Structural and Multidisciplinary Optimization (2021) 64:957–990
https://doi.org/10.1007/s00158-021-02859-6

EDUCATIONAL PAPER

PolyDyna: a Matlab implementation for topology optimization of structures subjected to dynamic loads

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

Received: 10 June 2020 / Revised: 2 January 2021 / Accepted: 19 January 2021 / Published online: 3 July 2021

Abstract

We present a Matlab implementation for topology optimization of structures subjected to dynamic loads. The code, which we name PolyDyna, is built on top of PolyTop—a Matlab code for static compliance minimization based on polygonal finite elements. To solve the structural dynamics problem, we use the HHT-α method, which is a generalization of the classical Newmark-β method. In order to handle multiple regional volume constraints efficiently, PolyDyna uses a variation of the ZPR design variable update scheme enhanced by a sensitivity separation technique, which enables it to solve non-self-adjoint topology optimization problems. We conduct the sensitivity analysis using the adjoint method with the "discretize-then-differentiate" approach, such that the sensitivity analysis is consistently evaluated on the discretized system (both in space and time). We present several numerical examples, which are explained in detail and summarized in a library of benchmark problems. PolyDyna is intended for educational purposes and the complete Matlab code is provided as electronic supplementary material.

Keywords Topology optimization · Compliance minimization · HHT-α method · Newmark-β method · Elastodynamics · Sensitivity separation · ZPR update scheme

1 Introduction

We present PolyDyna, a Matlab implementation for topology optimization of structures subjected to dynamic loads built on top of PolyTop (Talischi et al. 2012b). The code adds to a series of educational computer codes for topology optimization on unstructured polygonal finite element meshes, which have been developed to solve a variety of problems. The first code in the series is PolyTop (Talischi et al. 2012b), which solves compliance minimization problems. PolyTop comes with a companion code called PolyMesher (Talischi et al. 2012a), which is used for the polygonal finite element mesh generation. The structure of PolyTop is modular, such that the analysis and optimization routines are decoupled and can be easily modified, as needed, to solve other topology optimization problems. For instance, Pereira et al. (2016) modified PolyTop's finite element analysis routine to solve topology optimization problems for power dissipation in Stokes flow. Similarly, Sanders et al. (2018) applied a few modifications to PolyTop to solve compliance minimization problems for multi-material structures. More recently, Giraldo-Londoño and Paulino (2020b) modified both the analysis and optimization routines in PolyTop to solve topology optimization problems with local stress constraints using the augmented Lagrangian method (Bertsekas 1999; Nocedal and Wright 2006). They used a variation of the AL-based formulation by Senhora et al. (2020) together with the polynomial vanishing constraint by Giraldo-Londoño and Paulino (2020c) to solve problems with a large number of local stress constraints.

Here, we extend PolyTop to PolyDyna, such that we can design structures subjected to general dynamic loading, for minimum dynamic compliance, minimum strain energy, or minimum squared displacement of a target degree of freedom with one or more volume constraints. Our main goal is to provide an educational code and all the

Responsible Editor: Gengdong Cheng

✉ 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

associated theoretical and computational details needed to understand and implement dynamic topology optimization problems solved in the time domain. `PolyDyna` accommodates dynamic loading that can change magnitude, direction, or location over a given period of time, as well as ground acceleration. We replace `PolyTop`'s finite element analysis routine with one tailored to solve structural dynamics problems. The new routine uses the HHT-α method (Hilber et al. 1977), which is a generalization of the Newmark-β method that maintains unconditional stability for a suitable choice of parameters. The HHT-α method is suitable for problems with many degrees of freedom because the α parameter tends to dampen high frequency modes with negligible effect on lower frequency modes. Furthermore, we adopt a *discretize-then-differentiate* approach for adjoint sensitivity analysis (Jensen et al. 2014) to eliminate consistency errors observed in the commonly-used *differentiate-then-discretize* approach. In order to efficiently handle multiple regional volume constraints, we adopt the ZPR design variable update scheme (Zhang et al. 2018) augmented by a sensitivity separation scheme (Jiang et al. 2021) that yields a non-monotonous convex approximation such that we can treat sensitivities of unrestricted sign.

The remainder of this paper is organized as follows. Section 2 discusses several studies related to topology optimization for dynamic problems, while Section 3 details the topology optimization statement leading to the implementation of `PolyDyna`. We present the HHT-α method in Section 4, followed by the details of the consistent sensitivity analysis in Section 5. Next, Section 6 explains the derivation of the ZPR-based design variable update scheme. We discuss several details of the implementation of `PolyDyna` in Section 7, followed by five numerical examples in Section 8. Afterwards, we provide several appendices containing some parts of the code and add a library of examples, which can be used as benchmark problems. The entire Matlab code is provided as Electronic Supplementary Material.

2 Brief literature review and related work

Structures under real operating conditions are often subjected to dynamic loading that greatly affects the structural response and must be considered in the design process. As a means of obtaining efficient and organic designs, topology optimization has emerged as a powerful computational tool for obtaining optimized structural layouts that minimize some measure of performance of a system while satisfying a set of imposed constraints. For instance, constraints can be imposed to limit the amount of material, maximum member size, material strength, among others. Although the majority of studies in topology optimization have focused on static problems (e.g., minimum compliance problems or strain energy minimization problems under static loads), a considerable amount of research has focused on topology optimization considering dynamic loads.

In the realm of dynamic topology optimization, design problems have been formulated either in the frequency domain or in the time domain. Topology optimization formulations in the frequency domain are useful to design structures subjected to periodic loads or when modal quantities are of interest. Frequency-domain topology optimization problems have found applications in a variety of design problems. For example, Filipov et al. (2016) formulated a multi-resolution topology optimization approach with polygonal finite elements and applied it to eigenfrequency optimization and dynamic compliance minimization. A different application is that by Giraldo-Londoño and Paulino (2020a), who optimized the microstructure of multi-phase viscoelastic materials with tailored energy dissipation for a given frequency or for a range of frequencies. Overall, the idea of frequency-domain problems is to minimize an objective function measuring the structural dynamic response. For instance, typical objective functions are the mean dynamic compliance, displacement amplitude at a given point or at a set of points, or the natural frequency of the system (e.g., see Ma et al. 1993; Ma et al. 1995; Jog, 2002; Yoon, 2010; Shue et al. 2011; Liu et al. 2015; Liu et al. 2017). Recently, Martin and Deierlein (2020) defined a new objective function based on the sum of modal compliances to design the lateral bracing system of tall buildings.

When the dynamic loads are not periodic, the dynamic topology optimization problems are formulated in the time domain. These types of problems are more computationally expensive than frequency-domain problems because the structural problem requires time integration. Nevertheless, a number of researchers consider time domain methods when time-dependent quantities are of interest. For example, in the time domain, Min et al. (1999) minimized the mean dynamic compliance of linear structures and Shobeiri (2019) that of nonlinear structures. Turteltaub (2005) minimized the time-average stress energy of two-phase functionally graded composites, Dahl et al. (2008) minimized nodal displacements to generate band-gap structures, and Zhao and Wang (2017) both minimized a target displacement and dynamic compliance over a time interval. All of the above consider a single, time-independent constraint. In contrast, Rong et al. (2000) considered mass as the objective function and imposed constraints on mean square displacements at each point in the structure and solved the problem using the evolutionary structural optimization method. On a related note, Verbart and Stolpe (2018) considered time-dependent constraints, but solved a sequence of sub-problems using a small sub-set of the constraints.

Another approach used to solve dynamic topology optimization problems in the time domain is the equivalent static load method introduced by Choi and Park (2002). The method defines a set of equivalent static loads at every time step that generates the same displacement field as that produced during the dynamic analysis. The static loads are used to solve an equivalent static problem with multiple load cases (i.e., an equivalent static load vector per time step). To alleviate the computational burden of having too many load cases, they only consider the most critical static load vectors and neglect the rest. An extension for multi-body dynamic systems is presented by Kang et al. (2005). More recently, Jang et al. (2012) used the equivalent static load method, but instead of minimizing the mean dynamic compliance (Choi and Park 2002), they aimed to minimize the peak compliance values in a time window. This difference in the objective function leads to a formulation that prevents the optimized structure from experiencing large deformation at a certain time range. Lee and Park (2015) extended the work by Jang et al. (2012) and applied the method for dynamic topology optimization of nonlinear structures.

As indicated previously, solving dynamic topology optimization problems in the time domain is computationally expensive. In order to alleviate this computational burden, some researchers have resorted to model reduction methods, which solve the dynamic analysis problem using, e.g., a mode superposition method. For instance, Zhao and Wang (2016) explored the effectiveness of two model reduction methods to solve dynamic topology optimization problems. One of them is the mode displacement method, which solves the dynamic analysis problem using a linear combination of the eigenvectors associated with the first few eigenfrequencies. The other method is the mode acceleration method, which is similar to the first, yet it adds a correction factor that corresponds to a pseudo-static displacement vector. Although effective, these model reduction methods can be problematic, especially when high-frequency modes are relevant. In a study by Hooijkamp and van Keulen (2018), a model reduction method was also employed to solve topology optimization problems of linear transient thermomechanical problems. Their reduced-order model allows eliminating the backward transient analysis that is typically required for the sensitivity analysis of transient problems.

3 Topology optimization problem

This section presents theoretical aspects for topology optimization of linear elastic structures subjected to dynamic loading. In Section 3.1, we describe the problem setting in which the topology is defined by a continuous density field, ρ. In Section 3.2, we discretize the design domain and density field to formulate an optimization problem that can be implemented numerically in PolyDyna. Most of the notation used in this section follows that by Talischi et al. (2012b), and thus, interested readers are referred to that reference for additional information that we may have omitted here for the sake of brevity.

3.1 Continuum setting

The primary goal of topology optimization is to find the shape of a structure that optimizes a given objective function defining some performance measure, while satisfying some constraints. In the context of density-based topology optimization, the shape of the structure is defined by a density field, $\rho \in \mathcal{A}$, in which $\rho = 0$ indicates the absence of material (i.e., void) and $\rho = 1$ indicates the presence of material (i.e., solid). In general, both the objective function, $f(\rho, \mathbf{u})$, and the constraints, $g_j(\rho, \mathbf{u})$, $j = 1, \ldots, K$, depend on the density field and on the solution, $\mathbf{u}$, of a given boundary value problem that depends on the physics of the problem of interest. A topology optimization problem of the form described above can be formally written in the following form:

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

where

$$\mathcal{A} = \left\{ \mathcal{P}(\eta) : \eta \in L^{\infty}(\Omega; [0, 1]) \right\} \tag{2}$$

defines the space of admissible density fields, in which η is the design function and

$$\mathcal{P}(\eta) = (\mathcal{P}_F \circ \mathcal{P}_s)(\eta) \tag{3}$$

is a regularization map used to ensure the well-posedness of the topology optimization problem and defined in terms of a smooth regularization filter, $\mathcal{P}_F$, and an operator, $\mathcal{P}_s$, used to impose additional constraints such as symmetries or pattern repetition to the set of admissible density fields (Talischi et al. 2012b; Giraldo-Londoño and Paulino 2020b). The smooth regularization filter, $\mathcal{P}_F$, is defined by a convolution of the design function and a kernel, F, so that ρ inherits the smoothness characteristics of F. The smooth regularization filter is expressed as

$$\mathcal{P}_F(\eta)(\mathbf{x}) = \int_{\Omega} F(\mathbf{x}, \bar{\mathbf{x}})\, \eta(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}, \tag{4}$$

where a popular choice for F is the nonlinear kernel of radius R,

$$F(\mathbf{x}, \bar{\mathbf{x}}) = c(\mathbf{x}) \max\left(1 - \frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{R},\ 0\right)^q, \tag{5}$$

q is a nonlinear exponent, and $c(\mathbf{x})$ is a normalization factor used to ensure that $\int_{\Omega} F(\mathbf{x}, \bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}} = 1$.

Given that we are interested in developing a formulation tailored to solve elastodynamics problems, the state variables are found such that, for a given initial displacement field, $\mathbf{u}_0(\mathbf{x})$, initial velocity field, $\mathbf{v}_0(\mathbf{x})$, prescribed displacements, $\bar{\mathbf{u}}$ (applied to the portion, Γ_D, of $\partial\Omega$), and prescribed boundary tractions, $\mathbf{t}$ (applied to the portion, Γ_N, of $\partial\Omega$), $\mathbf{u} \in \mathcal{V}$ satisfies the following boundary value problem:

$$\begin{aligned}
&\operatorname{div}\boldsymbol{\sigma} + \mathbf{b} = \bar{\rho}\ddot{\mathbf{u}} + \bar{c}\dot{\mathbf{u}}, && \text{in } \Omega \times (0, t_f] \\
&\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}, t), && \text{on } \Gamma_D \times (0, t_f] \\
&\boldsymbol{\sigma}(\mathbf{x}, t) \cdot \mathbf{n} = \mathbf{t}(\mathbf{x}, t), && \text{on } \Gamma_N \times (0, t_f] \\
&\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), && \text{in } \Omega \\
&\dot{\mathbf{u}}(\mathbf{x}, 0) = \mathbf{v}_0(\mathbf{x}), && \text{in } \Omega,
\end{aligned} \tag{6}$$

where t is the temporal variable; t_f is the duration of the dynamic event;

$$\mathcal{V} = \left\{ \mathbf{u} \in H^1(\Omega \times (0, t_f], \mathbb{R}^2 \times \mathbb{R}) \colon \mathbf{u}|_{\Gamma_D} = \bar{\mathbf{u}}(\mathbf{x}, t) \right\} \tag{7}$$

is the space of admissible displacements; $\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}$ is the stress tensor, defined in terms of the linear isotropic elasticity tensor, $\mathbb{C}$, and the infinitesimal strain tensor, $\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)$; $\mathbf{b}$ is the body force field; and $\mathbf{n}$ is the unit outward normal vector to Γ_N. The elasticity tensor $\mathbb{C}$ depends on the density field, and it can be computed using either SIMP (solid isotropic material with penalization) (Bendsøe 1989; Zhou and Rozvany 1991; Rozvany et al. 1992) or RAMP (rational approximation of material properties) (Stolpe and Svanberg 2001; Bendsøe and Sigmund 2003). Moreover, $\bar{\rho}$ refers to the physical density at a point $\mathbf{x} \in \Omega$, which is typically defined in terms of the density field, ρ, as $\bar{\rho} = m_V(\rho)\rho_0$, in which ρ_0 is the mass density of the solid material and $m_V(\rho)$ is a volume interpolation function relating the volume fraction at a point $\mathbf{x} \in \Omega$ with its density at that point. Finally, $\bar{c}$ is a damping factor, which leads to energy dissipation in the system. The design domain and boundary conditions, among other variables of interest for the dynamic topology optimization problem, are depicted in Fig. 1.

The continuum topology optimization statement (1) is valid for arbitrary objective and constraint functions. For instance, a typical objective function for structural dynamics applications is the mean compliance of the system,

$$f(\rho, \mathbf{u}) = \frac{1}{t_f}\int_0^{t_f}\int_{\Gamma_N} \mathbf{t} \cdot \mathbf{u} \mathrm{d}\mathbf{x}\mathrm{d}t, \tag{8}$$

or the mean strain energy the system,

$$f(\rho, \mathbf{u}) = \frac{1}{2t_f}\int_0^{t_f}\int_{\Omega} \boldsymbol{\sigma} : \boldsymbol{\varepsilon} \mathrm{d}\mathbf{x}\mathrm{d}t. \tag{9}$$

We will obtain solutions for these two objective functions as well as for additional objective functions of interest, as shown later in the numerical results. To limit the amount of material that can be used for a design, one could use a constraint of the form[1]

$$g(\rho) = \frac{1}{|\Omega|}\int_{\Omega} m_V(\rho)\mathrm{d}\mathbf{x} - \bar{v} \leq 0, \tag{10}$$

where $\bar{v}$ is the volume fraction limit.

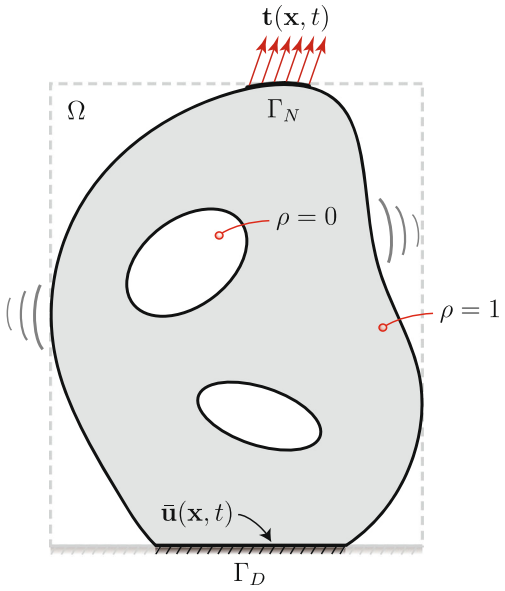


Fig. 1 Design domain and boundary conditions for the dynamic topology optimization problem

3.2 Discrete setting

To solve a topology optimization problem such as (1), we need to discretize both the design space and the displacement field.[2] Our first step toward that goal is to discretize the design domain, Ω, using a fixed partition, $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$, composed of N elements, in which $\Omega_k \cap \Omega_\ell = \emptyset \ \forall k \neq \ell$, and $\cup_{\ell=1}^N \overline{\Omega}_\ell = \overline{\Omega}$. We use $\mathcal{T}_h$ to define a piecewise constant discretization of the design space, $\mathcal{A}$, and to discretize the space of admissible displacements, $\mathcal{V}$. The piecewise constant discretization of $\mathcal{A}$ is given by

$$\mathcal{A}_h = \left\{ \mathcal{P}(\eta_h) : 0 \leq \eta_h \leq 1, \ \eta|_{\Omega_\ell} = \text{const} \ \forall \ell \right\}, \tag{11}$$

where

$$\eta_h = \sum_{\ell=1}^{N} z_\ell \chi_{\Omega_\ell}(\mathbf{x}) \tag{12}$$

is the discretized design function defined in terms of the value of a characteristic function, χ, evaluated on element

[1] This work uses a variation of this constraint, but imposed on subregions of the design domain. Details are provided in the next section.

[2] To be consistent with previous work, here we adopt the notation by Talischi et al. (2012b).

Ω_ℓ[3] and in terms of the vector of design variables $\mathbf{z} = \{z_\ell\}_{\ell=1}^N$, in which $\mathbf{z} \in [0,1]^N$.

The discretized density field that we use in PolyDyna is of the form

$$\tilde{\rho}_h(\mathbf{x}) = \sum_{\ell=1}^{N} y_\ell \chi_{\Omega_\ell}(\mathbf{x}), \tag{13}$$

where $y_\ell = \rho_h\left(\mathbf{x}_\ell^*\right)$ corresponds to the value of ρ_h at the centroid, $\mathbf{x}_\ell^*$, of element ℓ. If we neglect additional constraints such as pattern repetition or symmetries (i.e., if we neglect $\mathcal{P}_s$ in (3)) in the space of admissible density fields, the elemental values, y_ℓ, are given by[4]

$$\begin{aligned} y_\ell &= \rho_h\left(\mathbf{x}_\ell^*\right) \\ &= \mathcal{P}_F(\eta_h)\left(\mathbf{x}_\ell^*\right) \\ &= \int_\Omega F\left(\mathbf{x}_\ell^*, \bar{\mathbf{x}}\right) \eta_h(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}} \\ &= \sum_{k=1}^{N} z_k \int_{\Omega_k} F\left(\mathbf{x}_\ell^*, \bar{\mathbf{x}}\right) \mathrm{d}\bar{\mathbf{x}}, \end{aligned} \tag{14}$$

which can be written in a convenient way as

$$\mathbf{y} = \mathbf{P}\mathbf{z}, \tag{15}$$

where

$$P_{\ell k} = \int_{\Omega_k} F\left(\mathbf{x}_\ell^*, \bar{\mathbf{x}}\right) \mathrm{d}\bar{\mathbf{x}} = \frac{w_{\ell k} A_k}{\sum_{j=1}^{N} w_{\ell j} A_j}, \tag{16}$$

$A_k = |\Omega_k|$ is the area of element k, and $w_{k\ell}$ comes from the definition of F in (5):

$$w_{\ell k} = \max\left(0, 1 - \frac{\|\mathbf{x}_\ell - \mathbf{x}_k\|_2}{R}\right)^q. \tag{17}$$

The vector $\mathbf{y} = \{y_\ell\}_{\ell=1}^N$ is the vector of filtered densities given by $\mathbf{y} = \mathbf{P}\mathbf{z}$, where $\mathbf{P}$ is the filter matrix (16).

