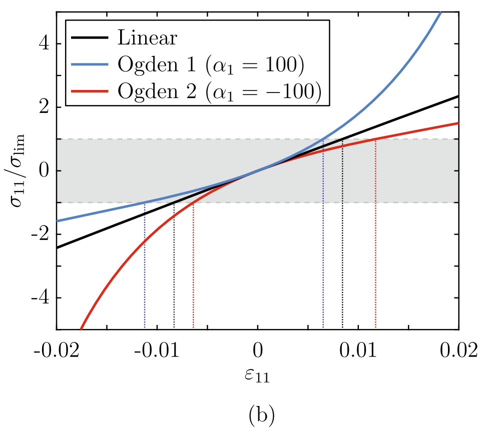


Fig. 15 Problem setup for antenna support bracket design: **a** antenna support bracket domain and boundary conditions and **b** uniaxial stress-strain curves for three materials used to obtain optimized designs for an antenna support bracket. The vertical dashed lines correspond to the intersection between the uniaxial stress-strain curves and $\sigma_{11}/\sigma_{\lim} = \pm 1$, which gives an indication of the expected level of strain that the optimized topologies should experience

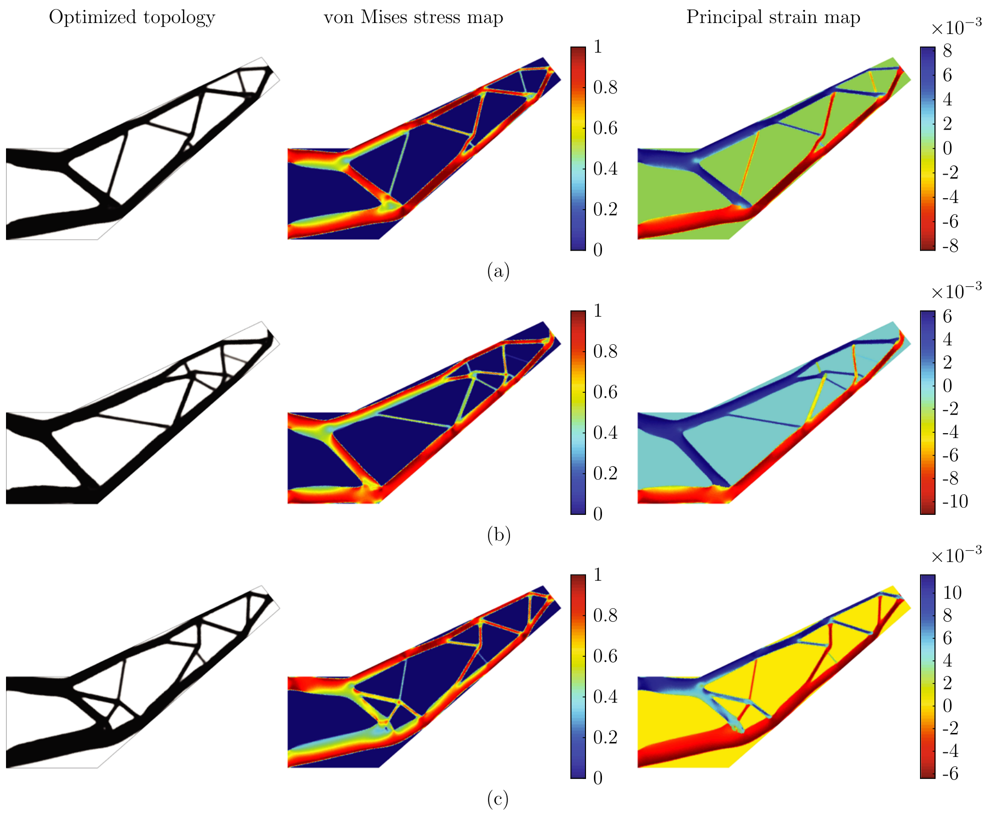
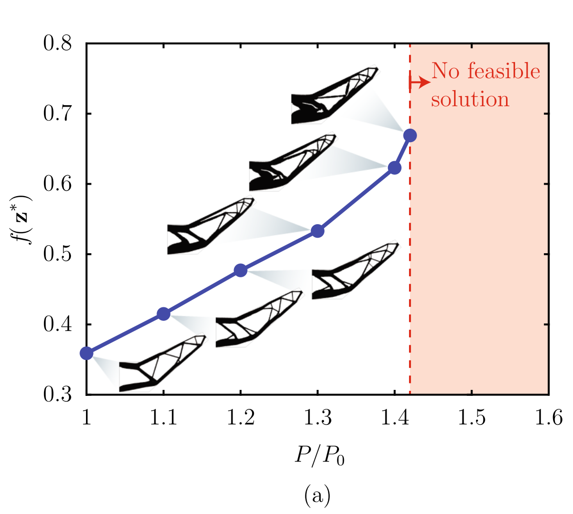


Fig. 16 Antenna support bracket topologies (left), von Mises stress maps (center), and principal strain maps (right) for **a** a linear material, **b** tension-dominated material (Ogden 1), and **c** compression-dominated material (Ogden 2) (cf. Fig. 15b)



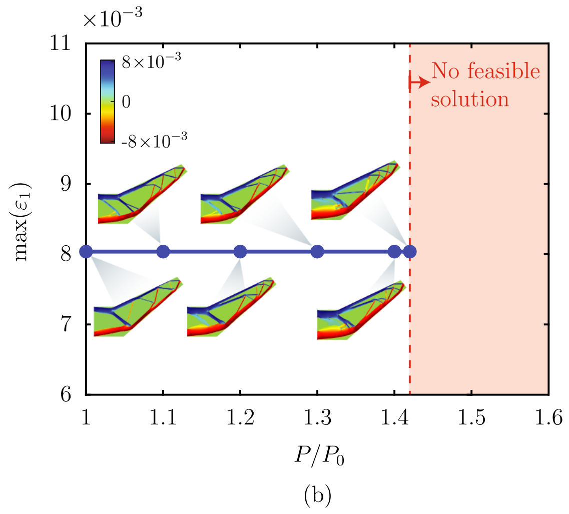


Fig. 17 Effect of the magnitude of the external load on the optimization results for the antenna support bracket considering a linear material: **a** optimized volume fraction as a function of P/P_0 and **b** maximum principal strains for optimized designs as a function of P/P_0. The value of P_0 corresponds to the magnitude of the load used to obtain the results in Fig. 16

We discretize the design domain using 30,000 polygonal finite elements (Talischi et al. 2012a) and consider a filter radius, $R = 0.07$ m to arrive at the optimized topologies shown in Fig. 16.[20] As observed from the results, the optimized topologies and the level of deformation obtained in the optimized designs are highly sensitive to the choice of material model. First, for the linear material model (Fig. 16a), we obtain that the principal strains oscillate between -8×10^{-3} and 8×10^{-3}. Second, for the tension-dominated material (Fig. 16b), the level of deformation under compression is larger than that under tension, leading to a distribution of principal stresses ranging between -12×10^{-3} and 6×10^{-3}. Finally, for the compression-dominated material (Fig. 16c), the optimized structure deforms more under tension than under compression, leading to a distribution of principal strains that ranges between -6×10^{-3} and 12×10^{-3}. The levels of deformation of the three optimized designs behave as expected from the analysis conducted for the uniaxial stress-strain curves from Fig. 15b. The strain levels for all designs shown in Fig. 16 are small enough so that the theory of small-strain elasticity holds.[21]

The magnitude of the external load affects plays a role in the optimization results. Specifically, as the magnitude of the external load increases, the optimized structures will have a larger volume fraction and vice-versa. Nonetheless, we argue that the external load level only plays a role on the value of the optimized volume fraction but not on the maximum and minimum strain levels of the optimized structures. That is because the maximum and minimum strain levels are essentially limited by the stress limit, $\sigma_{\lim}$ (see Fig. 15b). To verify this hypothesis, we obtain optimized topologies for the antenna support bracket made with a linear material considering various values of P/P_0, in which P_0 is the load used to obtain the results in Fig. 16. As we discussed above, the results depicted in Fig. 17 show that, as P/P_0 increases, the optimized volume fraction increases and the maximum strain for the optimized topologies remains unchanged.