We use the same partition, $\mathcal{T}_h$, to discretize the displacement field, and define a piecewise partition of the time domain, $\mathcal{S}_h = \{t_i\}_{i=0}^{N_t}$, in which t_i is the ith time step and N_t is the number of time steps. Following a standard FE procedure, the boundary value problem for elastodynamics (6) takes the form,[5]

$$\mathbf{M}\ddot{\mathbf{u}}_i + \mathbf{C}\dot{\mathbf{u}}_i + \mathbf{K}\mathbf{u}_i = \mathbf{f}_i, \quad i = 0, \ldots, N_t, \tag{18}$$

where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$, are the mass, damping, and stiffness matrices, respectively, $\mathbf{f}_i$ is the force vector at the ith time step, and $\ddot{\mathbf{u}}_i$, $\dot{\mathbf{u}}_i$, and $\mathbf{u}_i$ are, respectively, the acceleration, velocity, and displacement vectors at time step i. We evaluate the mass and stiffness matrices as

$$\mathbf{M} = \sum_{\ell=1}^{N} \widetilde{m}_V(y_\ell)\mathbf{m}_\ell \quad \text{and} \quad \mathbf{K} = \sum_{\ell=1}^{N} \widetilde{m}_E(y_\ell)\mathbf{k}_\ell, \tag{19}$$

where $\sum_{\ell=1}^N$ refers to the FE assembly operator;

$$\mathbf{m}_\ell = \int_{\Omega_\ell} \rho_0 \mathbf{N}_\ell^T \mathbf{N}_\ell \mathrm{d}\mathbf{x} \quad \text{and} \quad \mathbf{k}_\ell = \int_{\Omega_\ell} \mathbf{B}_\ell^T \mathbf{D}_0 \mathbf{B}_\ell \mathrm{d}\mathbf{x} \tag{20}$$

are the ℓth element mass and stiffness matrices, respectively; $\mathbf{N}_\ell$ and $\mathbf{B}_\ell$ are the respective shape functions and the strain-displacement matrix; and $\mathbf{D}_0$ is the material moduli matrix for a linear isotropic material. The volume interpolation function,

$$\widetilde{m}_V(y_\ell) = \epsilon + (1-\epsilon) m_V(y_\ell),$$

defines the volume fraction of each element as a function of its density, and the material interpolation function,

$$\widetilde{m}_E(y_\ell) = \epsilon + (1-\epsilon) m_E(y_\ell),$$

defines the stiffness of each element as a function of its density. The volume interpolation function is defined using the threshold projection function (Wang et al. 2011),[6]

$$m_V(y_\ell) = \frac{\tanh(\bar{\beta}\bar{\eta}) + \tanh\left(\bar{\beta}\left(y_\ell - \bar{\eta}\right)\right)}{\tanh(\bar{\beta}\bar{\eta}) + \tanh\left(\bar{\beta}\left(1 - \bar{\eta}\right)\right)}, \tag{21}$$

where $\bar{\eta}$ is the threshold density and $\bar{\beta}$ controls the aggressiveness of the projection, and the stiffness interpolation function is based on the RAMP function (Stolpe and Svanberg 2001; Bendsøe and Sigmund 2003),

$$m_E(y_\ell) = \frac{m_V(y_\ell)}{1 + p_0[1 - m_V(y_\ell)]}, \tag{22}$$

where p_0 is the RAMP penalization parameter. Both $\tilde{m}_V$ and $\tilde{m}_E$ use an Ersatz parameter, $\epsilon \ll 1$ to prevent numerical instabilities when $y_\ell \to 0$.[7] Finally, we compute the damping matrix using proportional damping, such that

$$\mathbf{C} = \alpha_r \mathbf{M} + \beta_r \mathbf{K}, \tag{23}$$

in which α_r and β_r are the Rayleigh damping parameters.

Based on the discretization of both the density and displacement fields, we arrive at our discretized elastodynamic topology optimization problem in which we seek to minimize an objective function, $f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t})$, while limiting the available amount of material. The topology is defined by the design variables, $\mathbf{z} \in [0,1]^N$, and the state variables, $\mathbf{u}_i, i = 0, \ldots, N_t$, represent the time-dependent

[3] The characteristic function, $\chi_{\Omega_\ell}(\mathbf{x})$, associated with element Ω_ℓ, is equal to 1 if $\mathbf{x} \in \Omega_\ell$ and 0 otherwise (Talischi et al. 2012b).

[4] Details related to the filter operation when considering symmetry are given by Giraldo-Londoño and Paulino (2020b).

[5] The form of (18) is valid when $\bar{\mathbf{u}}(\mathbf{x}, t) = \mathbf{0}$ (see Fig. 1). When ground accelerations are considered, $\mathbf{f}_i$ should be replaced by $-a_g(t_i)\mathbf{1}$, in which $a_g(t_i)$ is the ground acceleration at $t = t_i$.

[6] Although we use the threshold projection function to solve the examples in the present manuscript, PolyDyna has other volume interpolation functions built in, e.g., the Heaviside projection function by Guest (2009), as implemented in PolyTop (Talischi et al. 2012b).

[7] For dynamic topology optimization problems, we recommend the RAMP function because SIMP may lead to instabilities when the element densities approach zero.

physical response (displacement field) of the system, which we obtain from the finite element representation of the boundary value problem (6). Mathematically, we state this topology optimization problem as follows:

$$\begin{aligned}
&\min_{\mathbf{z}\in[0,1]^N} \; f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t}) \\
&\text{s.t. } g_j(\mathbf{z}) = \frac{\sum\limits_{\ell\in\mathcal{E}_j} A_\ell \widetilde{m}_V(y_\ell)}{\sum\limits_{\ell\in\mathcal{E}_j} A_\ell} - \bar{v}_j \leq 0, \quad j = 1, \ldots, N_c \\
&\text{with: } \mathbf{M}\ddot{\mathbf{u}}_i + \mathbf{C}\dot{\mathbf{u}}_i + \mathbf{K}\mathbf{u}_i = \mathbf{f}_i, \;\; i = 0, \ldots, N_t,
\end{aligned} \tag{24}$$

where A_ℓ is the area of element ℓ, N_c is the number of volume constraints, $\mathcal{E}_j$ are the element indices associated with constraint g_j, and $\bar{v}_j$ is the volume fraction limit associated with constraint g_j. The volume constraint definition used above allows imposing constraints to subregions of the design domain, which we can use to indirectly control the length scale of the final designs. This general volume constraint definition was introduced by Zhang et al. (2018) in the context of multi-material ground-structure optimization and extended by Sanders et al. (2018) for multi-material density-based topology optimization. The ability to control length scale using regional volume constraints was demonstrated by Sanders et al. (2018) and later adopted by Giraldo-Londoño et al. (2020) for the design of thermomechanical structures.

The optimization statement (24) is written such that it can be used with any objective function of the form $f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t})$. Different objective functions can be used depending on the desired optimization goal. For instance,

$$f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t}) = \frac{1}{N_t} \sum_{i=0}^{N_t} \mathbf{f}_i^T \mathbf{u}_i \tag{25}$$

if the goal is to minimize the mean dynamic compliance,

$$f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t}) = \frac{1}{2N_t} \sum_{i=0}^{N_t} \mathbf{u}_i^T \mathbf{K}\mathbf{u}_i \tag{26}$$

if the goal is to minimize the mean strain energy, and

$$f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t}) = \frac{1}{N_t} \sum_{i=0}^{N_t} \left(\mathbf{L}^T \mathbf{u}_i\right)^2 \tag{27}$$

if the goal is to minimize the squared displacement at a target degree of freedom. We have implemented all these objective functions in `PolyDyna` and present numerical results discussing the differences in the results achieved using each of these objective functions.

4 HHT-α method

The HHT-α method (Hilber et al. 1977) is a generalization of the Newmark-β (Newmark 1959) used to solve structural dynamics problems. The HHT-α method modifies the equation of motion $(24)_3$ using a parameter α representing a numerical lag between the damping, stiffness, and external force vector, as follows:

$$\begin{aligned}
&\mathbf{M}\ddot{\mathbf{u}}_i + (1-\alpha)\mathbf{C}\dot{\mathbf{u}}_i + \alpha\mathbf{C}\dot{\mathbf{u}}_{i-1} + (1-\alpha)\mathbf{K}\mathbf{u}_i + \alpha\mathbf{K}\mathbf{u}_{i-1} \\
&= (1-\alpha)\mathbf{f}_i + \alpha\mathbf{f}_{i-1}, \;\; i = 1, \ldots, N_t.
\end{aligned} \tag{28}$$

The HHT-α method is used together with the Newmark-β finite difference (FD) relationships,

$$\begin{aligned}
\mathbf{u}_i &= \mathbf{u}_{i-1} + \Delta t\dot{\mathbf{u}}_{i-1} + \Delta t^2 \left[\left(\frac{1}{2} - \beta\right)\ddot{\mathbf{u}}_{i-1} + \beta\ddot{\mathbf{u}}_i\right], \\
\dot{\mathbf{u}}_i &= \dot{\mathbf{u}}_{i-1} + \Delta t\left[(1-\gamma)\ddot{\mathbf{u}}_{i-1} + \gamma\ddot{\mathbf{u}}_i\right],
\end{aligned} \tag{29}$$

so that it reduces to the Newmark-β when $\alpha = 0$. To ensure that the HHT-α method is at least second-order accurate and unconditionally stable, α, β, and γ must satisfy the following conditions (Hilber et al. 1977):

$$\begin{aligned}
&0 \leq \alpha \leq 1/3, \\
&\beta = (1+\alpha)^2/4, \;\; \text{and} \\
&\gamma = (1+2\alpha)/2.
\end{aligned} \tag{30}$$

Substituting the Newmark-β FD relationships (29) into (28), we obtain the discretized equation of motion in residual form,

$$\begin{aligned}
\mathbf{R}_i &= \mathbf{M}_1\ddot{\mathbf{u}}_i + \mathbf{M}_0\ddot{\mathbf{u}}_{i-1} + \mathbf{C}_0\dot{\mathbf{u}}_{i-1} + \mathbf{K}\mathbf{u}_{i-1} - (1-\alpha)\mathbf{f}_i \\
&\quad -\alpha\mathbf{f}_{i-1} = \mathbf{0},
\end{aligned} \tag{31}$$

where

$$\begin{aligned}
&\mathbf{M}_1 = \mathbf{M} + (1-\alpha)\gamma\Delta t\mathbf{C} + (1-\alpha)\beta\Delta t^2\mathbf{K}, \\
&\mathbf{M}_0 = (1-\alpha)(1-\gamma)\Delta t\mathbf{C} + (1-\alpha)\left(\frac{1}{2} - \beta\right)\Delta t^2\mathbf{K}, \text{ and} \\
&\mathbf{C}_0 = \mathbf{C} + (1-\alpha)\Delta t\mathbf{K}.
\end{aligned} \tag{32}$$

To solve the dynamic problem for each time step $i = 1, \ldots, N_t$, we solve (31) for $\ddot{\mathbf{u}}_i$ and then update $\mathbf{u}_i$ and $\dot{\mathbf{u}}_i$ using the Newmark-β FD relationships (29). For time step $i = 0$, we use $\mathbf{u}_0$ and $\dot{\mathbf{u}}_0$ from the initial conditions and compute $\ddot{\mathbf{u}}_0$ as $\ddot{\mathbf{u}}_0 = \mathbf{M}^{-1}(\mathbf{f}_0 - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{K}\mathbf{u}_0)$.

5 Sensitivity analysis

In pursuit of a computationally efficient implementation, we adopt the adjoint method for sensitivity analysis in which we avoid computing expensive derivatives of the state variables. Two main philosophies dominate adjoint sensitivity analysis for elastodynamics problems. The first

is the *differentiate-then-discretize* approach in which the adjoint problem is constructed in terms of a discretized state variable and continuous time variable and the solution is obtained by subsequently discretizing in time. The second is the *discretize-then-differentiate* approach in which the adjoint problem is constructed in terms of state and time variables that have both been discretized a priori. Although the first approach is simpler (it is not tied to a specific numerical integration scheme), Jensen et al. (2014) demonstrated that the differentiate-then-discretize approach leads to consistency errors, i.e., differences between the computed and exact sensitivities. Thus, we adopt the latter *discretize-then-differentiate* approach and note that by choosing the HHT-α method, which is a generalization of the Newmark-β method, the sensitivities derived here are valid for other classical numerical integration schemes (e.g., trapezoidal, explicit central differences, and average constant acceleration).

The sensitivity of a generic objective function, $f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t})$, with respect to a design variable, z_e, is expressed as

$$\frac{\mathrm{d}f}{\mathrm{d}z_e} = \frac{\partial f}{\partial z_e} + \sum_{i=0}^{N_t} \frac{\partial f}{\partial \mathbf{u}_i} \cdot \frac{\partial \mathbf{u}_i}{\partial z_e}. \tag{33}$$

To avoid the expensive computation of $\partial \mathbf{u}_i / \partial z_e$, we use the adjoint method, for which we use the residual form of the HHT-α method from (31) together with the initial condition

$$\mathbf{R}_0 = \mathbf{M}\ddot{\mathbf{u}}_0 + \mathbf{C}\dot{\mathbf{u}}_0 + \mathbf{K}\mathbf{u}_0 - \mathbf{f}_0 = \mathbf{0} \tag{34}$$

and the Newmark-β FD relationships (29). To facilitate the derivations, we rewrite (29) in residual form as follows:

$$\begin{aligned}
\mathbf{P}_i &= -\mathbf{u}_i + \mathbf{u}_{i-1} + \Delta t \dot{\mathbf{u}}_{i-1} \\
&\quad + \Delta t^2 \left[\left(\frac{1}{2} - \beta \right) \ddot{\mathbf{u}}_{i-1} + \beta \ddot{\mathbf{u}}_i \right] = \mathbf{0}, \quad i = 1, \ldots, N_t \\
\mathbf{Q}_i &= -\dot{\mathbf{u}}_i + \dot{\mathbf{u}}_{i-1} + \Delta t \left[(1-\gamma)\ddot{\mathbf{u}}_{i-1} + \gamma \ddot{\mathbf{u}}_i \right] \\
&= \mathbf{0}, \quad i = 1, \ldots, N_t.
\end{aligned} \tag{35}$$

Next, we introduce adjoint variables $\boldsymbol{\xi}_i$, $\boldsymbol{\mu}_i$, and $\boldsymbol{\nu}_i$, $i = 0, \ldots, N_t$ and rewrite (33) as

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}z_e} &= \frac{\partial f}{\partial z_e} + \sum_{i=0}^{N_t} \frac{\partial f}{\partial \mathbf{u}_i} \cdot \frac{\partial \mathbf{u}_i}{\partial z_e} + \sum_{i=0}^{N_t} \boldsymbol{\xi}_i^T \left[\frac{\partial \mathbf{R}_i}{\partial z_e} \right. \\
&\quad \left. + \sum_{\ell=0}^{N_t} \left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{u}_\ell} \cdot \frac{\partial \mathbf{u}_\ell}{\partial z_e} + \frac{\partial \mathbf{R}_i}{\partial \dot{\mathbf{u}}_\ell} \cdot \frac{\partial \dot{\mathbf{u}}_\ell}{\partial z_e} + \frac{\partial \mathbf{R}_i}{\partial \ddot{\mathbf{u}}_\ell} \cdot \frac{\partial \ddot{\mathbf{u}}_\ell}{\partial z_e} \right) \right] \\
&\quad + \sum_{i=1}^{N_t} \boldsymbol{\mu}_i^T \left[\frac{\partial \mathbf{P}_i}{\partial z_e} + \sum_{\ell=0}^{N_t} \left(\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_\ell} \cdot \frac{\partial \mathbf{u}_\ell}{\partial z_e} + \frac{\partial \mathbf{P}_i}{\partial \dot{\mathbf{u}}_\ell} \cdot \frac{\partial \dot{\mathbf{u}}_\ell}{\partial z_e} \right. \right. \\
&\quad \left. \left. + \frac{\partial \mathbf{P}_i}{\partial \ddot{\mathbf{u}}_\ell} \cdot \frac{\partial \ddot{\mathbf{u}}_\ell}{\partial z_e} \right) \right] \\
&\quad + \sum_{i=1}^{N_t} \boldsymbol{\nu}_i^T \left[\frac{\partial \mathbf{Q}_i}{\partial z_e} + \sum_{\ell=0}^{N_t} \left(\frac{\partial \mathbf{Q}_i}{\partial \mathbf{u}_\ell} \cdot \frac{\partial \mathbf{u}_\ell}{\partial z_e} + \frac{\partial \mathbf{Q}_i}{\partial \dot{\mathbf{u}}_\ell} \cdot \frac{\partial \dot{\mathbf{u}}_\ell}{\partial z_e} \right. \right. \\
&\quad \left. \left. + \frac{\partial \mathbf{Q}_i}{\partial \ddot{\mathbf{u}}_\ell} \cdot \frac{\partial \ddot{\mathbf{u}}_\ell}{\partial z_e} \right) \right].
\end{aligned} \tag{36}$$

From (35), it is clear that $\frac{\partial \mathbf{P}_i}{\partial z_e} = \mathbf{0}$ and $\frac{\partial \mathbf{Q}_i}{\partial z_e} = \mathbf{0}$. Moreover, we assume that the initial conditions are independent of the design variables, so that $\frac{\partial \mathbf{u}_0}{\partial z_e} = \mathbf{0}$ and $\frac{\partial \dot{\mathbf{u}}_0}{\partial z_e} = \mathbf{0}$. We apply these simplifications and rewrite (36) as