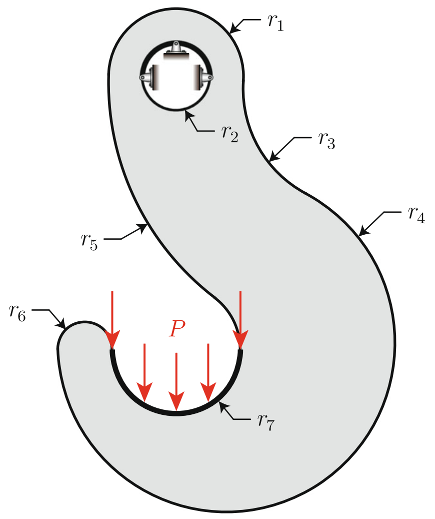


Fig. 18 Hook domain

[20]To generate the polygonal mesh to run this problem, line 11 of the provided PolyScript.m file should be replaced by `[Node,Element,Supp,Load,~] = PolyMesher(@AntennaDomain,NElem,100)` with `NElem=30000`.

[21]If the level of strains for a given problem increases, it may reach a point in which small-strain elasticity no longer holds. For this type of problems, the entire formulation must be modified to account for finite deformations, which is out of the scope of the present work.

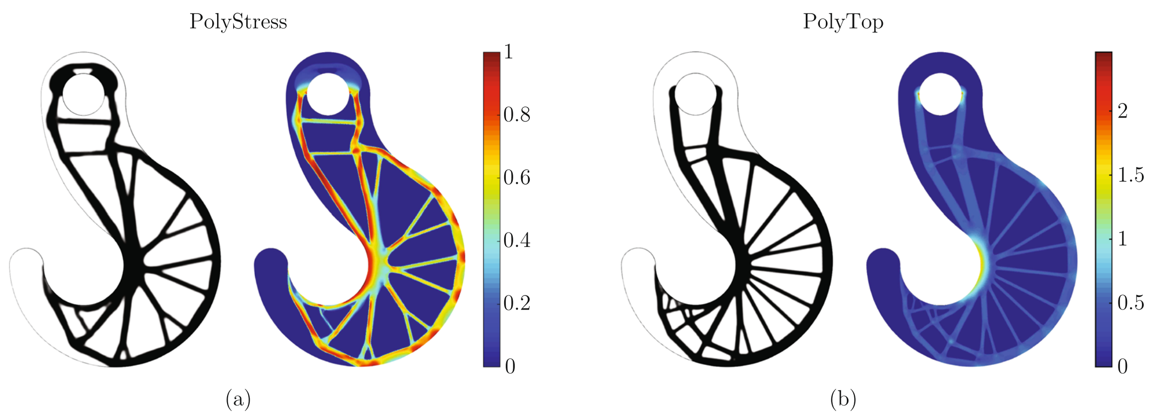


Fig. 19 Hook topology (left) and von Mises stress map (right) for (a) the stess-based design and (b) a compliance-based design. Both designs have a volume fraction of 0.35, yet the compliance-based design exceeds the stress limit by about 150%

7.4 Hook design

The last set of results corresponds to the design of a *hook* whose geometry and boundary conditions are shown in Fig. 18. The hook is made of a linear material with Young's modulus, $E_0 = 100$ GPa, Poisson's ratio, $\nu_0 = 0.25$, and stress limit, $\sigma_{\text{lim}} = 120$ MPa. In this example, we compare the optimized topology obtained from PolyStress (stress-based design) with that obtained from PolyTop (compliance-based design). We obtain both designs using the same material interpolation function and continuation scheme. That is, the volume interpolation function is given by (17) and the stiffness interpolation function is given by SIMP, as shown in (18). We run PolyTop starting with $\beta = 1$ and increase β by 1 and p by 0.5 every 25 iterations.[22] Unlike the stress-constrained problem, the compliance minimization problem is convex when $p = 1$. Thus, to solve the compliance minimization problem, we employ a continuation scheme such that we start with $p = 1$ (the convex case) and increase p by 0.5 each time β is updated, which helps penalizing intermediate densities and yields black-and-white solutions. Since the stress-constrained problem is not convex even for $p = 1$, there is no clear advantage of using a continuation scheme for p as in the compliance minimization case.

Using 100,000 polygonal finite elements (Talischi et al. 2012a) and a filter radius, $R = 0.04$ m, we obtain the optimized topologies shown in Fig. 19.[23]

[22]The 25 iterations per continuation step in PolyTop is consistent with the 5 AL steps per continuation step in PolyStress because, according to Table 3, PolyStress runs 5 MMA iterations for each AL step (i.e., MMA_Iter × BFreq = 25 FE solves per continuation step).

[23]To generate the polygonal mesh to run this problem, line 11 of the provided PolyScript.m file should be replaced by `[Node,Element,Supp,Load,~] = PolyMesher(@HookDomain,NElem,100)` with `NElem=100000`.

The results show clear topological differences between the two designs. For instance, in the stress-based design (Fig. 19a) the material underneath the circular hole moves outward (towards the outer boundary of the domain) to drive the stresses both at the location of the hole and in the lower part of the domain to values below the stress limit. Conversely, in the compliance-based design (Fig. 19b), the members that develop underneath the circular hole are closely spaced, which leads to a stress concentrations both at the sides of the hole and in the lower part of the design domain, which exceed the stress limit, σ_{lim}. The comparison between these two designs highlights the importance of stress-based formulations to obtain structures that will not fail under the applied loads.

8 Remarks

We provide results for several benchmark problems, which demonstrate the ability of PolyStress to solve stress-constrained topology optimization problems for various levels of mesh refinement and for various design domains. For example, we solved the classical *L-bracket* problem for various mesh sizes (from about 50,000 elements to about 500,000 elements) and showed that the solutions resemble one another independently of the mesh size. A computational efficiency analysis conducted on the L-bracket (considering a linear material model) revealed that the code runtime of PolyStress, which considers thousands of local constraints, is of the same order of magnitude as that of PolyTop, which is designed to solve compliance minimization problems with a single volume constraint.

In addition to the benchmark problems, we solve two additional problems in which we consider nonlinear material models. The first of those problems pertains to the design of a corbel structure made of either a linear or a bilinear material. For the nonlinear material case, we highlight the ability of PolyStress to obtain symmetric results by simply modifying the function that computes the filtered density field. The second of those examples focuses on the design of an antenna support bracket made of a compressible Ogden material. We use a one-term Ogden model to represent a tension-dominated and a compression-dominated material and use these two materials to obtain optimized topologies for the bracket. We clearly observe that the optimized topologies are highly sensitive to the choice of material model. We provide a final example in which we design a Hook made of a linear material. We compare the solution from PolyStress with that from PolyTop to point out the importance of a stress-based framework to obtain optimized structures that can carry the applied loads while avoiding material failure.

9 Conclusions

We have presented a framework for topology optimization considering local (non-aggregated) stress constraints that is based on the augmented Lagrangian (AL) method, which led to the educational Matlab code, PolyStress. The code belongs to a family of educational Matlab codes for topology optimization on unstructured polygonal finite element meshes including PolyTop (Talischi et al. 2012b), PolyFluid (Pereira et al. 2016), and PolyMat (Sanders et al. 2018). PolyStress considers material nonlinearities, and thus, we modify PolyTop's analysis routine to solve the nonlinear finite element problem using the Newton-Raphson method. To solve the problem with local stress constraints, we also modify the optimization algorithm and incorporate the AL method. Using the AL method, we solve the original optimization problem with many constraints as a series of unconstrained optimization problems and update the design variables using a version of MMA tailored to solve the unconstrained AL sub-problems. The solution of the convex approximation for the unconstrained sub-problems can be found explicitly, which saves a considerable amount of computational resources.

PolyStress is the first educational code for stress-constrained topology optimization available in the technical literature, and thus, we hope that it will motivate researchers in the field to explore and learn AL-based techniques to solve optimization problems beyond those investigated in this work. Because PolyStress has the unique ability to solve optimization problems with many stress constraints, we also hope that it will serve as a bridge toward the development of commercial software that can make topology optimization suitable for industry-relevant applications.

We conclude this work by quoting Duysinx and Bendsøe (1998):

"The improvement of the quality of the analysis of the stress field is also of great interest in order to make the stress design methods more relevant in many applications."

In the present context, we reinforce that the stress design methods in engineering applications rely on consistency between topology optimization considering stress constraints and continuum mechanics. Thus, stresses should be treated locally both in the optimization phase and in the numerical solution of the associated boundary value problem.

Acknowledgments This paper is dedicated to Prof. Martin P. Bendsøe. He motivated our interest in the research reported in the present paper, and in other recent ones, such as Senhora et al. (2020), and Giraldo-Londoño and Paulino (2020). More specifically, our intellectual curiosity in the topic was sparked by an early discussion between Profs. Paulino and Bendsøe during the WCSMO 10 (World Congress on Structural and Mutidisciplinary Optimization), which was held in Orlando, Florida (USA) from May 19th to 24th, 2013.

Funding information The authors received support from the US National Science Foundation under grant #1663244 and from the Raymond Allen Jones Chair at the Georgia Institute of Technology.

Compliance with ethical standards

Disclaimer The interpretation of the results of this work is solely that by the authors, and it does not necessarily reflect the views of the sponsors or sponsoring agencies.

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results All results presented in this manuscript can be replicated using the codes provided as electronic supplementary material.

Appendix A: Library of benchmark examples

We provide a summary of the design problems discussed herein, including a description of the design domain and the names of the Matlab files needed to generate the finite element mesh for each problem.

Appendix B: Imposing symmetry using the regularization filter

In a continuum setting, we can impose symmetries to the space of admissible density fields by means of the operator $\mathcal{P}_s(\eta(\mathbf{x}))$, which maps the design function $\eta\mathbf{x}$ according to the desired symmetries (e.g., $\mathcal{P}_s(\eta(\mathbf{x})) = \eta(x_1, |x_2|)$ if we desire symmetries with respect to the x_1-axis). The symmetrized space of admissible density fields is given by (12) and leads to a regularization mapping of the form $\mathcal{P}(\eta) = (\mathcal{P}_F \circ \mathcal{P}_s)(\eta)$.

Now, in a discretized setting, the symmetrized filter matrix $\mathbf{P}$ (i.e., the discrete counterpart of $\mathcal{P}$), becomes:

$$P_{\ell k} = \frac{w_{\ell k} v_k}{\sum_{j=1}^{N} w_{\ell j} v_j}, \quad w_{\ell k} = \max\left(0, 1 - \frac{\|\tilde{\mathbf{x}}_\ell - \tilde{\mathbf{x}}_k\|_2}{R}\right)^q, \tag{74}$$

where $\tilde{\mathbf{x}}_\ell$ (or $\tilde{\mathbf{x}}_k$) are given by:

$$\tilde{\mathbf{x}}_\ell = \left[x_1^\ell, \ |x_2^\ell|\right]^T, \tag{75}$$

when imposing symmetry about the x_1-axis:

$$\tilde{\mathbf{x}}_\ell = \left[|x_1^\ell|, \ x_2^\ell\right]^T, \tag{76}$$

when imposing symmetry about the x_2-axis, and:

$$\tilde{\mathbf{x}}_\ell = \left[|x_1^\ell|, \ |x_2^\ell|\right]^T, \tag{77}$$

when imposing symmetry about both the x_1-axis and the x_2-axis. The capability of imposing symmetry to the designs through the filter matrix has been added to the PolyFilter routine, which can be found in the electronic supplementary material.

Appendix C: Continuity proof for the bilinear material model

As discussed by Curnier et al. (1994), the specific strain energy function from (59) is continuous in the entire space