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}z_e} &= \frac{\partial f}{\partial z_e} + \sum_{i=0}^{N_t} \boldsymbol{\xi}_i^T \frac{\partial \mathbf{R}_i}{\partial z_e} + \left(\boldsymbol{\xi}_0^T \frac{\partial \mathbf{R}_0}{\partial \ddot{\mathbf{u}}_0} + \boldsymbol{\xi}_1^T \frac{\partial \mathbf{R}_1}{\partial \ddot{\mathbf{u}}_0} \right. \\
&\quad \left. + \boldsymbol{\mu}_1^T \frac{\partial \mathbf{P}_1}{\partial \ddot{\mathbf{u}}_0} + \boldsymbol{\nu}_1^T \frac{\partial \mathbf{Q}_1}{\partial \ddot{\mathbf{u}}_0} \right) \cdot \frac{\partial \ddot{\mathbf{u}}_0}{\partial z_e} \\
&\quad + \sum_{i=1}^{N_t} \sum_{\ell=1}^{N_t} \left(\boldsymbol{\xi}_\ell^T \frac{\partial \mathbf{R}_\ell}{\partial \mathbf{u}_i} + \boldsymbol{\mu}_\ell^T \frac{\partial \mathbf{P}_\ell}{\partial \mathbf{u}_i} \right. \\
&\quad \left. + \boldsymbol{\nu}_\ell^T \frac{\partial \mathbf{Q}_\ell}{\partial \mathbf{u}_i} + \frac{\partial f}{\partial \mathbf{u}_i} \right) \cdot \frac{\partial \mathbf{u}_i}{\partial z_e} \\
&\quad + \sum_{i=1}^{N_t} \sum_{\ell=1}^{N_t} \left(\boldsymbol{\xi}_\ell^T \frac{\partial \mathbf{R}_\ell}{\partial \dot{\mathbf{u}}_i} + \boldsymbol{\mu}_\ell^T \frac{\partial \mathbf{P}_\ell}{\partial \dot{\mathbf{u}}_i} + \boldsymbol{\nu}_\ell^T \frac{\partial \mathbf{Q}_\ell}{\partial \dot{\mathbf{u}}_i} \right) \cdot \frac{\partial \dot{\mathbf{u}}_i}{\partial z_e} \\
&\quad + \sum_{i=1}^{N_t} \sum_{\ell=1}^{N_t} \left(\boldsymbol{\xi}_\ell^T \frac{\partial \mathbf{R}_\ell}{\partial \ddot{\mathbf{u}}_i} + \boldsymbol{\mu}_\ell^T \frac{\partial \mathbf{P}_\ell}{\partial \ddot{\mathbf{u}}_i} + \boldsymbol{\nu}_\ell^T \frac{\partial \mathbf{Q}_\ell}{\partial \ddot{\mathbf{u}}_i} \right) \cdot \frac{\partial \ddot{\mathbf{u}}_i}{\partial z_e},
\end{aligned} \tag{37}$$

from which we obtain the sensitivity of the objective as

$$\frac{\mathrm{d}f}{\mathrm{d}z_e} = \frac{\partial f}{\partial z_e} + \sum_{i=0}^{N_t} \boldsymbol{\xi}_i^T \frac{\partial \mathbf{R}_i}{\partial z_e}, \tag{38}$$

and define the adjoint problem, such that

$$\boldsymbol{\xi}_0^T \frac{\partial \mathbf{R}_0}{\partial \ddot{\mathbf{u}}_0} + \boldsymbol{\xi}_1^T \frac{\partial \mathbf{R}_1}{\partial \ddot{\mathbf{u}}_0} + \boldsymbol{\mu}_1^T \frac{\partial \mathbf{P}_1}{\partial \ddot{\mathbf{u}}_0} + \boldsymbol{\nu}_1^T \frac{\partial \mathbf{Q}_1}{\partial \ddot{\mathbf{u}}_0} = \mathbf{0} \tag{39}$$

for $i = 0$ and

$$\begin{aligned}
&\sum_{\ell=1}^{N_t} \left(\boldsymbol{\xi}_\ell^T \frac{\partial \mathbf{R}_\ell}{\partial \mathbf{u}_i} + \boldsymbol{\mu}_\ell^T \frac{\partial \mathbf{P}_\ell}{\partial \mathbf{u}_i} + \boldsymbol{\nu}_\ell^T \frac{\partial \mathbf{Q}_\ell}{\partial \mathbf{u}_i} + \frac{\partial f}{\partial \mathbf{u}_i} \right) = \mathbf{0}, \\
&\sum_{\ell=1}^{N_t} \left(\boldsymbol{\xi}_\ell^T \frac{\partial \mathbf{R}_\ell}{\partial \dot{\mathbf{u}}_i} + \boldsymbol{\mu}_\ell^T \frac{\partial \mathbf{P}_\ell}{\partial \dot{\mathbf{u}}_i} + \boldsymbol{\nu}_\ell^T \frac{\partial \mathbf{Q}_\ell}{\partial \dot{\mathbf{u}}_i} \right) = \mathbf{0}, \quad \text{and} \\
&\sum_{\ell=1}^{N_t} \left(\boldsymbol{\xi}_\ell^T \frac{\partial \mathbf{R}_\ell}{\partial \ddot{\mathbf{u}}_i} + \boldsymbol{\mu}_\ell^T \frac{\partial \mathbf{P}_\ell}{\partial \ddot{\mathbf{u}}_i} + \boldsymbol{\nu}_\ell^T \frac{\partial \mathbf{Q}_\ell}{\partial \ddot{\mathbf{u}}_i} \right) = \mathbf{0}
\end{aligned} \tag{40}$$

for $i = 1, \ldots, N_t$. Notice that in (39) and (40), we designed the adjoint system such that all terms containing $\partial \mathbf{u}_i / \partial z_e$, $\partial \dot{\mathbf{u}}_i / \partial z_e$, and $\partial \ddot{\mathbf{u}}_i / \partial z_e$ vanish from (37).

We substitute the HHT-α residual (31), the initial conditions (34), and the residual form of the Newmark-β FD relationships (35) into (39)–(40) to obtain the solution of the adjoint problem as follows:

$$\boldsymbol{\mu}_{N_t} = \frac{\partial f}{\partial \mathbf{u}_{N_t}}, \quad \boldsymbol{\nu}_{N_t} = \mathbf{0}, \quad \mathbf{M}_1 \boldsymbol{\xi}_{N_t} = -\beta \Delta t^2 \boldsymbol{\mu}_{N_t} - \gamma \Delta t \boldsymbol{\nu}_{N_t} \tag{41}$$

for $i = N_t$,

$$\begin{aligned}\boldsymbol{\mu}_{i-1} &= \frac{\partial f}{\partial \mathbf{u}_{i-1}} + \mathbf{K}\boldsymbol{\xi}_i + \boldsymbol{\mu}_i, \quad \boldsymbol{v}_{i-1} = \mathbf{C}_0\boldsymbol{\xi}_i + \Delta t\boldsymbol{\mu}_i + \boldsymbol{v}_i \\ \mathbf{M}_1\boldsymbol{\xi}_{i-1} &= \mathbf{M}_0\boldsymbol{\xi}_i - \Delta t^2\left[\beta\boldsymbol{\mu}_{i-1} + \left(\frac{1}{2}-\beta\right)\boldsymbol{\mu}_i\right] \\ &\quad - \Delta t\left[\gamma\boldsymbol{v}_{i-1} + (1-\gamma)\boldsymbol{v}_i\right] \end{aligned} \tag{42}$$

for $i = 2, \ldots, N_t - 1$, and

$$\mathbf{M}\boldsymbol{\xi}_0 = \mathbf{M}_0\boldsymbol{\xi}_1 - \left(\frac{1}{2}-\beta\right)\Delta t^2\boldsymbol{\mu}_1 - (1-\gamma)\Delta t\boldsymbol{v}_1 \tag{43}$$

for $i = 0$.

Given that the optimization statement (24) uses density-related information through the volume interpolation function, $\widetilde{m}_V(\cdot)$, and the stiffness interpolation function, $\widetilde{m}_E(\cdot)$, it is convenient to recast the sensitivity information in terms of these fields. To this end, we adopt `PolyTop`'s notation and define the vector of element volume fractions as $\mathbf{V} = \widetilde{m}_V(\mathbf{y})$ and the vector of element stiffness parameters as $\mathbf{E} = \widetilde{m}_E(\mathbf{y})$, and express the sensitivity of $f(\mathbf{z}, \mathbf{u}_0, \ldots, \mathbf{u}_{N_t})$ using the chain rule as

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

or in vector form as

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{z}} = \frac{\partial \mathbf{E}}{\partial \mathbf{z}}\frac{\mathrm{d}f}{\mathrm{d}\mathbf{E}} + \frac{\partial \mathbf{V}}{\partial \mathbf{z}}\frac{\mathrm{d}f}{\mathrm{d}\mathbf{V}}, \tag{45}$$

where

$$\frac{\partial \mathbf{E}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_E}(\mathbf{P}\mathbf{z}) \quad \text{and} \quad \frac{\partial \mathbf{V}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_V}(\mathbf{P}\mathbf{z}), \tag{46}$$

$J_{m_E} = \mathrm{diag}\left(\widetilde{m}'_E(y_1), \ldots, \widetilde{m}'_E(y_N)\right)$ and $J_{m_V} = \mathrm{diag}\left(\widetilde{m}'_V(y_1), \ldots, \widetilde{m}'_V(y_N)\right)$ (Talischi et al. 2012b). The sensitivity of f with respect to the element volume fractions and stiffness parameters can be obtained as shown in (38), i.e.,

$$\begin{aligned}\frac{\mathrm{d}f}{\mathrm{d}E_\ell} &= \frac{\partial f}{\partial E_\ell} + \sum_{i=0}^{N_t}\boldsymbol{\xi}_i^T\frac{\partial \mathbf{R}_i}{\partial E_\ell} \quad \text{and} \\ \frac{\mathrm{d}f}{\mathrm{d}V_\ell} &= \frac{\partial f}{\partial V_\ell} + \sum_{i=0}^{N_t}\boldsymbol{\xi}_i^T\frac{\partial \mathbf{R}_i}{\partial V_\ell}. \end{aligned} \tag{47}$$

The partial derivatives, $\partial\mathbf{R}_i/\partial V_\ell$ and $\partial\mathbf{R}_i/\partial E_\ell$, are obtained directly from (34) for $i = 0$ and from (31) for $i = 1, \ldots, N_t$. That is,

$$\begin{aligned}\frac{\partial \mathbf{R}_i}{\partial E_\ell} &= \frac{\partial \mathbf{K}}{\partial E_\ell}(\mathbf{u}_0 + \beta_r\dot{\mathbf{u}}_0) = \mathbf{k}_\ell(\mathbf{u}_{\ell 0} + \beta_r\dot{\mathbf{u}}_{\ell 0}) \quad \text{and} \\ \frac{\partial \mathbf{R}_i}{\partial V_\ell} &= \frac{\partial \mathbf{M}}{\partial V_\ell}(\ddot{\mathbf{u}}_0 + \alpha_r\dot{\mathbf{u}}_0) = \mathbf{m}_\ell(\ddot{\mathbf{u}}_{\ell 0} + \alpha_r\dot{\mathbf{u}}_{\ell 0}) \end{aligned} \tag{48}$$

for $i = 0$, and

$$\begin{aligned}\frac{\partial \mathbf{R}_i}{\partial E_\ell} &= \frac{\partial \mathbf{K}}{\partial E_\ell}\left[(1-\alpha)(\mathbf{u}_i + \beta_r\dot{\mathbf{u}}_i) + \alpha(\mathbf{u}_{i-1} + \beta_r\dot{\mathbf{u}}_{i-1})\right] \\ &= \mathbf{k}_\ell\left[(1-\alpha)(\mathbf{u}_{\ell i} + \beta_r\dot{\mathbf{u}}_{\ell i}) + \alpha(\mathbf{u}_{\ell,i-1} + \beta_r\dot{\mathbf{u}}_{\ell,i-1})\right] \text{ and} \\ \frac{\partial \mathbf{R}_i}{\partial V_\ell} &= \frac{\partial \mathbf{M}}{\partial V_\ell}\left[\ddot{\mathbf{u}}_i + \alpha_r\left((1-\alpha)\dot{\mathbf{u}}_i + \alpha\dot{\mathbf{u}}_{i-1}\right)\right] \\ &= \mathbf{m}_\ell\left[\ddot{\mathbf{u}}_{\ell i} + \alpha_r\left((1-\alpha)\dot{\mathbf{u}}_{\ell i} + \alpha\dot{\mathbf{u}}_{\ell,i-1}\right)\right] \end{aligned} \tag{49}$$

for $i = 1, \ldots, N_t$. Subscripts (ℓi) or $(\ell, i-1)$ above indicate element-wise quantities (e.g., $\mathbf{u}_{\ell i}$ refers to the displacement vector of element ℓ at time step i). The matrices, $\mathbf{m}_\ell$ and $\mathbf{k}_\ell$, are computed according to (20). To obtain the expressions in (48) and (49), we used the fact that $\mathbf{C} = \alpha_r\mathbf{M} + \beta_r\mathbf{K}$. The last piece of information required to complete the sensitivity analysis relates to the partial derivatives, $\partial f/\partial E_\ell$, $\partial f/\partial V_\ell$, and $\partial f/\partial \mathbf{u}_i$, which depend on the particular choice of objective function. For instance, for mean dynamic compliance, we have

$$\frac{\partial f}{\partial E_\ell} = 0, \quad \frac{\partial f}{\partial V_\ell} = 0, \quad \text{and} \quad \frac{\partial f}{\partial \mathbf{u}_i} = \frac{1}{N_t}\mathbf{f}_i, \tag{50}$$

for mean strain energy we have

$$\frac{\partial f}{\partial E_\ell} = \frac{1}{2N_t}\sum_{i=0}^{N_t}\mathbf{u}_{\ell i}^T\mathbf{k}_\ell\mathbf{u}_{\ell i}, \quad \frac{\partial f}{\partial V_\ell} = 0, \quad \text{and} \quad \frac{\partial f}{\partial \mathbf{u}_i} = \frac{1}{N_t}\mathbf{K}\mathbf{u}_i, \tag{51}$$

and for mean squared displacement at a target DOF we have

$$\frac{\partial f}{\partial E_\ell} = 0, \quad \frac{\partial f}{\partial V_\ell} = 0, \quad \text{and} \quad \frac{\partial f}{\partial \mathbf{u}_i} = \frac{2}{N_t}\left(\mathbf{L}^T\mathbf{u}_i\right)\mathbf{L}. \tag{52}$$

Lastly, whenever ground accelerations are considered, the external force vector becomes design-dependent and must be replaced by $\mathbf{f}_i = -\mathbf{M}\mathbf{1}a_g(t_i)$, in which $a_g(t_i)$ is the ground acceleration at time step, t_i, and $\mathbf{1}$ is a column vector of unit entries. As a result, $\partial f/\partial V_\ell$ in (50)–(52) is no longer 0 and must be replaced by

$$\frac{\partial f}{\partial V_\ell} = \sum_{\ell=1}^{N}\sum_{i=0}^{N_t}\mathbf{g}_{\ell i}^T\mathbf{u}_{\ell i}, \quad \text{with} \quad \mathbf{g}_{\ell i} = \mathbf{m}_\ell\mathbf{1}a_g(t_i).$$

6 ZPR-based design variable update using sensitivity separation

Here, we present the ZPR design variable update scheme (Zhang et al. 2018), modified using the sensitivity separation concept (Jiang et al. 2021). First, we discuss the convex approximation of the objective function defined using the concept of sensitivity separation and then present the derivations of the ZPR-based design variable update scheme used in the implementation of `PolyDyna`.

6.1 Convex approximation based on sensitivity separation

We approximate the objective function, $f(\mathbf{z})$,[8] at optimization step k using the convex approximation,

$$\tilde{f}(\mathbf{z}) = f(\mathbf{z}_k) + \mathbf{a}^T \left(\mathbf{z}^{-\hat{\alpha}} - \mathbf{z}_k^{-\hat{\alpha}}\right) + \mathbf{b}^T(\mathbf{z} - \mathbf{z}_k), \tag{53}$$

where $\hat{\alpha} > 0$, and $\mathbf{a} = \{a_e\}_{e=1}^N \geq \mathbf{0}$ and $\mathbf{b} = \{b_e\}_{e=1}^N \geq \mathbf{0}$ are chosen such that $\partial \tilde{f}/\partial z_e = \partial f/\partial z_e$ at $\mathbf{z} = \mathbf{z}_k$. The first-order derivative of the convex approximation (53) at $\mathbf{z} = \mathbf{z}_k$ is

$$\frac{\partial \tilde{f}}{\partial z_e}(\mathbf{z}_k) = -\hat{\alpha} a_e \left(z_e^k\right)^{-\hat{\alpha}-1} + b_e. \tag{54}$$

We divide the the sensitivity of the objective function, $\partial f/\partial z_e$, into a positive part, $\partial f^+/\partial z_e$, and a negative part, $\partial f^-/\partial z_e$:

$$\frac{\partial f}{\partial z_e} = \frac{\partial f^+}{\partial z_e} + \frac{\partial f^-}{\partial z_e}, \quad \text{with} \quad \frac{\partial f^+}{\partial z_e} \geq 0 \quad \text{and} \quad \frac{\partial f^-}{\partial z_e} \leq 0 \tag{55}$$

and define the coefficients a_e and b_e such that

$$\begin{aligned} a_e &= -\frac{1}{\hat{\alpha}} \left(z_e^k\right)^{1+\hat{\alpha}} \frac{\partial f^-}{\partial z_e}(\mathbf{z}_k) \geq 0 \quad \text{and} \\ b_e &= \frac{\partial f^+}{\partial z_e}(\mathbf{z}_k) \geq 0, \quad e = 1, \ldots, N. \end{aligned} \tag{56}$$

If we substitute the expressions for a_e and b_e from (56) into (54) and use (55), it follows that $\frac{\partial \tilde{f}}{\partial z_e}(\mathbf{z}_k) = \frac{\partial f}{\partial z_e}(\mathbf{z}_k)$ (i.e., the sensitivities of the convex approximation correspond to that of the original objective function at $\mathbf{z} = \mathbf{z}_k$).

We can define $\partial f^+/\partial z_e$ and $\partial f^-/\partial z_e$ in an infinite number of ways. Here, we adopt the approach proposed by Giraldo-Londoño et al. (2020) and Giraldo-Londoño and Paulino (2020a), in which $\partial f^+/\partial z_e$ and $\partial f^-/\partial z_e$ are defined based on second-order sensitivity information. At iteration k, the second-order derivatives of the convex approximation (53) are

$$\frac{\partial^2 \tilde{f}}{\partial z_e^2} = a_e \hat{\alpha}(\hat{\alpha}+1) \left(z_e^k\right)^{-(\hat{\alpha}+2)}. \tag{57}$$

We substitute $(56)_1$ into (57) and solve for $\partial f^-/\partial z_e$:

$$\frac{\partial f^-}{\partial z_e} = -\frac{\partial^2 \tilde{f}}{\partial z_e^2} \frac{z_e^k}{\hat{\alpha}+1} \approx -\frac{h_e^k z_e^k}{\hat{\alpha}+1}, \tag{58}$$

where $h_e^k \approx \partial^2 f/\partial z_e^2$. Because we need to satisfy $\partial f^-/\partial z_e \leq 0$ and $\partial f^+/\partial z_e \geq 0$, (58) cannot be used directly to define $\partial f^-/\partial z_e$. One way to define $\partial f^-/\partial z_e$ and $\partial f^+/\partial z_e$ while satisfying the aforementioned conditions is (Giraldo-Londoño et al. 2020; Giraldo-Londoño and Paulino 2020a)

$$\frac{\partial f^-}{\partial z_e} = \min\left\{-\frac{|h_e^k| z_e^k}{\hat{\alpha}+1}, \frac{\partial f}{\partial z_e}\right\}, \qquad \frac{\partial f^+}{\partial z_e} = \frac{\partial f}{\partial z_e} - \frac{\partial f^-}{\partial z_e}, \tag{59}$$

which clearly satisfies $\partial f^-/\partial z_e \leq 0$ and $\partial f^+/\partial z_e \geq 0$. As discussed by Giraldo-Londoño and Paulino (2020a), we estimate the second-order sensitivity information at iteration k using a diagonal approximation of the Hessian matrix based on the Powell Symmetric Broyden (PSB) quasi-Newton update (Dennis and Schnabel 1996; Marjugi and Leong 2013):

$$\mathbf{h}_k = \mathbf{h}_{k-1} + \frac{\mathbf{c}_k^T \mathbf{s}_k - \mathbf{h}_{k-1}^T \mathbf{s}_k^2}{\left(\mathbf{s}_k^2\right)^T \mathbf{s}_k^2} \mathbf{s}_k^2, \tag{60}$$

where

$$\begin{aligned} \mathbf{h}_{k-1} &= \left[h_1^{k-1}, \ldots, h_N^{k-1}\right]^T \\ \mathbf{c}_k &= \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}_k) - \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}_{k-1}), \quad \text{and} \\ \mathbf{s}_k &= \mathbf{z}_k - \mathbf{z}_{k-1}. \end{aligned} \tag{61}$$

6.2 ZPR-based design variable update

In contrast to the traditional ZPR design variable update scheme (Zhang et al. 2018), we use the convex approximation (53) to define the approximate sub-problem,

$$\begin{aligned} &\min_{\mathbf{z} \in [\underline{\mathbf{z}}_k, \bar{\mathbf{z}}_k]} \quad \tilde{f}(\mathbf{z}) = \mathbf{a}^T \mathbf{z}^{-\hat{\alpha}} + \mathbf{b}^T \mathbf{z} \\ &\text{s.t.} \quad g_j(\mathbf{z}) = g_j(\mathbf{z}_k) + \sum_{\ell \in \mathcal{E}_j} \frac{\partial g_j}{\partial z_\ell}(z_\ell - z_\ell^k), \; j = 1, \ldots, N_c, \end{aligned} \tag{62}$$

where

$$\begin{aligned} \underline{z}_\ell^k &= \max\left(z_{\min}, z_\ell^k - \text{move}\right) \quad \text{and} \\ \bar{z}_\ell^k &= \min\left(z_{\max}, z_{\mathbf{e}}^k + \text{move}\right) \end{aligned} \tag{63}$$

are lower and upper bounds on the design variables at optimization step k. To solve the approximate sub-problem (62), we first obtain its Lagrangian, as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{z}, \lambda_1, \ldots, \lambda_{N_c}) &= \mathbf{a}^T \mathbf{z}^{-\hat{\alpha}} + \mathbf{b}^T \mathbf{z} + \sum_{j=1}^{N_c} \lambda_j g_j(\mathbf{z}) \\ &= \sum_{j=1}^{N_c} \mathcal{L}^{(j)}\left(\mathbf{z}, \lambda_j\right), \end{aligned} \tag{64}$$

where

$$\mathcal{L}^{(j)} = \sum_{\ell \in \mathcal{E}_j} \left[a_\ell z_\ell^{-\hat{\alpha}} + b_\ell z_\ell + \lambda_j \frac{\partial g_j}{\partial z_\ell}\left(z_\ell - z_\ell^k\right)\right] + \lambda_j g_j(\mathbf{z}_k). \tag{65}$$

[8] Without loss of generality, here we have dropped the explicit dependence of f on $\mathbf{u}_0, \ldots, \mathbf{u}_{N_t}$.