Table 6 Examples provided with PolyStress

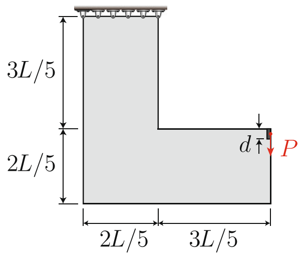
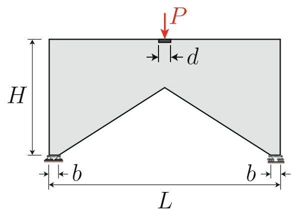
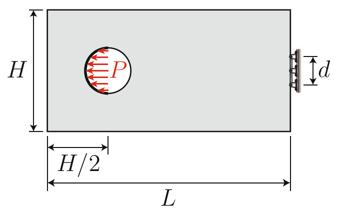
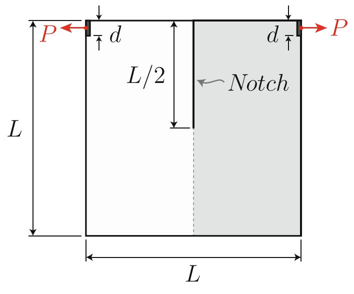
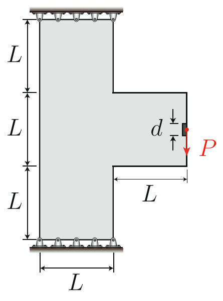
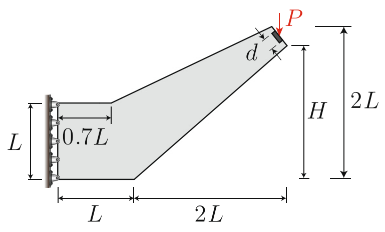
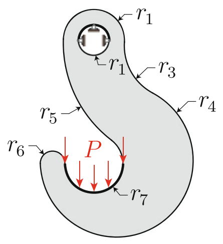
Domain	Description
	•PolyMesher domain file: @L_bracketDomain • Dimensions and loads: $L = 1$, $d = 0.06$, $P = 2$ • Material: linear with $E_0 = 70$ GPa, $\nu_0 = 0.25$, and $\sigma_{\lim} = 100$ MPa • Filter radius: $R = 0.05$
	•PolyMesher domain file: @PortalDomain • Dimensions and loads: $L = 12$, $H = 6$, $d = 1$, $P = 300$ • Material: linear material with $E_0 = 100$ GPa, $\nu_0 = 0.25$, and $\sigma_{\lim} = 1000$ MPa • Filter radius: $R = 0.25$
	•PolyMesher domain file: @EyeBarDomain • Dimensions and loads: $L = 1.6$, $H = 0.8$, $R = 0.15$, $d = 0.15$, $P = 70$ •Material: linear with $E_0 = 200$ GPa, $\nu_0 = 0.3$, and $\sigma_{\lim} = 450$ MPa • Filter radius: $R = 0.04$
	•PolyMesher domain file: @CrackDomain • Dimensions and loads: $L = 2$, $d = 0.1$, $P = 5$ • Material: linear with $E_0 = 70$ GPa, $\nu_0 = 0.25$, and $\sigma_{\lim} = 100$ MPa • Filter radius: $R = 0.045$

Table 6 (continued)

Domain	Description
	•PolyMesher domain file: @CorbelDomain • Dimensions and loads: $L = 2$, $d = 0.3$, $P = 15$ • Material 1: linear with $E_0 = 70$ GPa, $\nu_0 = 0.25$, and $\sigma_{\lim} = 90$ MPa • Material 2: bilinear material with $E_t = 70$ GPa, $E_c = 28$ GPa, $G = 28$ GPa, and $\sigma_{\lim} = 90$ MPa • Filter radius: $R = 0.15$
	•PolyMesher domain file: @AntennaDomain • Dimensions and loads: $L = 1$, $H = 1.75$, $d = 0.048$, $P = 40$ • Material 1: linear with $E_0 = 120$ GPa, $\nu_0 = 0.3$, and $\sigma_{\lim} = 1000$ MPa • Materials 2 and 3: compressible Ogden with $E_0 = 120$ GPa, $\nu_0 = 0.3$, $\alpha_1 = \pm 100$, and $\sigma_{\lim} = 1000$ MPa • Filter radius: $R = 0.08$
 r_1 r_1 r_3 r_4 r_5 r_6 P r_7	•PolyMesher domain file: @HookDomain • Dimensions and loads: dimensions are 1/100 of those from Talischi et al. (2012b) and $P = 5$ • Material: linear with $E_0 = 100$ GPa, $\nu_0 = 0.25$, and $\sigma_{\lim} = 120$ MPa • Filter radius: $R = 0.04$

of infinitesimal strains, $\mathcal{E}$. In fact, according to (59), the material behaves as a linear material with properties (λ_t, μ) when $\chi(\boldsymbol{\varepsilon}) = \mathrm{tr}(\boldsymbol{\varepsilon}) > 0$ (i.e., in the tension sub-domain, $\mathcal{E}_t$) and as a linear material with properties (λ_c, μ) when $\chi(\varepsilon) = \mathrm{tr}(\varepsilon) < 0$ (i.e., in the compression sub-domain, $\mathcal{E}_c$). Below, we demonstrate that both the strain energy and the stress tensor are continuous in the entire space of infinitesimal strains.

First, we demonstrate that the strain energy density function (59) is continuous in $\mathcal{E}$. According to (59)–(60), if $\mathrm{tr}(\boldsymbol{\varepsilon}) > 0$:

$$W_0 = W_0^+ = \frac{1}{2}\lambda_t \mathrm{tr}^2(\boldsymbol{\varepsilon}) + \mu \mathrm{tr}(\boldsymbol{\varepsilon}^2), \tag{78}$$

which is a twice continuously differentiable function on $\mathcal{E}_t$. Similarly, if $\mathrm{tr}(\boldsymbol{\varepsilon}) < 0$:

$$W_0 = W_0^- = \frac{1}{2}\lambda_c \mathrm{tr}^2(\boldsymbol{\varepsilon}) + \mu \mathrm{tr}(\boldsymbol{\varepsilon}^2), \tag{79}$$

which is a twice continuously differentiable function on $\mathcal{E}_c$. The strain energy is also continuous across the interface between the tension and the compression sub-domains (i.e., when $\mathrm{tr}(\boldsymbol{\varepsilon}) = 0$) because:

$$W_0 \,|_{\mathrm{tr}(\boldsymbol{\varepsilon})=0} = W_0^+ \,|_{\mathrm{tr}(\boldsymbol{\varepsilon})=0} = W_0^- \,|_{\mathrm{tr}(\boldsymbol{\varepsilon})=0} = \mu \mathrm{tr}(\boldsymbol{\varepsilon}^2). \tag{80}$$

Following the same procedure, we now demonstrate that the stress tensor, $\boldsymbol{\sigma} = \frac{\partial W_0}{\partial \varepsilon}$, is also continuous across the entire strain space. According to (60) and $(61)_1$, if $\mathrm{tr}(\varepsilon) > 0$:

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = \boldsymbol{\sigma}^+ = \lambda_t \mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon}, \tag{81}$$

which is continuously differentiable on $\mathcal{E}_t$. Similarly, if $\mathrm{tr}(\boldsymbol{\varepsilon}) < 0$:

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = \boldsymbol{\sigma}^- = \lambda_c \mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon}, \tag{82}$$

which is continuously differentiable on $\mathcal{E}_c$. Now, across the interface between the tension and the compression sub-domains (i.e., when $\mathrm{tr}(\boldsymbol{\varepsilon}) = 0$), we have:

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) \,|_{\mathrm{tr}(\boldsymbol{\varepsilon})=0} = \boldsymbol{\sigma}^+ \,|_{\mathrm{tr}(\boldsymbol{\varepsilon})=0} = \boldsymbol{\sigma}^- \,|_{\mathrm{tr}(\boldsymbol{\varepsilon})=0} = 2\mu\boldsymbol{\varepsilon}, \tag{83}$$

which concludes our proof of continuity. The elasticity tensor in $(61)_2$ can be discontinuous across the interface between the tension and the compression subdomains, but it is nonetheless piecewise continuous on $\mathcal{E}_t$ and $\mathcal{E}_c$, respectively. This discontinuity in the elasticity tensor has not shown any adverse effect in terms of numerical instabilities and has not caused convergence issues.