An important aspect of the formulation is that the Lagrangian (64) can be written in a separable form, $\mathcal{L}(\mathbf{z}, \lambda_1, \ldots, \lambda_{N_c}) = \sum_{j=1}^{N_c} \mathcal{L}^{(j)}\left(\mathbf{z}, \lambda_j\right)$, which is possible only when each design variable is associated with one constraint. The separability of $\mathcal{L}$ allows us to minimize each $\mathcal{L}^{(j)}$ independently with respect to the design variables. To minimize each $\mathcal{L}^{(j)}$, we write the first-order optimality conditions:

$$\frac{\partial \mathcal{L}^{(j)}}{\partial z_\ell} = -\hat{\alpha} a_\ell z_\ell^{-\hat{\alpha}-1} + b_\ell + \lambda_j \frac{\partial g_j}{\partial z_\ell} = 0, \tag{66}$$

from which we obtain

$$z_\ell^* = B_\ell\left(\lambda_j\right) = z_\ell^k \left[\frac{-\frac{\partial f^-}{\partial z_\ell}(\mathbf{z}_k)}{\frac{\partial f^+}{\partial z_\ell}(\mathbf{z}_k) + \lambda_j \frac{\partial g_j}{\partial z_\ell}(\mathbf{z}_k)} \right]^\eta, \quad \forall \ell \in \mathcal{E}_j, \tag{67}$$

where $\eta = 1/\left(1+\hat{\alpha}\right)$. After imposing the box constraints, $z_\ell \in \left[\underline{z}_\ell^k, \bar{z}_\ell^k\right]$, the final form of the primal-dual relationship of (62) takes the form

$$z_\ell\left(\lambda_j\right) = \begin{cases} \underline{z}_\ell^k, & \text{if } B_\ell\left(\lambda_j\right) \le \underline{z}_\ell^k \\ B_\ell\left(\lambda_j\right), & \text{if } \underline{z}_\ell^k < B_\ell\left(\lambda_j\right) \le \bar{z}_\ell^k \\ \bar{z}_\ell^k, & \text{if } B_\ell\left(\lambda_j\right) \ge \bar{z}_\ell^k \end{cases} \tag{68}$$

The dual-objective function associated with the optimization statement (62) can be written in the following separable form: $\mathcal{D}(\lambda_1, \ldots, \lambda_{N_c}) = \sum_{j=1}^{N_c} \mathcal{D}^{(j)}\left(\lambda_j\right)$, where $\mathcal{D}^{(j)}\left(\lambda_j\right) = \mathcal{L}^{(j)}\left(\mathbf{z}\left(\lambda_j\right), \lambda_j\right)$ and $\mathbf{z}\left(\lambda_j\right)$ are given by (68). This separability allows us to maximize each dual-objective function, $\mathcal{D}^{(j)}\left(\lambda_j\right)$, independently from one another to find the Lagrange multiplier, λ_j, associated with constraint $g_j(\mathbf{z})$. The stationary condition of $\mathcal{D}^{(j)}\left(\lambda_j\right)$ with respect to λ_j is given by

$$\frac{\partial \mathcal{D}^{(j)}\left(\lambda_j\right)}{\partial \lambda_j} = g_j(\mathbf{z}_k) + \sum_{\ell \in \mathcal{E}_j} \frac{\partial g_j}{\partial z_\ell}\left(z_\ell\left(\lambda_j\right) - z_\ell^k\right) = 0, \tag{69}$$

which is a monotonic function with respect to λ_j, and thus can be solved using an interval reducing method such as bisection.

The new update scheme reduces to the ZPR scheme when $\partial f^+/\partial z_\ell = 0$ and $\partial f^-/\partial z_\ell = \partial f/\partial z_\ell$, yet it has the ability to solve non-self-adjoint problems (Giraldo-Londoño et al. 2020; Giraldo-Londoño and Paulino 2020a). To improve the robustness of the approach, we update the design variables at optimization step k using a damping scheme, as follows:

$$\mathbf{z}_k = \zeta \mathbf{z}_k + (1-\zeta)\mathbf{z}_{k-1}, \tag{70}$$

where $\zeta \in (0, 1]$ is a damping factor, and $\mathbf{z}_k$ and $\mathbf{z}_{k-1}$ are the design variables computed at optimization steps k and $k-1$, respectively.

7 Matlab implementation

Here, we discuss several details of the implementation of `PolyDyna` in Matlab. Owing to its modular structure, we implement `PolyDyna` on top of `PolyTop` (Talischi et al. 2012b) and apply appropriate modifications where needed. One of the changes is to the finite element analysis routine, which now uses the HHT-α method to solve the structural dynamics problem. Another change is to the design variable update scheme, which uses the modified ZPR update scheme discussed previously and whose numerical implementation follows closely that from `PolyMat` (Sanders et al. 2018). We also add a new subroutine to solve the adjoint problem and adopt the filtering and material interpolation functions from `PolyStress` (Giraldo-Londoño and Paulino 2020b). The filter function in `PolyStress` can be used to impose symmetry to the space of admissible density fields, which comes in handy to solving some problems in this manuscript. The material interpolation function in `PolyStress` has the threshold projection function (21) built in, which allows us to obtain black-and-white solutions.

The implementation of the main `PolyDyna` code is based on the flowchart shown in Fig. 2. The code begins by reading the input data (i.e., the mesh, and boundary

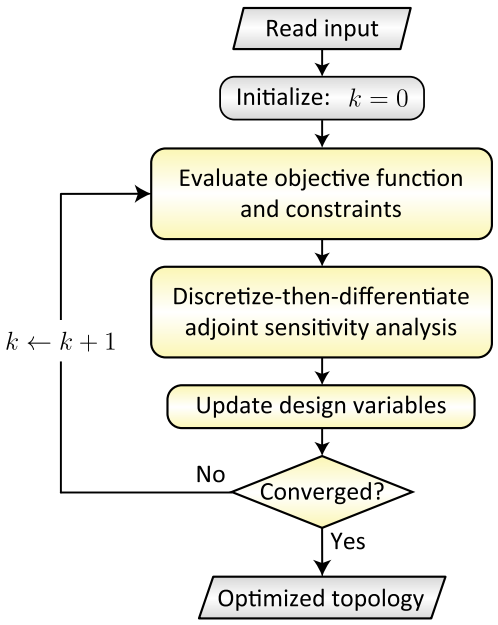


Fig. 2 Schematic flowchart outlining the main steps employed by `PolyDyna` to solve dynamic topology optimization problems

Table 1 Main Matlab files provided with PolyDyna

Function	Description
PolyScript	Main file containing all input data required to run PolyDyna
MatIntFnc	Computes **V**, **E**, based on (21)–(22), and their sensitivities, based on (46)
PolyFilter	Computes the filter matrix, **P**, as given by (16)
PreComputations	Computes the element stiffness matrices, load vector, and other required quantities
FEM_Dyna	Solves the structural dynamics problem using the HHT-α method, as discussed in Section 4
AdjointProblem	Solves the adjoint problem according to (41)–(43)
Centroids	Computes the centroids of all elements (useful to define the volume constraints)
Areas	Computes the areas of all elements (useful to define the volume constraints)

conditions, as well as parameters used for the optimizer) and then initializes the iteration counter, k. It then enters the main loop in which the code evaluates the objective function and constraints given in (24)–(27), followed by the discretize-then-differentiate adjoint sensitivity analysis discussed in Section 5. Once these are evaluated, PolyDyna updates the design variables using the modified ZPR update discussed in Section 6. The design variables are updated until a certain convergence criterion is met. In our case, the convergence criterion is $\|\mathbf{z}_k - \mathbf{z}_{k-1}\|_\infty < \texttt{Tol}$, in which $\|\cdot\|_\infty$ is the infinity norm and Tol is a prescribed tolerance. Once the convergence criterion is met, then PolyDyna outputs the optimized topology.

Table 2 Subroutines called inside the PolyDyna.m file

Subroutine	Description
ObjectiveFnc	Computes the objective function (e.g., that given by (25), (26), or (27)) and its sensitivity with respect to the design variables. This subroutine calls FEM_Dyna to solve the primal problem and AdjointProblem to solve the adjoint probelm
ConstraintFnc	Computes the volume constraints according to the constraint definition shown in the discretized optimization statement (24)
UpdateSchemeSS	Updates the design variables according to the ZPR-based update scheme discussed in Section 6
InitialPlot	Plotting routine used to display the density field during intermediate optimization iterations

7.1 Main functions

As in PolyTop, the implementation of PolyDyna relies on PolyMesher (Talischi et al. 2012a) for the finite element mesh generation. We provide PolyMesher as Electronic Supplementary Material, and it can be downloaded with PolyDyna. We provide additional functions together with PolyDyna, which we describe in Table 1. In addition to the functions described in Table 1, the PolyDyna.m file also contains several subroutines which are called to solve the discretized optimization problem (24). We also provide a summary of these functions in Table 2. Appendix C shows the complete Matlab code for PolyDyna. In the next sections, we describe in more detail several of the subroutines summarized in Tables 1 and 2.

7.2 The PolyScript file

Like in PolyTop, we use a file called PolyScript.m to collect all the input data needed to call PolyDyna. The PolyScript file contains two data structures, fem and opt. The former includes all the information required for the finite element analysis (e.g., FE mesh, boundary conditions, initial conditions, and material properties). The latter contains primarily the information needed by the optimizer (e.g., design variable lower and upper bounds, filter matrix, material interpolation function, volume constraint definition, stopping criteria). The main fields in the fem and opt data structures are summarized in Tables 3 and 4, respectively. Although additional fields are added to the fem structure during the execution of PolyDyna, Table 3 only displays those that are needed in the PolyScript file.

7.3 Dynamic FE analysis routine

This section presents some details of the Matlab implementation of the HHT-α method discussed in Section 4, which we use to solve the structural dynamics problem. The implementation of the HHT-α method, which we provide as a separate file called FEM_Dyna.m, is based on the pseudocode shown in Algorithm 1. The file FEM_Dyna.m can also be found in Appendix D. The FEM_Dyna routine takes as input the fem structure, as well as the vector of volume fractions, **V**, and stiffness parameters, **E**, and provides as outputs the displacement, velocity, and acceleration vectors, as well as the mass, damping, and stiffness matrices, which are used to solve the adjoint problem. The FEM_Dyna routine also populates the fem structure with new fields (e.g., the Cholesky decomposition of matrix $\mathbf{M}_1$ in (32)), which are useful to solve the adjoint problem efficiently.

Table 3 List of fields in `fem` structure

`fem` field	Description
`fem.NNode`	Number of nodes
`fem.NElem`	Number of elements
`fem.Node`	[`NNode` × 2] array of nodes
`fem.Element`	[`NElem` × `Var`] cell array of elements
`fem.Supp`	[`NSupp` × 3] support array
`fem.Load`	[`NLoad` × 2`NStep` + 1] load array
`fem.Mass`	[`NMass` × 2] array of lumped masses
`fem.SElem`	Cell array of passive solid elements
`fem.u0`	[`NDoF` × 1] vector of initial displacements
`fem.v0`	[`NDoF` × 1] vector of initial velocities
`fem.Thickness`	Element thickness
`fem.E0`	Young's modulus of solid material
`fem.Nu0`	Poisson's ratio of solid material
`fem.rho`	Mass density of solid material
`fem.Ar`	[1 × 2] Rayleigh damping parameters, $[\alpha_r, \beta_r]$
`fem.ag`	[`NStep` × 1] ground acceleration (if any)
`fem.Tmax`	Maximum simulation time for dynamic problem
`fem.NStep`	Number of time steps
`fem.alpha`	Parameter α for the HHT-α method
`fem.beta`	Parameter β for the HHT-α method
`fem.gamma`	Parameter γ for the HHT-α method
`fem.Obj`	Objective function (e.g., `'Compliance'`, `'Energy'`, or `'U_DOF'`)
`fem.LL`	[`NDoF` × 1] vector when `fem.Obj = 'U_DOF'` is used

Table 4 List of fields in `opt` structure

`opt` field	Description
`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.P`	Matrix that maps design to element variables
`opt.VolFrac`	Array of specified volume fraction constraints
`opt.NConstr`	Number of volume constraints
`opt.ElemInd`	Cell array of elements associated with each constraint
`opt.Tol`	Convergence tolerance on design variables
`opt.MaxIter`	Maximum number of optimization iterations
`opt.Move`	Allowable move step in the ZPR-based update scheme
`opt.Eta`	Exponent used in the ZPR-based update scheme

Algorithm 1 FEM_Dyna subroutine.

```
1:  function FEM_DYNA(fem, E, V)
2:      Compute K, M, and C using (19) and (23)
3:      if fem.Mass is non-empty then
4:          Add lumped masses to M
5:      end if
6:      if fem.ag is non-empty then
7:          Compute inertial force vector, f_i^g = f_g(t_i) = −M1 a_g(t_i), and set f_i ← f_i + f_i^g
8:      end if
9:      Initialize acceleration vector, ü_0 = M^{-1}(f_0 − C u̇_0 − K_0 u_0)
10:     for i = 1 to fem.NStep do
11:         Compute ü_i using (31)
12:         Update u̇_i and u_i using (29)
13:         if i = 1 then
14:             Store Cholesky decomposition of M_1 in (31) in the fem structure
15:         end if
16:     end for
17:     Return u_i, u̇_i, ü_i (i = 0, ..., fem.NStep), M, C, K, and updated fem structure
18: end function
```

In addition to externally applied dynamic loads, the FEM_Dyna code considers the effects of lumped masses (lines 14–17), which can be attached to specified nodes of the FE mesh. The information related to all lumped masses, if any, is included in the fem.Mass field. The fem.Mass field is $N_L \times 2$, in which the first column contains the node numbers where the lumped masses are located, the second column contains the magnitudes of the masses, and N_L is the number of lumped masses. For instance, if nodes 10 and 12 have each a lumped mass of 5 kg, then the fem.Mass field should read fem.Mass = [10 5;12 5]. PolyDyna also considers the effects of ground acceleration, as seen on lines 18–23 of the FEM_Dyna code. When considering ground acceleration, we simply add the inertia force vector, $-\mathbf{M1}a_g(t_i)$, to the external force vector, $\mathbf{f}_i$, where $\mathbf{M}$ is the mass matrix, $\mathbf{1}$ is a vector of unit entries, and $a_g(t_i)$ is the magnitude of the ground acceleration at time t_i.

To compute the displacement, velocity, and acceleration vectors at time t_i, we solve the residual (31) for $\ddot{\mathbf{u}}_i$ and use this vector to update the displacement and velocity vectors using the Newmark-β finite difference relationships (29). This procedure is repeated for all time steps, as shown on lines 27–39 of the FEM_Dyna code. The implementation of the adjoint problem (41)–(43) needed for the sensitivity analysis is similar to the implementation to solve the primal problem. The main difference is that we use the adjoint loads instead of the physical loads and conduct the loop over all time steps backwards in time. The subroutine AdjointProb.m contains the implementation of the adjoint problem and we provide it in Appendix E.

7.4 Analysis functions

We need to compute both the objective and constraint functions to solve the discretized optimization problem (24). The subroutines ObjectiveFnc and ConstraintFnc compute the objective and constraints, respectively, as well as their sensitivities with respect to z, using the current values of E and V. Both subroutines can be found inside the PolyDyna code, as shown in Appendix C. The ObjectiveFnc subroutine has been written so that it contains several built-in objective functions. Specifically, the subroutine works for the three objective functions discussed in Section 3 (i.e., mean dynamic compliance, mean strain energy, and mean squared displacement at a prescribed DOF). To compute the sensitivity of the objective function with respect to z, ObjectiveFnc also calls the AdjointProblem subroutine, which is provided in Appendix E. The implementation of the ConstraintFnc subroutine bears resemblance to that in PolyMat (Sanders et al. 2018), yet the routine used here is designed for a single material.

7.5 Constraint specification

A distinctive aspect of the formulation is that we can impose an arbitrary number of volume constraints at different sub-regions of the design domain and update the design variables for one constraint independently from the other constraints. This attribute is due to the separability of the Lagrangian function, as we discussed in Section 6. The volume constraint definition in PolyDyna requires the vector,

opt.VolFrac, and the cell array opt.ElemInd. Each entry of vector opt.VolFrac contains the volume fraction limit for constraint j and each entry of opt.ElemInd contains the element indices associated with constraint j. In terms of the optimization statement (24), each entry of opt.VolFrac corresponds to $\bar{v}_j$ and each cell entry of opt.ElemInd corresponds to $\mathcal{E}_j$. To specify the element indices in each entry of opt.ElemInd, one could use a separate file, which identifies the elements for a given volume constraint, as explained in detail by Sanders et al. (2018). However, for the problems solved here, we identify the element indices directly in the PolyScript file by first computing the centroids of all elements in the FE mesh and determining whether or not they belong to a specified sub-region of the design domain. In the last example of the numerical results, we show a sample code explaining how to define the volume constraints.

7.6 Design variable update

PolyDyna uses the design variable update scheme discussed in Section 6. The design variable update scheme adopted here is based on the ZPR scheme (Zhang et al. 2018), so that it can accommodate an arbitrary number of volume constraints in an efficient manner. Unlike the ZPR scheme, the one adopted here uses a different convex approximation based on the sensitivity separation concept (Jiang et al. 2021; Giraldo-Londoño et al. 2020; Giraldo-Londoño and Paulino 2020a), which allows the ZPR scheme to solve non-self-adjoint problems. The update scheme is one of the subroutines located inside PolyDyna.m file (see Table 2), and it is shown below for completeness:

```
function [zNew,dfdz,c,Change] = UpdateSchemeSS(dfdz,dfdz0,c,g,dgdz,z0,opt)
zMin=opt.zMin; zMax=opt.zMax;
move=opt.Move*(zMax-zMin); eta=opt.Eta;
dfndz=min(-abs(c).*z0.*eta,dfdz); dfpdz=dfdz-dfndz; %Sensitivity separation
l1=0; l2=1e6;
while l2-l1 > 1e-4
  lmid = 0.5*(l1+l2);
  B = -dfndz./(dfpdz+lmid*dgdz);
  zCnd = zMin+(z0-zMin).*B.^eta;
  zNew = max(max(min(min(zCnd,z0+move),zMax),z0-move),zMin);
  if (g+dgdz'*(zNew-z0)>0),   l1=lmid;
  else,                       l2=lmid;  end
end
yk = dfdz-dfdz0; sk = zNew-z0;
al=0.75; zNew = al*zNew+(1-al)*z0; % Damping scheme
Change = max(abs(zNew-z0))/(zMax-zMin);
c=c+(sk.'*yk-c.'*sk.^2)/(sum(sk.^4)).*sk.^2; %Hessian Approx. (PSB)
```

To update of all design variables, we loop over all volume constraints, each time calling the UpdateSchemeSS subroutine using only information corresponding to the element indices, $\mathcal{E}_j$, associated with constraint $g_j(\mathbf{z})$. The "for loop", which can be found on lines 19–23 of the PolyDyna.m file (see Appendix C), is as follows:

```
%Update design variable and analysis parameters
for j=1:opt.NConstr
  Eid = cell2mat(opt.ElemInd(j));
  [z(Eid),dfdz0(Eid),c(Eid),Change(j)] = UpdateSchemeSS(...
      dfdz(Eid),dfdz0(Eid),c(Eid),g(j),dgdz(Eid,j),z(Eid),opt);
end
```

8 Numerical examples

This section presents several numerical examples to illustrate the use of PolyDyna. Appendix A provides a summary of all examples presented in this section, including the filenames used to generate the design domains, as well as material properties and design objective, among other parameters, required to run the code. To facilitate reproduction of the results, we provide all domain files together with PolyDyna. The first two examples are adopted from Zhao and Wang (2016), whereas the other examples are introduced here to test the ability of PolyDyna to solve problems with loads that change in direction or position and to solve problems considering

ground excitation. For all the problems solved below, the PolyScript file can be written using that shown in Appendix B as a template. Note that the PolyScript file in Appendix B uses a continuation on the RAMP penalty parameter, p_0, such that p_0 starts at 0 and increases by 1.5 every 25 iterations and up to a maximum value of 9 (see lines 65–72 of the PolyScript file). Moreover, to ensure that the HHT-α method is at least second-order accurate and unconditionally stable, all problems use $\alpha = 0.05$, $\beta = (1+\alpha)^2/4$, and $\gamma = (1+2\alpha)/2$.

8.1 Cantilever beam design under half-cycle sinusoidal load

We use this example to compare the optimized designs obtained with either mean dynamic compliance or mean strain energy. As depicted in Fig. 3, the design domain is a cantilever beam subjected to a half-cycle sinusoidal load applied at the center of the free edge of the beam.[9] The beam has length, $L = 8$ m, thickness, $h = 0.01$ m, and it is made of steel with Young's modulus, $E_0 = 200$ GPa, Poisson's ratio, $\nu = 0.3$, and mass density, $\rho_0 = 7800$ kg/m^3. Using PolyMesher, we discretize the design domain into 25,088 regular quadrilateral elements to obtain the designs presented next. To solve this problem, we use the input parameters shown in Table 5.

Figure 4 presents optimized designs obtained for different durations, t_f, of the applied load. Specifically, Fig. 4 (top) displays the results for mean compliance and Fig. 4 (bottom) displays those for mean strain energy. For each of the two objective functions, the optimized topologies obtained for $t_f = 0.05$ s (Fig. 4a) and $t_f = 0.03$ s (Fig. 4b) are almost the same. However, when $t_f = 0.01$ s (Fig. 4c), the optimized topologies significantly differ from those obtained for larger values of t_f.

When $t_f = 0.01$ s, both the mean compliance solution and the mean strain energy solution place a large amount of material toward the free edge of the beam, which provides inertial forces to counteract the fast applied load. Despite both having a large amount of material towards the free edge, the mean compliance solution connects this mass to the supports by means of horizontal members primarily subjected to bending, while the mean strain energy solution creates a bulb-like structure with a hinge close to the support. The beam-like members of the mean compliance solution help reduce the vertical deflection of the beam, while the bulb-like structure of the mean strain energy solution creates a mechanism so that the structure deforms mostly as a rigid body, reducing the strains, and thus minimizing the total strain energy of the system. To observe the difference in their dynamic response, Fig. 5 shows the deformed shape of the two solutions at various instances in time (i.e., for $t = 0.25t_f$, $t = 0.50t_f$, $t = 0.75t_f$, and $t = t_f$). The results show that, given the large mass at the tip of the cantilever, the mean compliance solution deflects less than the mean strain energy solution. However, the mean compliance solution deforms more than the mean strain energy solution. As a result, we expect the total strain energy for the mean compliance solution to be larger than that for the mean strain energy solution.

To verify the above hypothesis, we plot (for each of the designs) the vertical displacements at the point of load application as well as the total strain energy as a function of time. These results, which are depicted in Fig. 6, show that the dynamic responses for $t_f = 0.05$ s and $t_f = 0.03$ s are similar for each of the two objective functions, but differ considerably for $t_f = 0.01$ s. As observed from the displacement history, the design obtained using mean compliance deflects less than the design obtained using mean strain energy when $t_f = 0.01$ s, which is consistent with the deformed shapes shown in Fig. 5. The results also show that, although the mean compliance design deflects less than the mean strain energy design (for $t_f = 0.01$ s), the latter results in less strain energy due to the hinge that develops towards the support, which is also consistent with the deformed shapes shown in Fig. 5. The results for $t_f = 0.01$ s (Fig. 6c) also show that both designs continue deflecting downward even when the magnitude of the load begins decreasing after $t = 0.005$ s, which happens due to the large inertial forces that have accumulated up to that point and keep driving the structures downward. Interestingly, even when both structures keep deflecting after $t = 0.005$ s, the total strain energy reaches a peak value shortly after and begin to decrease. This can be attributed to the fact that the structures initially bend more due to the inertial effects in addition to the localized strains close to the point of load application, and after the magnitude of the load decreases, the localized strains close to the point of load application diminish, thus causing the total strain energy to decrease.

The results discussed above suggest that the objective function in dynamic topology optimization can play a significant role in the type of optimized designs that can be obtained, especially when the inertial forces are significant (which may happen when the applied load is applied at a fast rate of speed). Therefore, the objective function in dynamic topology optimization must be selected with caution to suit the desired design considerations.

[9] Although this example assumes the load duration to be equal to the time integration duration used to evaluate the objective function, they need not be equal. For example, the load duration could be set as $t = t_s$ and the load defined as $f(t) = f_0 \sin(\pi t/t_s)(1 - H(t - t_s))$, in which $H(\cdot)$ is the unit step function, while the duration of the dynamic event, t_f, could be set as $t_f = 2t_s$ and use the displacement field up to $t = t_f$ to evaluate the objective function.

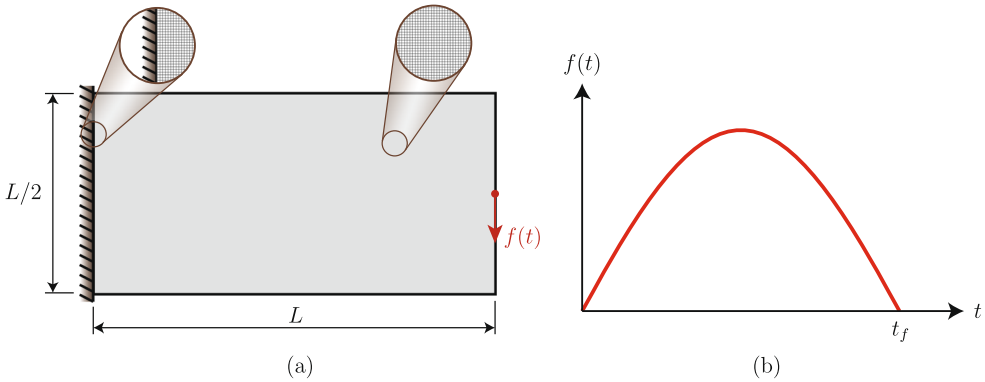


Fig. 3 Problem setup for the cantilever beam problem: **a** domain and boundary conditions and **b** applied load

Table 5 Input parameters used for the cantilever beam problem

Simulation time, `fem.Tmax`	0.05 s, 0.03 s, and 0.01 s
Number of time steps, `fem.NStep`	100
Young's modulus of solid material, `fem.E0`	200×10^9 Pa
Poisson's ratio of solid material, `fem.Nu0`	0.3
Mass density of solid material, `fem.rho`	7800 kg/m^3
Element thickness, `fem.Thickness`	0.01 m
Rayleigh damping parameters, `fem.Ag`	$[10,\ 1 \times 10^{-5}]$
Objective function, `fem.Obj`	`'Compliance'` or `'Energy'`
Tag for regular mesh, `fem.Reg`	1
Volume fraction limit, `opt.VolFrac`	0.5
Filter radius and filter exponent, R and q	0.2 and 1
Filter matrix, P	`PolyFilter(fem,R,q,'X')`

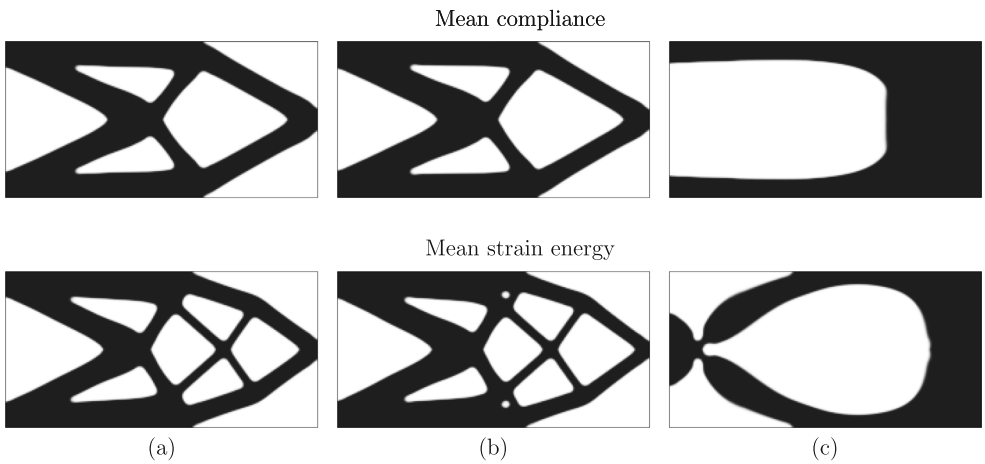


Fig. 4 Optimized topologies for the cantilever beam problem when the mean compliance is used as objective function (top) and when the mean strain energy is used as objective function (bottom). The results are obtained for **a** $t_f = 0.05$ s, **b** $t_f = 0.03$ s, and **c** $t_f = 0.01$ s

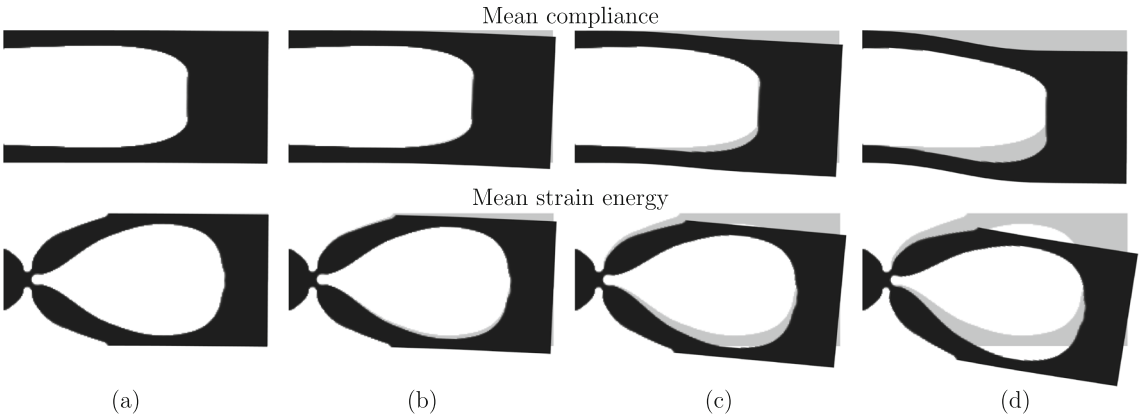


Fig. 5 Snapshots of the deformed shape of the cantilever structures obtained for $t_f = 0.01$ s. The top row shows the deformed cantilever structures obtained for mean compliance and the bottom row those obtained for mean strain energy. The deformed shapes are displayed for instances in time: **a** $t = 0.25t_f$, **b** $t = 0.50t_f$, **c** $t = 0.75t_f$, and **d** $t = t_f$. The undeformed structures are shown in color gray and the displacement field for the deformed structures has been amplified for visualization purposes

8.2 Clamped beam design under half-cycle cosine load

In this example, we use a beam clamped at the two ends and loaded vertically with a half-cycle cosine load applied at the lower part of the mid-span, as shown in Fig. 7. The beam has in-plane dimensions $L = 12$ m and $H = 2$ m, thickness, $h = 0.01$ m, and it is made of steel with Young's modulus, $E_0 = 200$ GPa, Poisson's ratio, $\nu = 0.3$, and mass density, $\rho_0 = 7800$ kg/m^3. For this example, the objective function is the mean squared displacement at the DOF where the load is applied. The optimized designs presented next are based on a mesh with 30,246 regular quadrilateral elements, which we generate using `PolyMesher`. We use the parameters shown in Table 6 as input data to create the `PolyScript` file to solve this problem.

Figure 8 shows three optimized designs, each obtained for a different value of the load duration, t_f. Specifically, Fig. 8a corresponds to $t_f = 0.5$ s, Fig. 8b corresponds to $t_f = 0.1$ s, and Fig. 8c corresponds to $t_f = 0.03$ s. As observed from the results, the optimized topologies are sensitive to t_f and the results obtained for $t_f = 0.5$ s (i.e., the longest load duration) resemble those from a static topology optimization formulation. When the load is applied at a faster rate of speed (i.e., for $t_f = 0.1$ s or $t_f = 0.03$ s), the optimizer places less material around the load application point, which we argue helps to accommodate the increased local deformation around the load

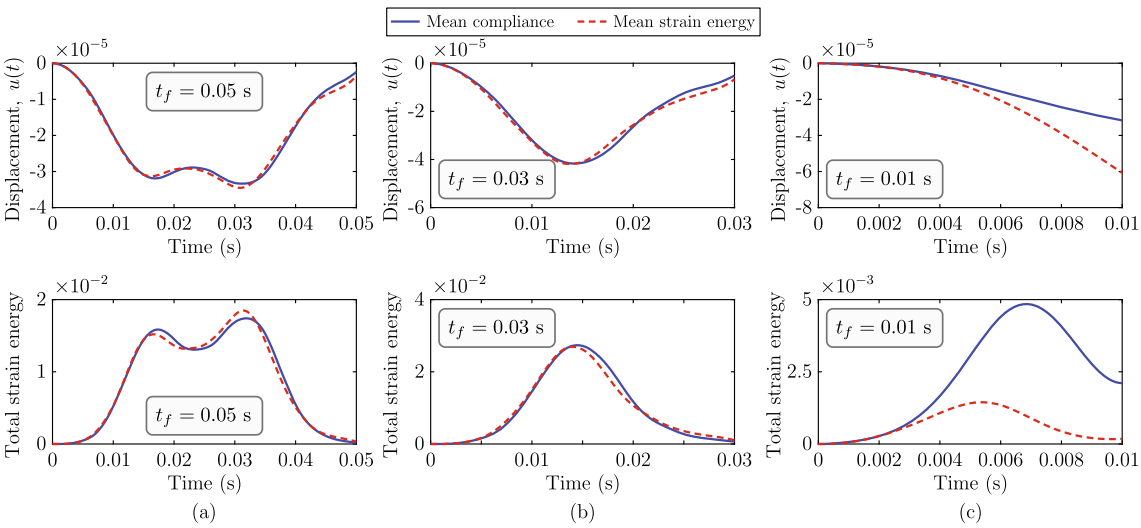


Fig. 6 Time histories of vertical deflection at the load application point (top) and of total strain energy (bottom), for each of the designs shown in Fig. 4. The results are depicted for **a** $t_f = 0.05$ s, **b** $t_f = 0.03$ s, and **c** $t_f = 0.01$ s

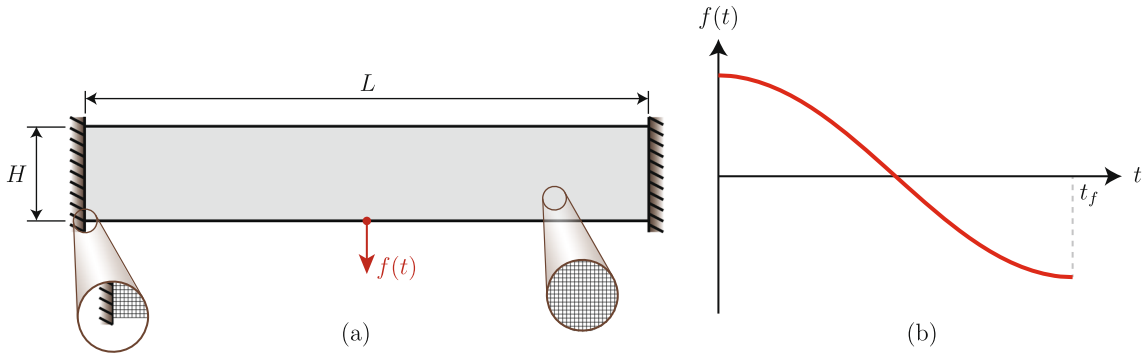


Fig. 7 Problem setup for the clamped beam problem: **a** domain and boundary conditions and **b** applied load

application point that occurs due to the increasing dynamic effects.

Figure 9 illustrates the dynamic response of each of the three designs shown in Fig. 8. The figures on the left column show the time history of vertical displacement at the load application point, those on the middle column show the total strain energy, and those on the right column show the total kinetic energy. The effects of damping are clearly manifested in the results obtained for $t_f = 0.5$ s, as the amplitude of the deflections, total strain energy, and total kinetic energy, all dampen over time. Similarly, damping effects seem to decrease for smaller values of t_f due to the shorter duration of the load. The left plots also report the static vertical displacement at the load application point, which helps us identify some dynamic amplification effects that occur during the first time steps of each simulation. These dynamic amplification effects certainly affect the response of the structure and cannot be captured by static optimization formulations.

8.3 Support structure design under rotating load

As illustrated in Fig. 10, this example deals with a square domain of side $L = 3$ m, supported at the bottom and subjected to a constant load, P, rotating at a prescribed angular frequency, ω. The square has thickness, $h = 0.05$ m, and it is made of aluminum with Young's modulus, $E_0 = 70$ GPa, Poisson's ratio, $\nu = 0.3$, and mass density, $\rho_0 = 2700$ kg/m^3. The purpose of this example is to investigate the effect of angular frequency in the optimized designs obtained using `PolyDyna`. To this end, we select the mean dynamic compliance as the objective function. Moreover, we use `PolyMesher` to discretize the domain using 20,022 regular quadrilateral elements. The `PolyScript` file to run this problem can be created using the information provided in Table 7.

In Fig. 11, we present three optimized designs, which we obtain for $\omega = 50\pi$ rad/s (1500 rpm), $\omega = 100\pi$ rad/s (3000 rpm), and $\omega = 200\pi$ rad/s (6000 rpm), respectively. The first two designs (Fig. 11a, b) are composed of two diagonal members, while the third (Fig. 11c) has an additional lateral bracing system to restrict the lateral movement of the structure.

To further understand the dynamic behavior of the optimized designs, Fig. 12 displays the time history of displacements, total strain energy, and total kinetic energy for each of the three topologies shown in Fig. 11. The displacements reported in these figures correspond to the

Table 6 Input parameters used for the clamped beam problem

Simulation time, `fem.Tmax`	0.5 s, 0.1 s, and 0.03 s
Number of time steps, `fem.NStep`	400
Young's modulus of solid material, `fem.E0`	200×10^9 Pa
Poisson's ratio of solid material, `fem.Nu0`	0.3
Mass density of solid material, `fem.rho`	7800 kg/m^3
Element thickness, `fem.Thickness`	0.01 m
Rayleigh damping parameters, `fem.Ag`	$[10,\ 1 \times 10^{-5}]$
Objective function, `fem.Obj`	`'U_DOF'`
Target DOF	`2*Load(1,1)`
Tag for regular mesh, `fem.Reg`	1
Volume fraction limit, `opt.VolFrac`	0.5
Filter radius and filter exponent, R and q	0.15 and 1
Filter matrix, P	`PolyFilter(fem,R,q,'Y')`

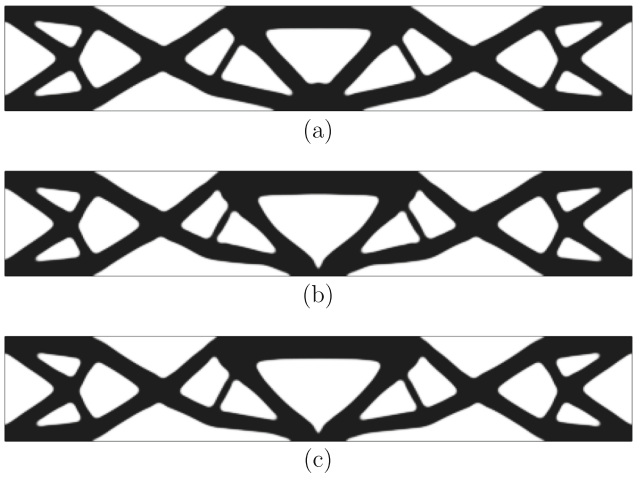


Fig. 8 Optimized topologies for the clamped beam problem obtained for **a** $t_f = 0.5$ s, **b** $t_f = 0.1$ s, and **c** $t_f = 0.03$ s

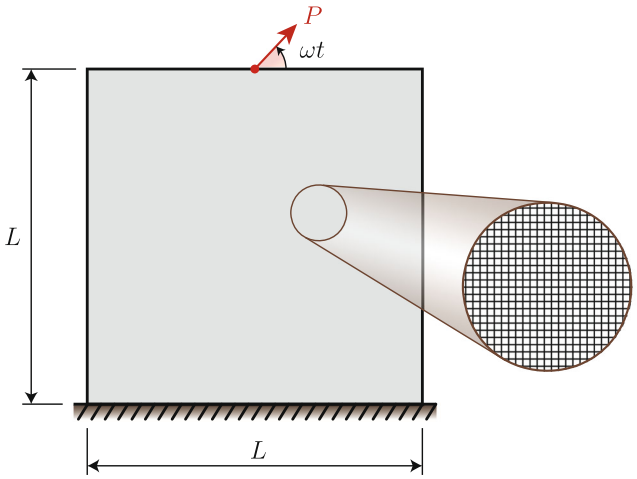


Fig. 10 Support structure domain and boundary conditions

horizontal displacement at the point of load application. The results in Fig. 12a show large dynamic effects that occur in the first 0.04 s, which are then dissipated due to external damping. Similarly, dynamic effects occur during the first time steps for $\omega = 100\pi$ rad/s (Fig. 12b) and $\omega = 200\pi$ rad/s (Fig. 12c). The results also show that the total strain energy as well as the total kinetic energy attain their maximum values early on in the simulations, before the dynamic effects are dampened out.