Appendix D: PolyScript

```
%------------------------------- PolyStress ------------------------------%
% Ref: O Giraldo-Londoño, GH Paulino, "PolyStress: A Matlab implementation%
% for topology optimization with local stress constraints using the       %
% augmented Lagrangian method", Structural and Multidisciplinary          %
% Optimization, DOI 10.1007/s00158-020-02760-8                            %
%-------------------------------------------------------------------------%
clear; clc; close all
restoredefaultpath; addpath(genpath('./')); %Use all folders and subfolders
set(0,'defaulttextinterpreter','latex')
%% ------------------------------------------------------------ CREATE Mesh
[Node,Element,Supp,Load] = Mesh_L_bracket(10000);
NElem = size(Element,1); % Number of elements
%% ---------------------------------------------------- CREATE 'fem' STRUCT
E0 = 70e3; % E0 in MPa
G = E0/2.5; Et = E0; Ec = E0;  % 0<=(Et,Ec)<=3*G; %Material props. (linear)
fem = struct(...
  'NNode',size(Node,1),...      % Number of nodes
  'NElem',size(Element,1),...   % Number of elements
  'Node',Node,...               % [NNode x 2] array of nodes
  'Element',{Element},...       % [NElement x Var] cell array of elements
  'Supp',Supp,...               % Array of supports
  'Load',Load,...               % Array of loads
  'Passive',[],...              % Passive elements
  'Thickness',1,...             % Element thickness
  'MatModel','Bilinear',...     % Material model ('Bilinear','Polynomial')
  'MatParam',[Et,Ec,G],...      % Material parameters for MatModel
  'SLim',100,...                % Stress limit
  'TolR', 1e-8, ...             % Tolerance for norm of force residual
  'MaxIter', 15, ...            % Max NR iterations per load step
  'MEX', 'No');                  % Tag to use MEX functions in NLFEM routine
%% ---------------------------------------------------- CREATE 'opt' STRUCT
R = 0.05; q = 3; % Filter radius and filter exponent
p = 3.5; eta0 = 0.5;
m = @(y,B)MatIntFnc(y,'SIMP-H1',[p,B,eta0]);
P = PolyFilter(fem,R,q);
zIni = 0.5*ones(size(P,2),1);
opt = struct(...
  'zMin',0.0,...              % Lower bound for design variables
  'zMax',1.0,...              % Upper bound for design variables
  'zIni',zIni,...             % Initial design variables
  'MatIntFnc',m,...           % Handle to material interpolation fnc.
  'contB',[5,1,1,10],...      % Threshold projection continuation params.
  'P',P,...                   % Matrix that maps design to element vars.
  'Tol',0.002,...             % Convergence tolerance on design vars.
  'TolS',0.003,...            % Convergence tolerance on stress constraints
  'MaxIter',150,...           % Maximum number of AL steps
  'MMA_Iter',5,...            % Number of MMA iterations per AL step
  'lambda0',zeros(NElem,1),...% Initial Lagrange multiplier estimators
  'mu0',10,...                % Initial penalty factor for AL function
  'mu_max',10000,...          % Maximum penalty factor for AL function
  'alpha',1.1,...             % Penalty factor update parameter
  'Move',0.15,...             % Allowable move step in MMA update scheme
  'Osc',0.2,...               % Osc parameter in MMA update scheme
  'AsymInit',0.2,...          % Initial asymptote in MMA update shecme
  'AsymInc',1.2,...           % Asymptote increment in MMA update scheme
  'AsymDecr',0.7...           % Asymptote decrement in MMA update scheme
   );
%% --------------------------------------------------------- RUN 'PolyStress'
fem = preComputations(fem); % Run preComputations before running PolyStress
[z,V,fem] = PolyStress(fem,opt);
%-------------------------------------------------------------------------
```