8.4 Bridge design under moving load

This example aims to illustrate how moving loads can easily be considered within PolyDyna. To this end, we consider a bridge of length $L = 30$ m and height $H = 13.5$ m subjected to a uniform load of length L_0 that moves over the deck of the bridge at a constant speed, V_0 (see Fig. 13). Unlike in the other examples, here we consider a passive region to represent the deck of the bridge and use

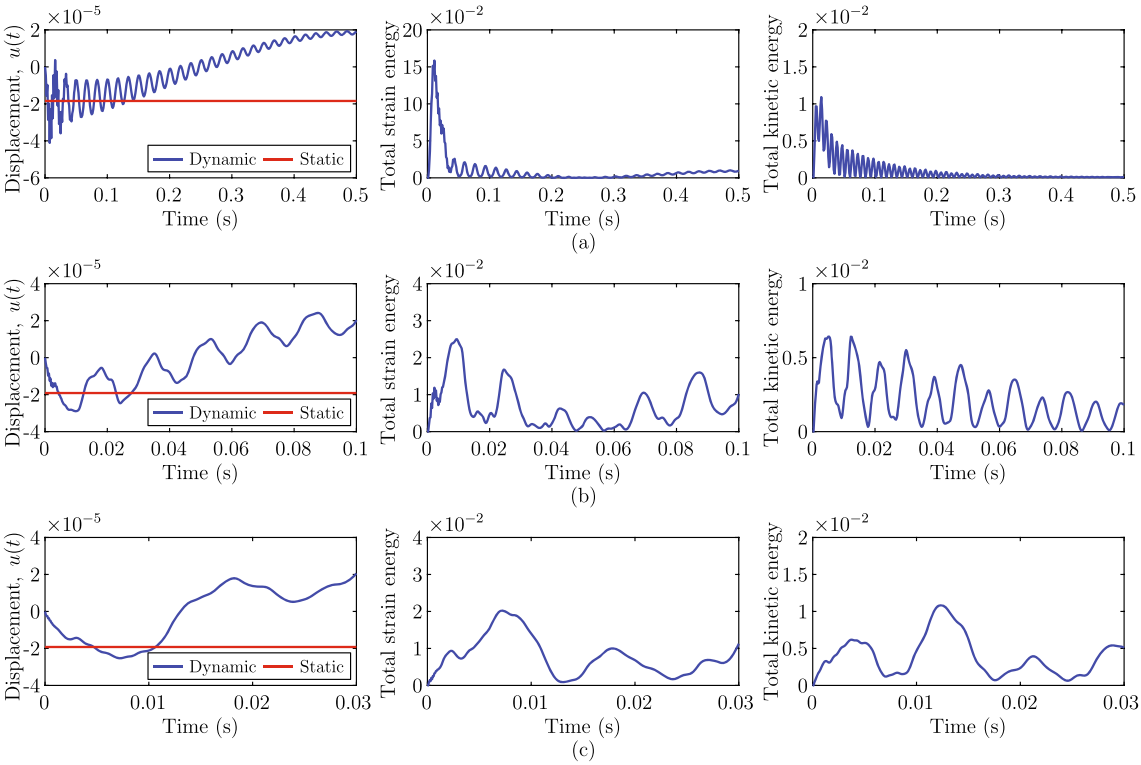


Fig. 9 Time histories of vertical deflection at the load application point (left), total strain energy (center), and total kinetic energy (right), corresponding to each of the designs shown in Fig. 8. The results are depicted for **a** $t_f = 0.5$ s, **b** $t_f = 0.1$ s, and **c** $t_f = 0.03$ s

Table 7 Input parameters used for the support structure problem

Simulation time, `fem.Tmax`	$10\pi/\omega$, with $\omega = 50\pi$, 100π, and 200π rad/s
Number of time steps, `fem.NStep`	150
Young's modulus of solid material, `fem.E0`	70×10^9 Pa
Poisson's ratio of solid material, `fem.Nu0`	0.33
Mass density of solid material, `fem.rho`	2700 kg/m^3
Element thickness, `fem.Thickness`	0.05 m
Rayleigh damping parameters, `fem.Ag`	$[50,\ 3 \times 10^{-5}]$
Objective function, `fem.Obj`	`'Compliance'`
Tag for regular mesh, `fem.Reg`	1
Volume fraction limit, `opt.VolFrac`	0.25
Filter radius and filter exponent, R and q	0.08 and 1
Filter matrix, P	`PolyFilter(fem,R,q,'Y')`

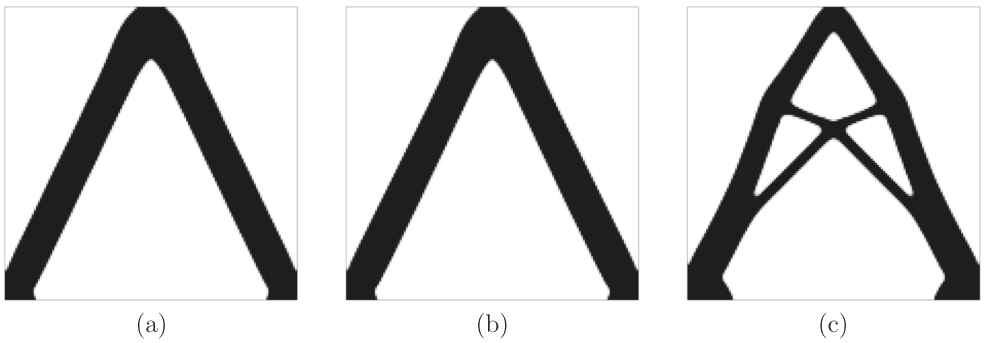


Fig. 11 Optimized topologies for the support structure under a rotating load for various values of angular frequency: **a** $\omega = 50\pi$ rad/s, **b** $\omega = 100\pi$ rad/s, and **c** $\omega = 200\pi$ rad/s

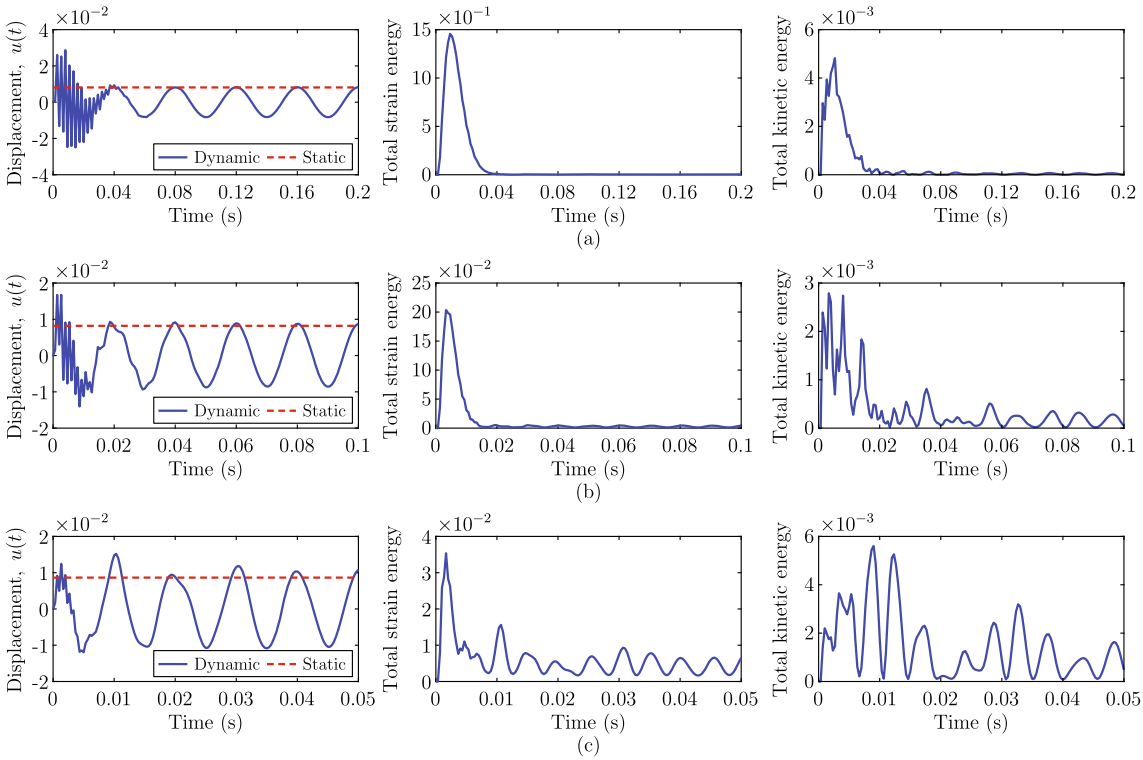


Fig. 12 Time histories of horizontal deflection at the load application point (left), total strain energy (center), and total kinetic energy (right), corresponding to each of the designs shown in Fig. 11. The results are depicted for **a** $\omega = 50\pi$ rad/s, **b** $\omega = 100\pi$ rad/s, and **c** $\omega = 200\pi$ rad/s

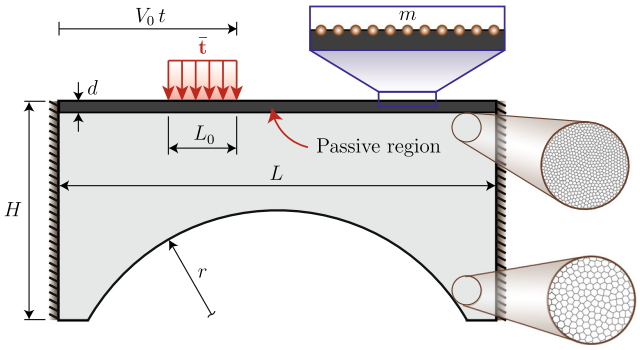


Fig. 13 Bridge domain and boundary conditions

PolyMesher to generate a graded mesh, which is coarse towards the bottom of the bridge and becomes fine towards its top. The graded mesh is used because we expect most of the complexity of the optimized structure to be located towards the top of the bridge. To represent the dead weight of the pavement and other non-structural elements, we also consider lumped masses, m, distributed uniformly across all elements of the deck surface (see Fig. 13), such that they add up to 0.18×10^6 kg. To create the PolyScript file that runs this problem, we refer to the input data provided in Table 8.

We investigate the effect of both the load speed, V_0, and the objective function, on the final designs obtained for the bridge. As in previous examples, we consider mean dynamic compliance and mean strain energy as the two candidate objective functions to design the bridge. We emphasize that in the context of dynamic topology optimization, the mean dynamic compliance solutions are meant to reduce deflections, but not necessarily to reduce the strains (and consequently the stresses) of the optimized structure. Conversely, the mean strain energy solutions are meant to reduce the strains in the structure, but are not necessarily meant to reduce their deflections. These observations are put to test in this example, as we compare the dynamic performance of the two types of solutions for the bridge under consideration.

Figure 14 shows the optimized designs obtained for mean dynamic compliance (left) and mean strain energy (right). The first observation from the results is that the designs obtained for mean dynamic compliance and mean strain energy are nearly identical for $V_0 = 30$ km/h, which is expected due to the fact that dynamic effects are not relevant at low speeds. However, one starts noticing differences in the two designs as V_0 increases. For example, as compared to the round arch-like structure obtained for mean dynamic compliance, the type of arches obtained for mean strain energy resemble an arch with an acutely pointed head. Besides that observation, there are other minor topological differences in the two types of design, and their effects in the dynamic response of the optimized structures have yet to be analyzed.

In order to understand the dynamic response of the two types of design (i.e., mean dynamic compliance or mean strain energy designs), Fig. 15 shows their dynamic response, including the vertical deflection at the mid-span of the bridge as well as the total strain energy and total kinetic energy, as a function of time. As observed from the results, the dynamic response for $V_0 = 30$ km/h is nearly identical for each of the two designs. However, the differences between the two types of designs become apparent for larger values of V_0. First, the results show that the mean strain energy solutions experience less vertical deflection than the mean dynamic compliance solutions (at least on average). As compared to the mean compliance solutions, those obtained using mean strain energy tend to experience less total strain energy and less total kinetic

Table 8 Input parameters used for the bridge problem

Simulation time, fem.Tmax	L/V_0, with $L = 30$ m $V_0 = 30,\ 60,\ 90$, and 120 km/h
Number of time steps, fem.NStep	100
Young's modulus of solid material, fem.E0	35×10^9 Pa
Poisson's ratio of solid material, fem.Nu0	0.25
Mass density of solid material, fem.rho	2400 kg/m^3
Element thickness, fem.Thickness	1 m
Rayleigh damping parameters, fem.Ag	$[2.5,\ 4.5 \times 10^{-4}]$
Objective function, fem.Obj	'Compliance' or 'Energy'
Lumped masses, fem.Mass	[Load(:,1), M], with M = 0.18e6*ones(NM,1)/NM kg and NM = size(Load,1)
Tag for regular mesh, fem.Reg	0
Volume constraint setting and passive regions	ConstraintsBridge
Filter radius and filter exponent, R and q	0.3 and 1
Filter matrix, P	PolyFilter(fem,R,q,'Y')

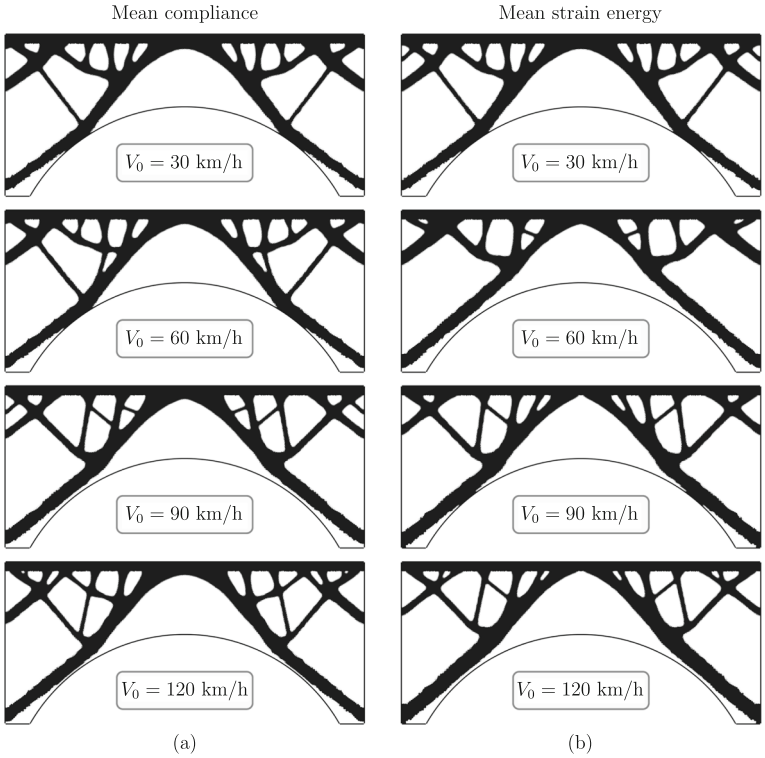


Fig. 14 Effect of load speed, V_0, on the optimized bridge designs obtained with **a** mean dynamic compliance and **b** mean strain energy. The load speeds range from 30 km/h to 120 km/h

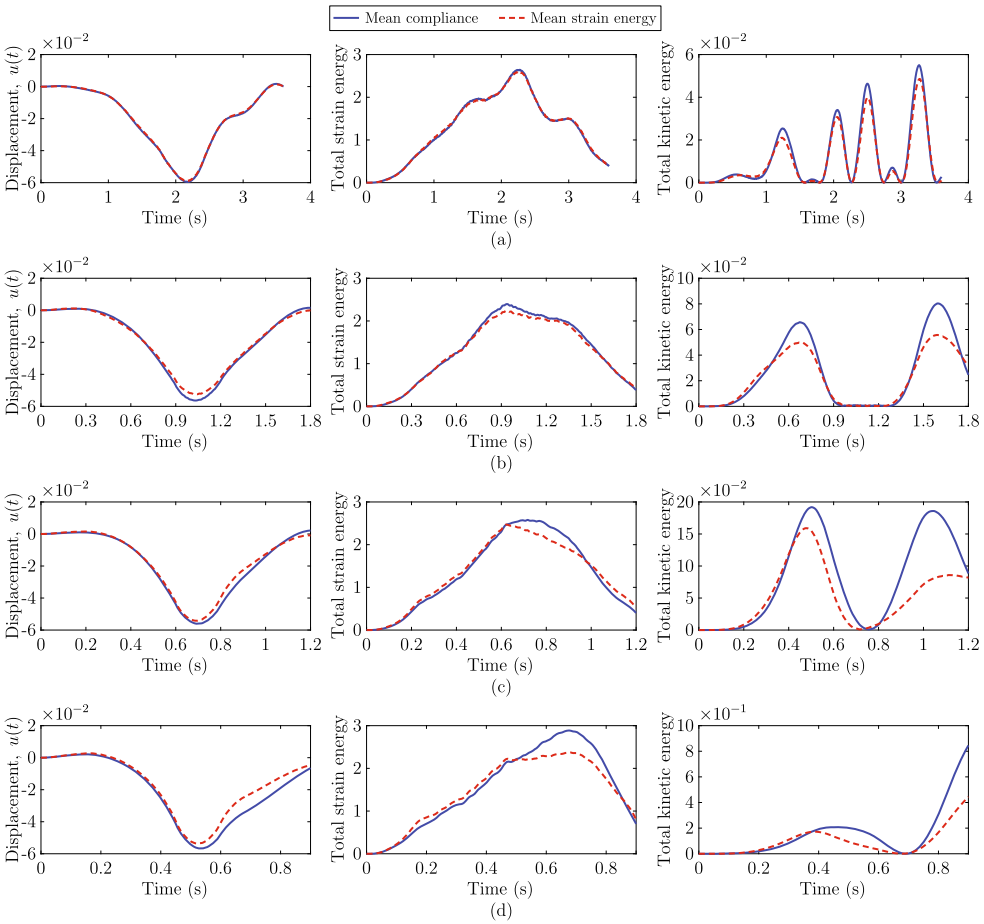


Fig. 15 Mid-span vertical deflection (left), total strain energy (middle), and total kinetic energy (right) for the optimized bridge designs of Fig. 14: **a** $V_0 = 30$ km/h, **b** $V_0 = 60$ km/h, **c** $V_0 = 90$ km/h, and **d** $V_0 = 120$ km/h

energy, which shows that the mean strain energy designs outperform the mean dynamic compliance designs. We highlight that small values of total strain energy indicate that overall the structure experiences a small deformation. Similarly, a small value of total kinetic energy is an indicator that the structure does not shake as much.

To gain a better understanding as to why the mean strain energy solutions deform less than the mean compliance solutions, Fig. 16 shows the deformed shapes, at different instances in time, for the two designs obtained for $V_0 = 120$ km/h. The results show that the deformed shapes at the beginning of each simulation (Fig. 16a) are not too different, which is in agreement with the results reported in Fig. 15d. When the moving load is close to the mid-span of the bridge (Fig. 16b, c), the mean strain energy solution concentrates most of the deformation close to the head of the acute arch, whereas the mean compliance solution spreads the deformation across a larger region. As a result, more strain energy develops in the mean compliance solution as compared to the mean strain energy solution. Due to the reduced dynamic response in the mean strain energy solution, as the load keeps moving forward, on average, the displacements for this design are smaller than those for the mean compliance design (Fig. 16d). These results suggest that mean strain energy should be considered as a candidate objective function when designing structures under moving loads. Of course, these are observations from a single test case and PolyDyna users could explore additional design problems to test such hypothesis.

8.5 Building design under ground excitation

The final example focuses on the design of a building subjected to a ground acceleration varying in time according to a sinusoidal function. Figure 17 depicts the problem setup, including the building geometry, ground acceleration, volume constraint specification, and passive region specification. Unlike the other examples, this example uses five regional volume constraints (see right-hand side of Fig. 17), which we use indirectly to control the maximum member size of the optimized structures (length-scale control through specification of regional volume constraints has been demonstrated by Sanders et al. (2018) and by Giraldo-Londoño et al. (2020)). The building domain is a truncated ellipse of height $H = 75$ m, width $B = 30$ m, and thickness $h = 1$ m. The building also has a lumped mass of magnitude M at the top. We use PolyMesher to discretize the design domain using 15,000 polygonal finite elements. Unlike the previous example, this one considers uniformly distributed polygonal finite elements. Table 9 provides all the required input data to create the PolyScript file to run this problem.

First, we discuss our approach to define both the regional volume constraints and the passive region for this problem. To define the volume constraints, we need to populate the fields opt.VolFrac and opt.ElemInd, which contain, respectively, the volume fraction limit and the element indices associated with each volume constraint, g_j. We recall that each entry of opt.VolFrac and opt.ElemInd corresponds to $\bar{v}_j$ and $\mathcal{E}_j$ in (24). In addition, to define the passive region, we need to populate the field fem.SElem. We obtain all the required fields for the definition of both the volume constraints and the passive region using the Matlab code shown below. Note that we call two subroutines, Centroids and Areas, which are also provided with PolyDyna because they become handy to define volume constraints and passive regions for a variety of problems beyond those discussed here. As their names indicate, subroutines Centroids and Areas compute the centroids and areas of all polygonal elements, respectively. The Matlab code shown below should be called in the PolyScript file right before the opt structure is defined.

```
NConstr = 5;                %Number of regional constraints
Hc = H/NConstr;             %Height of each sub-region
ElemCtrd = Centroids(fem); %Obtain the centroids of all elements
ElemArea = Areas(fem);      %Obtain the areas of all elements
ElemInd = cell(NConstr,1); %Initialize ElemInd cell array
for ii=1:NConstr %Find elements that belong to each sub-region
  ElemInd{ii} = find(abs(ElemCtrd(:,2)-(ii-1/2)*Hc)<=Hc/2);
end
sc = 0.8; a = sc*2/3*H; b = sc*B/2; c = sc*(H-a);
SElemInd = find(((ElemCtrd(:,2)-c)/a).^2+(ElemCtrd(:,1)/b).^2>=...
           1+30*eps)&(ElemCtrd(:,2)<=4/5*H)); %Elements in passive regions
for ii=1:NConstr %Remove elements from passive regions
  ElemInd{ii} = setdiff(ElemInd{ii},SElemInd);
end
Vmax = 0.5*sum(ElemArea(ElemInd{5})); % Max volume in each sub-region
for ii=1:NConstr %Volume fraction limit in each sub-region
  VolFrac(ii,1)=Vmax/sum(ElemArea(ElemInd{ii}));
end
```

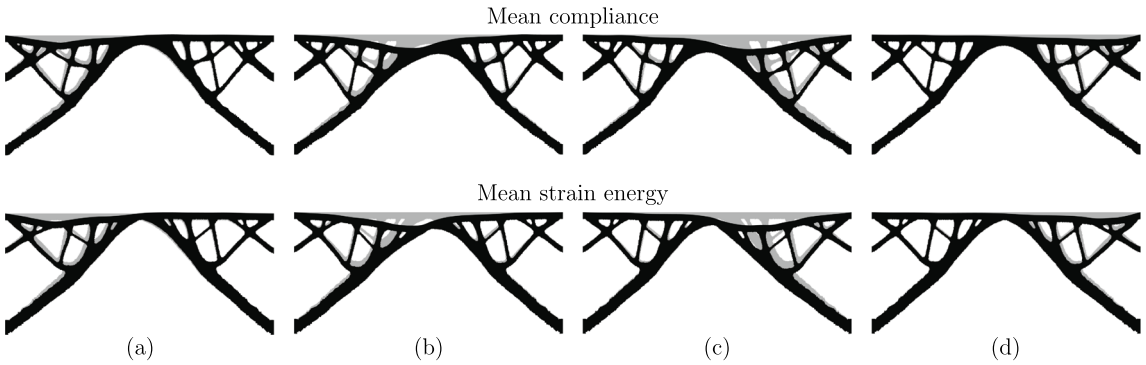


Fig. 16 Deformed shape of the bridges optimized using mean dynamic compliance (top) and mean strain energy (bottom), considering a design velocity, $V_0 = 120$ km/h, at various instances in time: **a** $t = 0.225$ s, **b** $t = 0.45$ s, **c** $t = 0.675$ s, and **d** $t = 0.9$ s. The undeformed structures are shown in color gray and the displacement field for the deformed structures has been amplified by a factor of 30

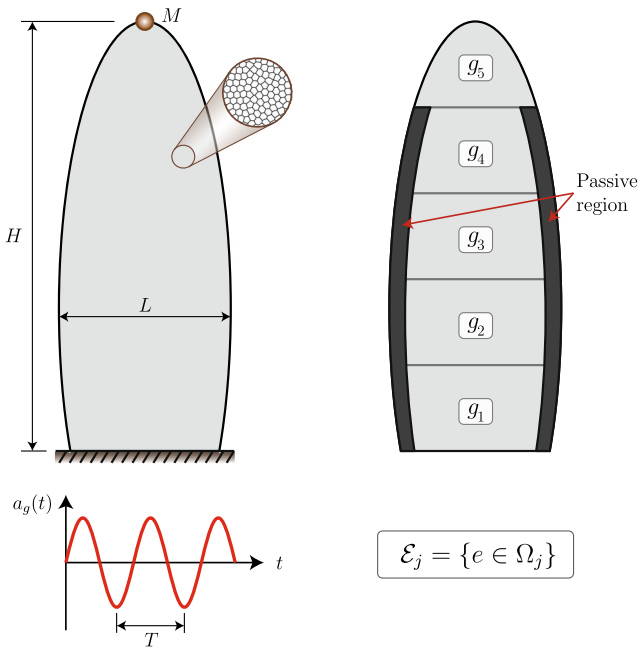


Fig. 17 Problem setup for the building design subjected to ground accelerations. The left figure shows the building domain and ground acceleration, and the right figure shows the regional volume constraint definition

Table 9 Input parameters used for the building problem

Simulation time, `fem.Tmax`	4.8 s
Number of time steps, `fem.NStep`	80
Young's modulus of solid material, `fem.E0`	35×10^9 Pa
Poisson's ratio of solid material, `fem.Nu0`	0.25
Mass density of solid material, `fem.rho`	2400 kg/m^3
Element thickness, `fem.Thickness`	1 m
Rayleigh damping parameters, `fem.Ag`	$[2,\ 2 \times 10^{-6}]$
Objective function, `fem.Obj`	`'U_DOF'`
Lumped masses, `fem.Mass`	`[N, M]`, with `N = find(sqrt(Node(:,1).^2+Node(:,2).^2)<1e-3)` and `M = 0.2e6, 0.3e6, 0.4e6, 0.5e6, 0.6e6` kg
Tag for regular mesh, `fem.Reg`	0
Volume constraint setting and passive regions	`ConstraintsBuilding`
Filter radius and filter exponent, R and q	0.75 and 1
Filter matrix, P	`PolyFilter(fem,R,q,'Y')`

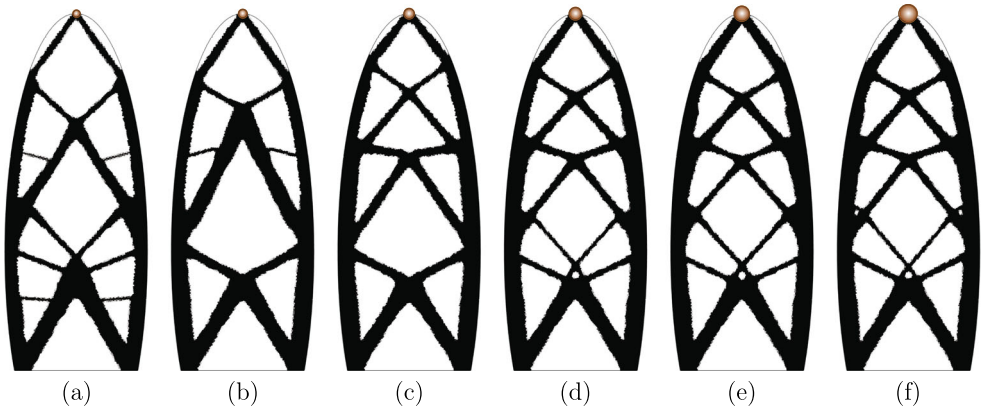


Fig. 18 Optimized topologies obtained for the building domain subjected to a sinusoidal ground acceleration of frequency, $\omega = 2.5\pi$ rad/s, and for various values of the lumped mass: **a** $M = 0.1 \times 10^6$ kg, **b** $M = 0.2 \times 10^6$ kg, **c** $M = 0.3 \times 10^6$ kg, **d** $M = 0.4 \times 10^6$ kg, **e** $M = 0.5 \times 10^6$ kg, and **f** $M = 0.6 \times 10^6$ kg

Now, we discuss how to impose the sinusoidal ground acceleration to the model. In general, to assign ground acceleration in PolyDyna, we need to populate the field fem.ag with the magnitude of the acceleration at every time step. For the case of the sinusoidal ground acceleration considered in this example, this field is populated as

```
fem.ag = 5*sin(w*t);
t = linspace(0,Tmax,NStep+1);
```

where w is the angular frequency of the sinusoidal excitation, Tmax is the maximum simulation time, and NStep is the number of time steps. Note the generality in the way the ground accelerations can be imposed. If a user of PolyDyna wants to use acceleration data from an actual earthquake, the field fem.ag needs to be populated using the value of the actual ground acceleration at every time step.

Now that we have discussed some of the implementation details for the setup of this problem, Fig. 18 presents several optimized topologies obtained for a ground acceleration with frequency, $\omega = 2.5\pi$ rad/s (i.e., for `fem.ag = 5*sin(2.5*pi*t)`), and for various values of the lumped mass, M. The results show that the optimized structure is highly sensitive to the magnitude of M. First, one can observe that the two diagonal members connected to the lumped mass become thicker as M increases. This is expected because the inertial forces transmitted from the lumped mass to the structure increase in magnitude as M increases. A similar trend can be observed in a study by Filipov et al. (2016), who added a lumped mass at the tip of a cantilever beam while optimizing for its natural frequency. Another interesting aspect from the results in Fig. 18 is the change in the lateral bracing system topology as M increases. For instance, when $M = 0.1 \times 10^6$ kg or $M = 0.2 \times 10^6$ kg (Fig. 18a, b), it is possible to identify two main bracing systems, one at the first quarter height of the building, and another starting at the first quarter height of the building and ending at the location of the lumped mass. For larger values of M (Fig. 18c–f), an additional lateral bracing system develops at the mid-height of the building to help transmit the increasing inertial forces to the supports.

9 Concluding remarks

In this paper, we have discussed details of both the theoretical framework as well as the numerical implementation of a topology optimization formulation considering dynamic loading effects. The numerical implementation led to an educational Matlab code called PolyDyna, which is written on top of PolyTop (Talischi et al. 2012b). The new software is intended to be a continuation of a family of educational codes for topology optimization on unstructured finite element meshes. PolyDyna updates the design variables using a design variable update scheme that combines the features of both the ZPR design variable update scheme (Zhang et al. 2018) and the sensitivity separation approach by Jiang et al. (2021), which enables PolyDyna to solve both self- and non-self-adjoint problems with an arbitrary number of volume constraints in an efficient manner. Therefore, the code is written such that one can impose an arbitrary number of regional volume constraints as well as passive regions. Lumped masses at specified nodes are also accounted for within the numerical implementation.

The sensitivity analysis in PolyDyna is based on the *discretize-then-differentiate* approach, so that the adjoint sensitivity analysis is conducted on the discretized (both

in space and time) topology optimization statement. The *discretize-then-differentiate* approach avoids consistency errors that arise when the adjoint method is used while considering time as a continuous variable. To make the sensitivity derivation as general as possible, the time integration scheme is based on the HHT-α method, a generalization of the classical Newmark-β method. The HHT-α has numerical damping controlled by a parameter, α, which dampens high-frequency modes while leaving low-frequency basically unaffected. The HHT-α reduces to the Newmark-β method for $\alpha = 0$, so the user can run the code using $\alpha = 0$ if desired.

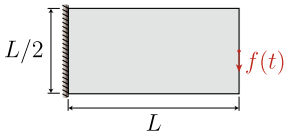
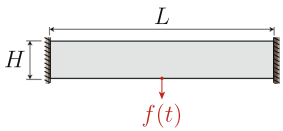
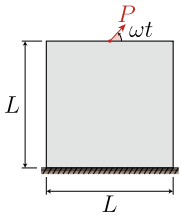
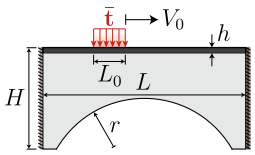
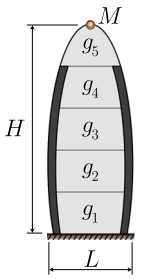
The software (`PolyDyna`) considers arbitrary dynamic loading, which can either vary in magnitude, direction, or position, and also considers ground acceleration. The dynamic loads can be assigned while defining the problem domain and boundary conditions in `PolyMesher`. To assign these loads, one needs to populate the `Load` field in `PolyMesher` with the magnitudes of the applied load at every time step. Moreover, to assign ground accelerations, the user only needs to populate the `fem.ag` field with values of ground acceleration for each time step. The generality of the loading setup is useful to solve a wide variety of dynamic topology optimization problems, as demonstrated by the examples.

In summary, `PolyDyna` offers the freedom to choose between several candidate objective functions, including mean dynamic compliance, mean strain energy, and mean squared displacement at a prescribed DOF. Additional objective functions can be incorporated into the code by modifying just a few lines of code. The results indicate that the type of objective function in dynamic topology optimization problems plays an important role, and it must be chosen with caution depending on the type of design that is desired. For some of the problems discussed in this work, the results obtained using either mean dynamic compliance or mean strain energy (which are equivalent in a static formulation) lead to significant differences in topology, especially when the inertial forces are considerable. One example that demonstrates these differences is the cantilever beam problem, which shows interesting topological differences when the load is applied at a fast rate (of speed). The bridge example results also show that subtle topological differences between mean dynamic compliance and mean strain energy results can have important implications in the dynamic response of the structure.

Appendix A: Library of examples

Table 10 summarizes of all examples discussed in this manuscript. The first column depicts the domain geometries for all examples, while the second column shows the name of the domain files needed by `PolyMesher` to generate the finite element meshes. The last column provides a description for each of the problems, including dimensions of the domains, magnitude of the dynamic loads, material properties, and filter radius. These problems can be verified with the electronic supplementary material (ESM) provided with this paper. Moreover, the user can easily modify these examples or create new ones using `PolyDyna`.

Table 10 Examples provided with PolyDyna

Domain	PolyMesher domain file	Description
	@CantileverDomain	• Dimensions: $L = 8$ m, $h = 0.01$ m • Load: $f(t) = f_0 \sin(\pi t/t_f)$, $f_0 = 1$ kN • Material: $E_0 = 200$ GPa, $\nu_0 = 0.3$, $\rho_0 = 7800$ kg/m^3 • Damping parameters: $\alpha_r = 10$, $\beta_r = 1 \times 10^{-5}$ • Filter: $R = 0.2$ m, $q = 1$
	@ClampedDomain	• Dimensions: $L = 12$ m, $H = 2$ m, $h = 0.01$ m • Load: $f(t) = f_0 \cos(\pi t/t_f)$, $f_0 = 1$ kN • Material: $E_0 = 200$ GPa, $\nu_0 = 0.3$, $\rho_0 = 7800$ kg/m^3 • Damping parameters: $\alpha_r = 10$, $\beta_r = 1 \times 10^{-5}$ • Filter: $R = 0.15$ m, $q = 1$
	@SupportDomain	• Dimensions: $L = 3$ m, $h = 0.05$ m • Load: $\mathbf{f}(t) = [P\cos(\omega t), \quad P\sin(\omega t)]^T$, $P = 1000$ kN • Material: $E_0 = 70$ GPa, $\nu_0 = 0.3$, $\rho_0 = 2700$ kg/m^3 • Damping parameters: $\alpha_r = 50$, $\beta_r = 3 \times 10^{-5}$ • Filter: $R = 0.08$ m, $q = 1$
	@BridgeDomain	• Dimensions: $L = 30$ m, $H = 13.5$ m, $r = 15$ m, $y_c = 7.5$ m, $h = 1$ m • Load: $\bar{\mathbf{t}} = 500$ kN/m moving at speed V_0 (variable) • Material: $E_0 = 35$ GPa, $\nu_0 = 0.3$, $\rho_0 = 2400$ kg/m^3 • Damping parameters: $\alpha_r = 2.5$, $\beta_r = 4.5 \times 10^{-4}$ • Filter: $R = 0.3$ m, $q = 1$
	@BuildingDomain	• Dimensions: $L = 30$ m, $H = 75$ m, $2a = 4H/3$ (major axis), $2b = L$ (minor axis), $h = 1$ m • Load: $a_g(t) = 5\sin(2.5\pi t)$ m/s^2 (ground acceleration) • Material: $E_0 = 35$ GPa, $\nu_0 = 0.3$, $\rho_0 = 2400$ kg/m^3 • Damping parameters: $\alpha_r = 2$, $\beta_r = 2 \times 10^{-6}$ • Filter: $R = 0.75$ m, $q = 1$

Appendix B: `PolyScript`

```
%------------------------------ PolyScript -------------------------------%
% Ref: O Giraldo-Londono, GH Paulino, "PolyDyna: A Matlab implementation  %
% for topology optimization of structures subjected to dynamic loads",    %
% Structural and Multidisciplinary Optimization, 2021                     %
% DOI http://dx.doi.org/10.1007/s00158-021-02859-6                        %
%-------------------------------------------------------------------------%
clear; clc; close all
restoredefaultpath; addpath(genpath('./')); %Use all folders and subfolders
%% ------------------------------------------------------ CREATE 'fem' STRUCT
global NStep Tmax Tf
Tmax = .05; Tf = Tmax; NStep = 100;
[Node,Element,Supp,Load] = Mesh_Cantilever(25000); close all;
alpha = 0.05; beta = (1+alpha)^2/4; gamma = (1+2*alpha)/2; % HHT-alpha
O = zeros(2*size(Node,1),1);
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
  'Mass',[],...                 % Array of lumped masses
  'SElem',[],...                % Elements in passive solid regions
  'u0',O,...                    % Initial displacement vector
  'v0',O,...                    % Initial velocity vector
  'Thickness',0.01,...          % Element thickness
  'E0',200e9,...                % Young's modulus of solid material
  'Nu0',0.3,...                 % Poisson's ratio of solid material
  'rho',7800,...                % Mass density of solid material (kg/m^3)
  'Ar',[10,1e-5],...            % Rayleigh damping param. C=Ar(1)*M+Ar(2)*K
  'ag',[],...                   % Ground acceleration
  'Tmax',Tmax,...               % Simulation time
  'NStep',NStep,...             % Total number of steps
  'alpha', alpha,...            % alpha parameter for HHT-alpha method
  'beta', beta,...              % beta parameter for HHT-alpha method
  'gamma',gamma,...             % gamma parameter for HHT-alpha method
  'Obj','Compliance',...        % Objective function (e.g., 'Compliance')
  'LL',[],...                   % Vector for UDOF
  'Reg',1 ...                   % Tag for regular meshes
   );
%% ------------------------------------------------------ CREATE 'opt' STRUCT
R = 0.2; q = 1;
VolFrac = 0.5;
fem.SElem = []; % Elements in passive solid regions
ElemInd{1} = (1:fem.NElem)'; % Element indices for volume constraint j
P = PolyFilter(fem,R,q,'X');
zIni = ones(size(P,2),1);
for ii=1:length(VolFrac); zIni(ElemInd{ii})=VolFrac(ii); end % Initial DVs
opt = struct(...
  'zMin',0,...                  % Lower bound for design variables
  'zMax',1.0,...                % Upper bound for design variables
  'zIni',zIni,...               % Initial design variables
  'MatIntFnc',0,...             % Handle to material interpolation fnc.
  'P',P,...                     % Matrix that maps design to element vars.
  'VolFrac',VolFrac,...         % Arrat if specified volume fraction const.
  'NConstr',size(VolFrac,1),...% Number of volume constraints
  'ElemInd',{ElemInd},...       % Element indices assoc. with each constr.
```

```
   'Tol',0.001,...                % Convergence tolerance on design vars.
   'MaxIter',25,...               % Max. number of optimization iterations
   'Move',0.2,...                 % Allowable move step in OC update scheme
   'Eta',0.5 ...                  % Exponent used in OC update scheme
    );
%% ------------------------------------------------------- RUN 'PolyDyna'
fem = preComputations(fem); %Run preComputations before running PolyDyna
figure; B = 1; p_i = 0:1.5:9; tic;
for ii=1:length(p_i)          %Continuation on the penalty parameter
  if ii==length(p_i); opt.MaxIter = 100; end
  disp(['current p: ', num2str(p_i(ii)), '  current B: ', num2str(B)]);
  opt.MatIntFnc = @(y)MatIntFnc(y,'RAMP-H1',[p_i(ii),B,0.5]);
  [opt.zIni,V,E,fem] = PolyDyna(fem,opt);
  B = min(B+2,10);
end
t0=toc;
if t0<=60, fprintf('Optimization time: %i seconds \n', round(t0))
elseif t0<=3600, fprintf('Optimization time: %1.1f minutes \n', t0/60)
elseif t0<=86400, fprintf('Optimization time: %1.1f hours \n', t0/3600)
else, fprintf('Optimization time: %1.1f days \n', t0/86400)
end
%-------------------------------------------------------------------------%
```