Appendix E: PolyStress

```
%------------------------------- PolyStress ------------------------------%
% Ref: O Giraldo-Londoño, GH Paulino, "PolyStress: A Matlab implementation%
% for topology optimization with local stress constraints using the       %
% augmented Lagrangian method", Structural and Multidisciplinary          %
% Optimization, DOI 10.1007/s00158-020-02760-8                            %
%-------------------------------------------------------------------------%
function [z,V,fem] = PolyStress(fem,opt)
%% Initialize variables
Tol = opt.Tol*(opt.zMax-opt.zMin); TolS = opt.TolS;
z = opt.zIni; P = opt.P; Iter = 0; SM = 10;
lambda = opt.lambda0; mu = opt.mu0;
BFreq = opt.contB(1); B = opt.contB(2);
Binc = opt.contB(3); Bmax = opt.contB(4);
[E,dEdy,V,dVdy] = opt.MatIntFnc(P*z,B);
Eid = setdiff((1:fem.NElem)',fem.Passive); % Element ID for active elements
z(fem.Passive) = 1; % Set z = 1 for passive elements
%% Plot initial density distribution and von Mises stress map
figure; [hV,hS] = InitialPlot(fem,V,0*V);
%% MMA parameters
Change = 2*opt.Tol; zold1 = z; zold2 = z;
L = opt.zMin.*ones(fem.NElem,1); U = opt.zMax.*ones(fem.NElem,1);
AsymInc = opt.AsymInc; AsymDecr = opt.AsymDecr;
%% Optimization iterations
tic;
while (Iter<opt.MaxIter) && (Change>Tol || max(SM)>1+TolS) %AL steps
  Iter = Iter + 1;
  for j = 1:opt.MMA_Iter %MMA iterations per AL step
    %% Compute cost functionals and sensitivity analysis
    [~,dJdz,f,h,fem] = AL_Function(fem,E,dEdy,V,dVdy,lambda,mu,P);
    %% Update design variable and analysis parameters
    [z(Eid),zold1(Eid),zold2(Eid),L(Eid),U(Eid),AsymInc,AsymDecr,Change]...
        = MMA_unconst(dJdz(Eid),z(Eid),zold1(Eid),zold2(Eid),...
                      L(Eid),U(Eid),Iter,AsymInc,AsymDecr,opt);
    [E,dEdy,V,dVdy] = opt.MatIntFnc(P*z,B);
    SM = E.*fem.VM_Stress0/fem.SLim; % Normalized stress measure
    fprintf(['It:%3i_%1i Obj: %1.3f Max_VM: %1.3f |dJ|: %1.3f ',...
        'Ch/Tol: %1.3f\n'],Iter,j,f,max(SM),norm(dJdz),Change/Tol);
    if (Change<=Tol && max(SM)<=1+TolS), break; end
  end
  %% Update lagrange multiplier estimators and penalty parameter
  lambda = lambda + mu*h;
  mu = min(opt.alpha*mu,opt.mu_max);
  %% Update material interpolation function
  if mod(Iter,BFreq)==0; B = min(B+Binc,Bmax); end
  %% Update density and stress plots
  set(hV,'FaceColor','flat','CData',1-V); drawnow
  set(hS,'FaceColor','flat','CData',SM); drawnow
end
t = toc;
if t<=60, fprintf('Optimization time: %i seconds \n', round(t))
elseif t<=3600, fprintf('Optimization time: %1.1f minutes \n', t/60)
elseif t<=86400, fprintf('Optimization time: %1.1f hours \n', t/3600)
else, fprintf('Optimization time: %1.1f days \n', t/86400)
end
%% ----------------------------- NORMALIZED AL FUNCTION AND ITS SENSITIVITY
function [J,dJdz,f,h,fem] = AL_Function(fem,E,dEdy,V,dVdy,lambda,mu,P)
[f,dfdV,dfdE,fem] = ObjectiveFnc(fem,E,V);
```

```
[h,Penal,dPenaldV,dPenaldE,fem] = PenalFnc(fem,E,V,lambda,mu);
N = fem.NElem;
J = f + Penal/N; %Normalized AL function
dJdz = P'*(dEdy.*(dfdE+dPenaldE./N)+dVdy.*(dfdV+dPenaldV./N)); %Sensitivity
%% ----------------------------------- OBJECTIVE FUNCTION (VOLUME FRACTION)
function [f,dfdV,dfdE,fem] = ObjectiveFnc(fem,E,V)
f = sum(fem.ElemArea.*V)/sum(fem.ElemArea); %Mass ratio
dfdV = fem.ElemArea./sum(fem.ElemArea);
dfdE = zeros(size(E));
%% ---------------------------------- PENALTY FUNCTION (STRESS CONSTRAINTS)
function [h,Penal,dPenaldV,dPenaldE,fem] = PenalFnc(fem,E,V,lambda,mu)
[U,fem] = NLFEM(fem,E); %Run nonlinear FEM routine
[fem.VM_Stress0,dVM_dU] = von_Mises_Stress(fem,U); %VM_Stress and Sensit.
dhdVM = zeros(size(V)); dPenaldV = dhdVM; dPenaldE = dhdVM;
s = fem.VM_Stress0/fem.SLim-1;
g = E.*(s.^3+s);     %Polynomial vanishing constraint
h = max(g,-lambda./mu);
Penal = sum(lambda.*h + mu/2.*h.^2); %Penalty term of AL function
a1 = h==g; %Find entries of h==g for sensitivity computation
dhdVM(a1) = E(a1).*(3.*s(a1).^2+1).*1/fem.SLim; %Sensit. of h wrt VM_Stress
dPenaldVM = (lambda+mu.*h).*dhdVM; % Sensit. of penalty term wrt VM_Stress
dPenaldE(a1) = (lambda(a1)+mu.*h(a1)).*(s(a1).^3+s(a1));
% Adjoint method for sensitivity computation
Adjoint_Load = -accumarray(fem.eDof(:),dVM_dU(:).*dPenaldVM(fem.DofE));
Adjoint_Vector(fem.s) = fem.L'\(fem.L\Adjoint_Load(fem.FreeDofs(fem.s),:));
dFintdE = sparse(fem.eDof,fem.DofE,fem.f_NL);
dPenaldE = dPenaldE + (Adjoint_Vector*dFintdE(fem.FreeDofs,:))';
%% --------------------------------- VON MISES STRESS AND SENSITIVITY WRT U
function [VM_Stress,dVM_dU] = von_Mises_Stress(fem,U)
V = [1 -1/2 0; -1/2 1 0; 0 0 3]; % von Mises matrix
ElemU = U(fem.eDof);    %ELement displacement vectors
ee_elem = fem.B0*ElemU; %Strains at the cenroid of all elements
ee_elem = reshape(ee_elem,3,[]); %Strains at the cenroid of all elements
[Cauchy_S, D0] = material_model(fem.MatModel,fem.MatParam,ee_elem);
D0 = sparse(fem.rowD,fem.colD,reshape(D0,9*fem.NElem,1));
DB = D0*fem.B0;
VM_Stress = max(sqrt(sum(Cauchy_S.*(V*Cauchy_S))),eps)'; % von Mises stress
dVM_dCauchy = (V*Cauchy_S)./repmat(VM_Stress',3,1);
dVM_dU = DB'*dVM_dCauchy(:); % Sensitivity of VM_Stress wrt U
%% -------------------------------------- MMA UPDATE SCHEME (UNCONSTRAINED)
function [zNew,zold1,zold2,L,U,AsymInc,AsymDecr,Change] = ...
           MMA_unconst(dfdz,z,zold1,zold2,L,U,Iter,AsymInc,AsymDecr,opt)
zMin = opt.zMin; zMax = opt.zMax;
move = opt.Move*(zMax-zMin); Osc=opt.Osc; AsymInit = opt.AsymInit;
xmin = max(zMin,z-move); xmax = min(zMax,z+move);
% Compute asymptotes L and U:
AsymInc = min(1+Osc,AsymInc);
AsymDecr = max(1-2*Osc,AsymDecr);
if Iter<=2
  L = z - AsymInit*(xmax-xmin);  U = z + AsymInit*(xmax-xmin);
else
  sgn = (z-zold1).*(zold1-zold2);
  s = ones(size(z)); s(sgn>0) = AsymInc; s(sgn<0) = AsymDecr;
  L = z - s.*(zold1 - L); U = z + s.*(U - zold1);
end
% Compute bounds alpha and beta
alpha = 0.9*L + 0.1*z;   beta = 0.9*U + 0.1*z;
alpha = max(xmin,alpha); beta = min(xmax,beta);
% Solve unconstrained subproblem
```

```
feps = 0.000001;
p = (U-z).^2.*(max(dfdz,0)+0.001*abs(dfdz)+feps./(U-L));
q = (z-L).^2.*(-min(dfdz,0)+0.001*abs(dfdz)+feps./(U-L));
zCnd = (L.*p - U.*q + (U - L).*sqrt(p.*q))./(p - q);
zNew = max(alpha,min(beta,zCnd));
zold2 = zold1; zold1=z;
Change = sum(abs(zNew-z))/length(z);
%% ----------------------------------------------------------- INITIAL PLOT
function [handle1,handle2] = InitialPlot(fem,z01,z02)
ElemNodes = cellfun(@length,fem.Element); %Number of nodes of each element
Faces = NaN(fem.NElem,max(ElemNodes));    %Populate Faces with NaN
for el = 1:fem.NElem; Faces(el,1:ElemNodes(el)) = fem.Element{el}(:); end
ax1 = subplot(1,2,1); title('Element Densities');
patch('Faces',Faces,'Vertices',fem.Node,'FaceVertexCData',0.*z01,...
      'FaceColor','flat','EdgeColor','k','linewidth',1.5);
handle1 = patch('Faces',Faces,'Vertices',fem.Node,'FaceVertexCData',...
                1-z01,'FaceColor','flat','EdgeColor','none');
axis equal; axis off; axis tight; colormap(ax1,gray); caxis([0 1]);
hsp1 = get(gca, 'Position'); % Get position of subplot 1
ax2 = subplot(1,2,2); title('Normalized von Mises Stress')
handle2 = patch('Faces',Faces,'Vertices',fem.Node,'FaceVertexCData',...
                z02,'FaceColor','flat','EdgeColor','none');
axis equal; axis off; axis tight; colormap(ax2,'jet'); c = colorbar;
w = get(c,'Position');
hsp2 = get(gca, 'Position'); % Get position of subplot 2
set(ax1, 'Position', [hsp1(1)-w(3), hsp1(2:end)]);
set(ax2, 'Position', [hsp2(1)-2*w(3), hsp2(2),  hsp1(3:4)]);
drawnow;
%-------------------------------------------------------------------------%
```

Appendix F: NLFEM

```
%-------------------------------- PolyStress ------------------------------%
% Ref: O Giraldo-Londoño, GH Paulino, "PolyStress: A Matlab implementation%
% for topology optimization with local stress constraints using the       %
% augmented Lagrangian method", Structural and Multidisciplinary          %
% Optimization, DOI 10.1007/s00158-020-02760-8                            %
%-------------------------------------------------------------------------%
function [U,fem] = NLFEM(fem,E)
U = zeros(2*fem.NNode,1); if ~isfield(fem,'U'); fem.U=U; end
if (strcmp(fem.MatModel,'Bilinear')==1)&&(fem.MatParam(1)==fem.MatParam(2))
  K = sparse(fem.i,fem.j,E(fem.e).*fem.k0); % Assemble stiffness matrix
  K = (K+K')/2;
  K = K(fem.FreeDofs,fem.FreeDofs);
  [U(fem.FreeDofs),L,s] = SolveLinSys(K,fem.Fext(fem.FreeDofs));
  fem.L = L; fem.s = s; % Store Cholesky decomposition information
  fem.f_NL = sparse(fem.iK0,fem.jK0,fem.k0)*U(fem.eDof);
else
  U = fem.U; % Use previously converged U as initial guess
  [K,~,Res,~,fem] = GlobalK(fem,U,E); %Initial stiffness mtrx. & Res vector
  nRes0 = norm(fem.Fext); nRes = nRes0; % Initial norm of force residual
  % NEWTON-RAPHSON ITERATIONS
  Iter = 0; % Initialize Newton-Raphson iteration counter
  while (nRes>fem.TolR*nRes0 && Iter<=fem.MaxIter)
    [Delta_U, L, s] = SolveLinSys(K,Res);
    [K,~,Res,nRes,fem,U] = LineSearch(U,Delta_U,nRes,-K*Res./nRes,fem,E);
    Iter = Iter+1;
  end
  fem.L = L; fem.s = s; % Store Cholesky decomposition information
  fem.U = U; % Store converged displacement vector
end
%% -------------------------------------------- WEAK LINE SEARCH ALGORITHM
function [K,Fint,Res,nRes,fem,Un] = LineSearch(U_temp,p,phix,gphix,fem,E)
sigma = 1e-4; alpha_min = 1e-5; alpha_max = 1; MaxIt = 4;
pgphi = p'*gphix;
alpha = alpha_max;
Un = U_temp;
Un(fem.FreeDofs) = U_temp(fem.FreeDofs) + alpha*p;
[K,Fint,Res,nRes,fem] = GlobalK(fem,Un,E); phixn=nRes;
it = 0; % Start iteration counter
while (phixn>phix+sigma*alpha*pgphi && alpha>alpha_min && it<=MaxIt)
  mu = -0.5 * pgphi * alpha / (phixn - phix - alpha * pgphi );
  if mu < 0.1
    mu = 0.5; % Do not trust quadratic interpolation from far away
  end
  alpha = mu*alpha; % New value of line search parameter alpha
  Un(fem.FreeDofs) = U_temp(fem.FreeDofs) + alpha*p; % New disp. vector
  [K,Fint,Res,nRes,fem] = GlobalK(fem,Un,E); phixn=nRes;
  it = it+1;
end
%% ---------------------- GLOBAL STIFFNESS MATRIX AND RESIDUAL FORCE VECTOR
function [K,Fint,Res,nRes,fem] = GlobalK(fem,U,E)
if strcmp(fem.MEX,'Yes')==1
  Compile_mex_file;
  [k0, f_NL] = Global_K_compile_mex(fem.ElemNDof,fem.Element,...
                 fem.MatModel,fem.MatParam,fem.Thickness,E,...
                 fem.W,fem.dNdxi,fem.Node,U);
elseif strcmp(fem.MEX,'No')==1
  [k0, f_NL] = Global_K_compile(fem.ElemNDof,fem.Element,...
                fem.MatModel,fem.MatParam,fem.Thickness,E,...
                fem.W,fem.dNdxi,fem.Node,U);
else
  error('You should use fem.MEX=Yes or fem.MEX=No');
end
fem.f_NL = f_NL; % Store this for sensitivity computations
f0 = E(fem.DofE).*f_NL;
K = sparse(fem.i,fem.j,E(fem.e).*k0); % Assemble stiffness matrix
K = (K+K')/2; % Symmetrize stiffness matrix
K = K(fem.FreeDofs,fem.FreeDofs);
Fint = sparse(fem.eDof,ones(sum(fem.ElemNDof),1),f0,2*fem.NNode,1);
Res = fem.Fext(fem.FreeDofs)-Fint(fem.FreeDofs); nRes = norm(Res);
%% ----------------------- SOLVE LINEAR SYSTEM USING CHOLESKY DECOMPOSITION
function [Delta_U, L_T, s] = SolveLinSys(K,Res)
[L_T,~,s] = chol(K,'lower','vector');
Delta_U(s,:) = L_T'\(L_T\Res(s,:));
%------------------------------------------------------------------------%
```

References

Bendsøe MP (1989) Optimal shape design as a material distribution problem. Struct Optim 1(4):193–202

Bendsøe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 93:291–318

Bendsøe MP (1995) Optimization of structural topology, shape, and material. Springer, Berlin

Bendsøe MP, Sigmund O (2003) Topology optimization: theory, methods and applications. Springer, Berlin

Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Scientific, Nashua

Borrvall T, Petersson J (2001) Topology optimization using regularized intermediate density control. Comput Methods Appl Mech Eng 190(37-38):4911–4928

Bourdin B (2001) Filters in topology optimization. Int J Numer Methods Eng 50(9):2143–2158

Bruggi M (2008) On an alternative approach to stress constraints relaxation in topology optimization. Struct Multidiscip Optim 36(2):125–141

Bruggi M, Duysinx P (2012) Topology optimization for minimum weight with compliance and stress constraints. Struct Multidiscip Optim 46(3):369–384

Cheng GD, Jiang Z (1992) Study on topology optimization with stress constraints. Eng Optim 20(2):129–148

Chi H, Ramos DL, Ramos AS Jr, Paulino GH (2019) On structural topology optimization considering material nonlinearity: Plane strain versus plane stress solutions. Adv Eng Softw 131:217–231

Chu S, Gao L, Xiao M, Luo Z, Li H, Gui X (2018) A new method based on adaptive volume constraint and stress penalty for stress-constrained topology optimization. Struct Multidiscip Optim 57(3):1163–1185

Curnier A, He QC, Zysset P (1994) Conewise linear elastic materials. J Elast 37(1):1–38

da Silva GA, Beck AT, Cardoso EL (2018) Topology optimization of continuum structures with stress constraints and uncertainties in loading. Int J Numer Methods Eng 113(1):153–178

da Silva GA, Beck AT, Sigmund O (2019a) Stress-constrained topology optimization considering uniform manufacturing uncertainties. Comput Methods Appl Mech Eng 344:512–537

da Silva GA, Beck AT, Sigmund O (2019b) Topology optimization of compliant mechanisms with stress constraints and manufacturing error robustness. Comput Methods Appl Mech Eng 354:397–421

De Leon DM, Alexandersen J, Fonseca JS, Sigmund O (2015) Stress-constrained topology optimization for compliant mechanism design. Struct Multidiscip Optim 52(5):929–943

Duysinx P, Bendsøe MP (1998) Topology optimization of continuum structures with local stress constraints. Int J Numer Methods Eng 43(8):1453–1478

Emmendoerfer HJr, Fancello EA (2014) A level set approach for topology optimization with local stress constraints. Int J Numer Methods Eng 99(2):129–156

Emmendoerfer HJr, Fancello EA (2016) Topology optimization with local stress constraint based on level set evolution via reaction-diffusion. Comput Methods Appl Mech Eng 305:62–88

Emmendoerfer HJr, Silva ECN, Fancello EA (2019) Stress-constrained level set topology optimization for design-dependent pressure load problems. Comput Methods Appl Mech Eng 344:569–601

Fan Z, Xia L, Lai W, Xia Q, Shi T (2019) Evolutionary topology optimization of continuum structures with stress constraints. Struct Multidiscip Optim 59(2):647–658

Fancello EA (2006) Topology optimization for minimum mass design considering local failure constraints and contact boundary conditions. Struct Multidiscip Optim 32(3):229–240

Feng ZQ, Peyraut F, He QC (2006) Finite deformations of Ogden's materials under impact loading. Int J Nonlin Mech 41(4):575–585

Giraldo-Londoño O, Paulino GH (2020) A unified approach for topology optimization with local stress constraints considering various failure criteria: von Mises, Drucker–Prager, Tresca, Mohr–Coulomb, Bresler–Pister, and William–Warnke. Proceedings of the Royal Society A. 476:20190861

Guest JK, Prévost JH, Belytschko T (2004) Achieving minimum length scale in topology optimization using nodal design variables and projection functions. Int J Numer Methods Eng 61(2):238–254

Guo X, Zhang WS, Wang MY, Wei P (2011) Stress-related topology optimization via level set approach. Comput Methods Appl Mech Eng 200(47-48):3439–3452

Holmberg E, Torstenfelt B, Klarbring A (2013a) Global and clustered approaches for stress constrained topology optimization and deactivation of design variables. In: 10th world congress on structural and multidisciplinary optimization

Holmberg E, Torstenfelt B, Klarbring A (2013b) Stress constrained topology optimization. Struct Multidiscip Optim 48(1):33–47

Kiyono C, Vatanabe S, Silva E, Reddy J (2016) A new multi-p-norm formulation approach for stress-based topology optimization design. Compos Struct 156:10–19

Kreisselmeier G, Steinhauser R (1979) Systematic control design by optimizing a vector performance index. In: IFAC proceedings volumes, vol 12, pp 113–117. IFAC Symposium on computer Aided Design of Control Systems, Zurich, Switzerland, 29-31 August

Le C, Norato J, Bruns T, Ha C, Tortorelli D (2010) Stress-based topology optimization for continua. Struct Multidiscip Optim 41(4):605–620

Lee E, James KA, Martins JRRA (2012) Stress-constrained topology optimization with design-dependent loading. Struct Multidiscip Optim 46(5):647–661

Lee K, Ahn K, Yoo J (2016) A novel p-norm correction method for lightweight topology optimization under maximum stress constraints. Comput Struct 171:18–30

Lian H, Christiansen AN, Tortorelly DA, Sigmund O (2017) Combined shape and topology optimization for minimization of maximal von mises stress. Struct Multidiscip Optim 55(5):1541–1557

Liu H, Yang D, Hao P, Zhu X (2018) Isogeometric analysis based topology optimization design with global stress constraint. Comput Methods Appl Mech Eng 342:625–652

Luo Y, Wang MY, Kang Z (2013) An enhanced aggregation method for topology optimization with local stress constraints. Comput Methods Appl Mech Eng 254:31–41

Nocedal J, Wright SJ (2006) Numerical optimization, 2nd edn. Springer, Berlin

Ogden RW (1972) Large deformation isotropic elasticity–on the correlation of theory and experiment for incompressible rubberlike solids. Proc Roy Soc Lond Math Phys Sci 326(1567):565–584

Paris J, Navarrina F, Colominas I, Casteleiro M (2009) Topology optimization of continuum structures with local and global stress constraints. Struct Multidiscip Optim 39(4):419–437

Paris J, Navarrina F, Colominas I, Casteleiro M (2010) Block aggregation of stress constraints in topology optimization of structures. Adv Eng Softw 41(3):433–441

Park YK (1995) Extensions of optimal layout design using the homogenization method. Ph.D thesis, University of Michigan, Ann Arbor

Pereira A, Talischi C, Paulino GH, Menezes IF, Carvalho MS (2016) Fluid flow topology optimization in polytop: stability and computational implementation. Struct Multidiscip Optim 54(5):1345–1364

Pereira JT, Fancello EA, Barcellos CS (2004) Topology optimization of continuum structures with material failure constraints. Struct Multidiscip Optim 26(1-2):50–66

Rozvany GI, Zhou M, Birker T (1992) Generalized shape optimization without homogenization. Structural Optimization 4(3–4):250–252

Sanders ED, Pereira A, Aguiló MA, Paulino GH (2018) Polymat: an efficient matlab code for multi-material topology optimization. Struct Multidiscip Optim 58(6):2727–2759

Senhora FV (2019) Personal communication

Senhora FV, Giraldo-Londoño O, Menezes IFM, Paulino GH (2020) Topology optimization with local stress constraints: a stress aggregation-free approach. Struct Multidiscip Optim 62(4):1639–1668

Svanberg K (1987) The method of moving asymptotes—A new method for structural optimization. Int J Numer Methods Eng 24(2):359–373

Talischi C, Paulino GH, Pereira A, Menezes IFM (2012a) Polymesher: a general-purpose mesh generator for polygonal elements written in Matlab. Struct Multidiscip Optim 45(3):309–328

Talischi C, Paulino GH, Pereira A, Menezes IFM (2012b) Polytop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. Struct Multidiscip Optim 45(3):329–357

Verbart A, Langelaar M, van Keulen F (2016) Damage approach: a new method for topology optimization with local stress constraints. Struct Multidiscip Optim 53(5):1081–1098

Wang F, Lazarov BS, Sigmund O (2011) On projection methods, convergence and robust formulations in topology optimization. Struct Multidiscip Optim 43(6):767–784

Xia L, Zhang L, Xia Q, Shi T (2018) Stress-based topology optimization using bi-directional evolutionary structural optimization method. Comput Methods Appl Mech Eng 333:356–370

Xia Q, Shi T, Liu S, Wang MY (2012) A level set solution to the stress-based structural shape and topology optimization. Comput Struct 90:55–64

Yang RJ, Chen CJ (1996) Stress-based topology optimization. Struct Optim 12(2):98–105

Zhang WS, Guo X, Wang MY, Wei P (2013) Optimal topology design of continuum structures with stress concentration alleviation via level set method. Int J Numer Methods Eng 93(9):942–959

Zhou M, Rozvany GIN (1991) The COC algorithm, part II: topological, geometrical and generalized shape optimization. Comput Methods Appl Mech Eng 89(1–3):309–336

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.