Appendix C: PolyDyna

```
%------------------------------ PolyDyna ---------------------------------%
% Ref: O Giraldo-Londono, GH Paulino, "PolyDyna: A Matlab implementation  %
% for topology optimization of structures subjected to dynamic loads",    %
% Structural and Multidisciplinary Optimization, 2021                     %
% DOI http://dx.doi.org/10.1007/s00158-021-02859-6                        %
%-------------------------------------------------------------------------%
function [z,V,E,fem] = PolyDyna(fem,opt)
Iter=0; Tol=opt.Tol*(opt.zMax-opt.zMin);z=opt.zIni; P=opt.P;
Change = 2*opt.Tol;
[E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
[FigHandle] = InitialPlot(fem,V);
c = 0.001*ones(fem.NElem,1); dfdz0=0.*c; % Initialize sens. separ. params
while (Iter<opt.MaxIter) && (max(Change)>Tol) %Optimization iterations
  Iter = Iter + 1;
  %Compute cost functionals and sensitivities
  [f,dfdz,fem] = ObjectiveFnc(fem,E,V,dEdy,dVdy,P);
  [g,dgdz,fem] = ConstraintFnc(fem,opt,E,V,opt.VolFrac,dEdy,dVdy,P);
  %Update design variable and analysis parameters
  for j=1:opt.NConstr
    Eid = cell2mat(opt.ElemInd(j));
    [z(Eid),dfdz0(Eid),c(Eid),Change(j)] = UpdateSchemeSS(...
        dfdz(Eid),dfdz0(Eid),c(Eid),g(j),dgdz(Eid,j),z(Eid),opt);
  end
  [E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
  %Output results
  fprintf('It: %3i \t Obj.: %1.3f\t Max. Const.: %1.3f\t Change: %1.3f\n',...
           Iter,f,max(g),max(Change));
  set(FigHandle,'FaceColor','flat','CData',1-V); drawnow
end
fprintf('Optimized Objective: %1.3e\n',f/fem.SF);
%------------------------------------------------------- OBJECTIVE FUNCTION
function [f,dfdz,fem] = ObjectiveFnc(fem,E,V,dEdy,dVdy,P)
[U,Ud,Udd,M,C,K,fem] = FEM_Dyna(fem,E,V); %Run dynamics code
dfdV = zeros(size(V));
dfdE = zeros(size(V));
if strcmp(fem.Obj,'Compliance')==1 %Mean compliance
  f = trace((fem.Fext+fem.Fa)'*U);
  dfdU = fem.Fext+fem.Fa;
  dfdV = cumsum(sum(fem.Fa0.*U(fem.eDof,:),2));
  dfdV = dfdV(cumsum(fem.ElemNDof));
  dfdV = [dfdV(1);dfdV(2:end)-dfdV(1:end-1)];
elseif strcmp(fem.Obj,'Energy')==1 %Mean strain energy
  f = 0.5*trace(U'*K*U);
  dfdU = K*U;
  temp = cumsum(U(fem.i,:).*fem.k0.*U(fem.j,:));
  temp = temp(cumsum(fem.ElemNDof.^2),:);
  dfdE = 0.5*sum([temp(1,:);temp(2:end,:)-temp(1:end-1,:)],2);
elseif strcmp(fem.Obj,'U_DOF')==1 %Mean square displacement at DOF
  f = sum((fem.LL'*U).^2);
  dfdU = 2*(fem.LL'*U).*fem.LL;
end
%Sensitivity analysis using adjoint method
Adj_Vec = AdjointProblem(fem,M,C,K,dfdU); %Solve adjoint problem
al = fem.alpha;
m0 = sparse(fem.iK0,fem.jK0,fem.m0);
k0 = sparse(fem.iK0,fem.jK0,fem.k0);
U = [U(:,1)+fem.Ar(2)*Ud(:,1),(1-al)*(U(:,2:end)+fem.Ar(2)*Ud(:,2:end))+...
```

```
                                al*(U(:,1:end-1)+fem.Ar(2)*Ud(:,1:end-1))];
Udd = [Udd(:,1)+fem.Ar(1)*Ud(:,1),Udd(:,2:end)+...
       fem.Ar(1)*((1-al)*Ud(:,2:end)+al*Ud(:,1:end-1))];
Fa0 = [fem.Fa0(:,1), (1-al).*fem.Fa0(:,2:end)+al*fem.Fa0(:,1:end-1)];
for ii=1:fem.NStep+1
  dRdV = m0*Udd(fem.eDof,ii)-Fa0(:,ii);
  dRdV = sparse(fem.eDof,fem.DofE,dRdV);
  dRdE = k0*U(fem.eDof,ii);
  dRdE = sparse(fem.eDof,fem.DofE,dRdE);
  dfdV = dfdV + (Adj_Vec(fem.FreeDofs,ii)'*dRdV(fem.FreeDofs,:))';
  dfdE = dfdE + (Adj_Vec(fem.FreeDofs,ii)'*dRdE(fem.FreeDofs,:))';
end
if ~isfield(fem,'SF'); fem.SF=fem.NStep/f; end %Normalization factor
s = fem.SF;
f = s*f./fem.NStep; dfdV = s*dfdV./fem.NStep; dfdE = s*dfdE./fem.NStep;
dfdz = P'*(dEdy.*dfdE + dVdy.*dfdV); % Chain rule for sensitivity analysis
%------------------------------------------------------ CONSTRAINT FUNCTION
function [g,dgdz,fem] = ConstraintFnc(fem,opt,E,V,VolFrac,dEdy,dVdy,P)
g = zeros(opt.NConstr,1);
dgdV = zeros(fem.NElem,opt.NConstr); dgdE = dgdV;
for j=1:opt.NConstr
  Eid = cell2mat(opt.ElemInd(j));
  g(j) = sum(fem.ElemArea(Eid).*V(Eid))/sum(fem.ElemArea(Eid))-VolFrac(j);
  dgdV(Eid,j) = fem.ElemArea(Eid)/sum(fem.ElemArea(Eid));
end
dgdz = P'*(dEdy.*dgdE + dVdy.*dgdV);
%---------------------------------------- SENSITIVITY SEPARATION UPDATE
function [zNew,dfdz,c,Change] = UpdateSchemeSS(dfdz,dfdz0,c,g,dgdz,z0,opt)
zMin=opt.zMin; zMax=opt.zMax;
move=opt.Move*(zMax-zMin); eta=opt.Eta;
dfndz=min(-abs(c).*z0.*eta,dfdz); dfpdz=dfdz-dfndz; %Sensitivity separation
l1=0; l2=1e6;
while l2-l1 > 1e-4
  lmid = 0.5*(l1+l2);
  B = -dfndz./(dfpdz+lmid*dgdz);
  zCnd = zMin+(z0-zMin).*B.^eta;
  zNew = max(max(min(min(zCnd,z0+move),zMax),z0-move),zMin);
  if (g+dgdz'*(zNew-z0)>0),  l1=lmid;
  else,                      l2=lmid;  end
end
yk = dfdz-dfdz0; sk = zNew-z0;
al=0.75; zNew = al*zNew+(1-al)*z0;           %Damping scheme
Change = max(abs(zNew-z0))/(zMax-zMin);
c=c+(sk.'*yk-c.'*sk.^2)/(sum(sk.^4)).*sk.^2; %Hessian approximation (PSB)
%------------------------------------------------------------- INITIAL PLOT
function [handle] = InitialPlot(fem,z0)
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
patch('Faces',Faces,'Vertices',fem.Node,'FaceVertexCData',0.*z0,...
      'FaceColor','flat','EdgeColor','k','linewidth',2);
handle = patch('Faces',Faces,'Vertices',fem.Node,'FaceVertexCData',...
                1-z0,'FaceColor','flat','EdgeColor','none');
axis equal; axis off; axis tight; colormap(gray); caxis([0 1]);
%-------------------------------------------------------------------------%
```

Appendix D: `FEM_Dyna`

```
%------------------------------ FEM_Dyna ---------------------------------%
% Ref: O Giraldo-Londono, GH Paulino, "PolyDyna: A Matlab implementation  %
% for topology optimization of structures subjected to dynamic loads",    %
% Structural and Multidisciplinary Optimization, 2021                     %
% DOI http://dx.doi.org/10.1007/s00158-021-02859-6                        %
%-------------------------------------------------------------------------%
function [un,vn,an,M,C,K,fem] = FEM_Dyna(fem,E,V)
al = fem.alpha; B = fem.beta;  g = fem.gamma; % HHT-alpha parameters
DT = fem.Tmax/fem.NStep;  % Time increment
fDOF = fem.FreeDofs;
an = zeros(2*fem.NNode,fem.NStep+1); un = an; vn = an;
un(:,1) = fem.u0; vn(:,1) = fem.v0; % Initial conditions
[M,C,K] = GlobalMCK(fem,E,V); % Compute stiffness and mass matrices
if ~isempty(fem.Mass)
  DOF = [2*fem.Mass(:,1)-1;2*fem.Mass(:,1)];
  M(DOF,DOF)= M(DOF,DOF) + [fem.Mass(:,2);fem.Mass(:,2)];
end
if ~isempty(fem.ag)
  fem.Fa = -M*ones(2*fem.NNode,1).*fem.ag;
  Ft = fem.Fext+fem.Fa;
else
  Ft = fem.Fext;
end
M1 = M+(1-al)*g*DT.*C+(1-al)*B*DT^2.*K;
an(fDOF,1) = M(fDOF,fDOF)\(Ft(fDOF,1)-K(fDOF,fDOF)*fem.u0(fDOF)...
             -C(fDOF,fDOF)*fem.v0(fDOF)); % Initialize accelerat. vector
for It=1:fem.NStep % Loop over time steps
  Fext = -C*(vn(:,It)+DT*an(:,It)*(1-al)*(1-g))-...
          K*(an(:,It)*(1/2-B)*(1-al)*DT^2+vn(:,It)*(1-al)*DT+un(:,It))+...
          (1-al)*Ft(:,It+1)+al*Ft(:,It);
  if It==1
    [an(fDOF,It+1),L,s] = SolveLinSys(M1(fDOF,fDOF),Fext(fDOF));
    fem.L = L; fem.s = s; % Store Cholesky decomposition information
  else
    an(fDOF(s),It+1) = L'\(L\Fext(fDOF(s)));
  end
  un(:,It+1) = un(:,It)+DT*vn(:,It)+(1/2-B)*DT^2*an(:,It)+B*DT^2*an(:,It+1);
  vn(:,It+1) = vn(:,It)+(1-g)*DT*an(:,It)+g*DT*an(:,It+1);
end
%% ------------------------------------ GLOBAL STIFFNESS AND MASS MATRICES
function [M,C,K] = GlobalMCK(fem,E,V)
K = sparse(fem.i,fem.j,E(fem.e).*fem.k0); % Assemble stiffness matrix
K = (K+K')/2;
M = sparse(fem.i,fem.j,V(fem.e).*fem.m0); % Assemble mass matrix
M = (M+M')/2;
C = fem.Ar(1)*M + fem.Ar(2)*K;            % Compute damping matrix
%% ---------------------- SOLVE LINEAR SYSTEM USING CHOLESKY DECOMPOSITION
function [U, L, s] = SolveLinSys(K,F)
[L,~,s] = chol(K,'lower','vector');
U(s,:) =L'\(L\F(s,:));
%-------------------------------------------------------------------------%
```

Appendix E: AdjointProblem

```
%--------------------------- AdjointProblem ------------------------------%
% Ref: O Giraldo-Londono, GH Paulino, "PolyDyna: A Matlab implementation  %
% for topology optimization of structures subjected to dynamic loads",    %
% Structural and Multidisciplinary Optimization, 2021                     %
% DOI http://dx.doi.org/10.1007/s00158-021-02859-6                        %
%-------------------------------------------------------------------------%
function [xi] = AdjointProblem(fem,M,C,K,dfdu)
al = fem.alpha; B = fem.beta;  g = fem.gamma; % HHT-alpha parameters
DT = fem.Tmax/fem.NStep;  % Time increment
fDOF = fem.FreeDofs;
nu = zeros(2*fem.NNode,fem.NStep+1); xi = nu; mu = nu;
M0 = (1-al)*(1-g)*DT.*C+(1-al)*(1/2-B)*DT^2.*K;
C0 = C+(1-al)*DT*K;
mu(:,end) = dfdu(:,end);
Fext = -B*DT^2*mu(:,end)-g*DT*nu(:,end);
xi(fDOF(fem.s),end) = fem.L'\(fem.L\Fext(fDOF(fem.s)));
for It=fem.NStep+1:-1:3 % Loop over time steps
  mu(:,It-1) = dfdu(:,It-1)+K*xi(:,It)+mu(:,It);
  nu(:,It-1) = C0*xi(:,It)+DT*mu(:,It)+nu(:,It);
  Fext = -M0*xi(:,It)-DT^2*(B*mu(:,It-1)+(1/2-B)*mu(:,It))-...
         DT*(g*nu(:,It-1)+(1-g)*nu(:,It));
  xi(fDOF(fem.s),It-1) = fem.L'\(fem.L\Fext(fDOF(fem.s)));
end
Fext = -M0*xi(:,2)-DT^2*((1/2-B)*mu(:,2))-DT*(1-g)*nu(:,2);
xi(fDOF,1) = M(fDOF,fDOF)\Fext(fDOF);
%-------------------------------------------------------------------------%
```

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00158-021-02859-6.

Acknowledgements We acknowledge Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. We also thank the support provided by the National Science Foundation (NSF) under grant number 1663244. We are grateful to the insightful comments by Emily D. Sanders and Americo Cunha, which contributed to substantial improvements to the paper. 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.

Funding The authors received support provided by the National Science Foundation (NSF) under grant number 1663244.

Declarations

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.

References

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

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

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

Choi WS, Park GJ (2002) Structural optimization using equivalent static loads at all time intervals. Comput Methods Appl Mech Eng 191(19-20):2105–2122

Dahl J, Jensen JS, Sigmund O (2008) Topology optimization for transient wave propagation problems in one dimension. Struct Multidiscip Optim 36(6):585–595

Dennis JE, Schnabel RB (1996) Numerical methods for unconstrained optimization and nonlinear equations, vol 16. SIAM, Philadelphia

Filipov ET, Chun J, Paulino GH, Song J (2016) Polygonal multiresolution topology optimization (PolyMTOP) for structural dynamics. Struct Multidiscip Optim 53(4):673–694

Giraldo-Londoño O, Mirabella L, Dalloro L, Paulino GH (2020) Multi-material thermomechanical topology optimization with applications to additive manufacturing: Design of main composite part and its support structure. Comput Methods Appl Mech Eng 363:112812

Giraldo-Londoño O, Paulino GH (2020a) Fractional topology optimization of periodic multi-material viscoelastic microstructures

with tailored energy dissipation. Comput Methods Appl Mech Eng 372:113307
Giraldo-Londoño O, Paulino GH (2020b) PolyStress: a Matlab implementation for local stress-constrained topology optimization using the augmented Lagrangian method. Struct Multidiscip Optim 63(4):2065–2097
Giraldo-Londoño O, Paulino GH (2020c) 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. Proc R Soc A 476(2238):20190861
Guest JK (2009) Imposing maximum length scale in topology optimization. Struct Multidiscip Optim 37(5):463–473
Hilber HM, Hughes TJ, Taylor RL (1977) Improved numerical dissipation for time integration algorithms in structural dynamics. Earthq Eng Struct Dyn 5(3):283–292
Hooijkamp EC, van Keulen F (2018) Topology optimization for linear thermo-mechanical transient problems: Modal reduction and adjoint sensitivities. Int J Numer Methods Eng 113(8):1230–1257
Jang HH, Lee HA, Lee JY, Park GJ (2012) Dynamic response topology optimization in the time domain using equivalent static loads. AIAA J 50(1):226–234
Jensen JS, Nakshatrala PB, Tortorelli DA (2014) On the consistency of adjoint sensitivity analysis for structural optimization of linear dynamic problems. Struct Multidiscip Optim 49:831–837
Jiang Y, Ramos AS Jr, Paulino GH (2021) Topology optimization with design-dependent loading: An adaptive sensitivity-separation design variable update scheme. Struct Multidiscip Optim. Accepted
Jog CS (2002) Topology design of structures subjected to periodic loading. J Sound Vib 253(3):687–709
Kang B-S, Park G-J, Arora JS (2005) Optimization of flexible multibody dynamic systems using the equivalent static load method. AIAA J 43(4):846–852
Lee HA, Park GJ (2015) Nonlinear dynamic response topology optimization using the equivalent static loads method. Comput Methods Appl Mech Eng 283:956–970
Liu B, Huang X, Huang C, Sun G, Yan X, Li G (2017) Topological design of structures under dynamic periodic loads. Eng Struct 142:128–136
Liu H, Zhang W, Gao T (2015) A comparative study of dynamic analysis methods for structural topology optimization under harmonic force excitations. Struct Multidiscip Optim 51(6):1321–1333
Ma ZD, Kikuchi N, Cheng HC (1995) Topological design for vibrating structures. Comput Methods Appl Mech Eng 121(1-4):259–280
Ma ZD, Kikuchi N, Hagiwara I (1993) Structural topology and shape optimization for a frequency response problem. Comput Mech 13(3):157–174
Marjugi SM, Leong WJ (2013) Diagonal Hessian approximation for limited memory quasi-Newton via variational principle. J Appl Math 2013:1–8
Martin A, Deierlein GG (2020) Structural topology optimization of tall buildings for dynamic seismic excitation using modal decomposition. Eng Struct 216:110717
Min S, Kikuchi N, Park YC, Kim S, Chang S (1999) Optimal topology design of structures under dynamic loads. Struct Optim 17(2-3):208–218
Newmark NM (1959) A method of computation for structural dynamics. J Eng Mech Div 85(3):67–94
Nocedal J, Wright SJ (2006) Numerical optimization, 2nd edn. Springer, New York
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
Rong JH, Xie YM, Yang XY, Liang QQ (2000) Topology optimization of structures under dynamic response constraints. J Sound Vib 234(2):177–189
Rozvany GI, Zhou M, Birker T (1992) Generalized shape optimization without homogenization. Struct Optim 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, Giraldo-Londoño O, Menezes IFM, Paulino GH (2020) Topology optimization with local stress constraints: A stress aggregation-free approach. Struct Multidiscip Optim 62:1639–1668
Shobeiri V (2019) Bidirectional evolutionary structural optimization for nonlinear structures under dynamic loads. Int J Numer Methods Eng 121(5):888–903
Shu L, Wang MY, Fang Z, Ma Z, Wei P (2011) Level set based structural topology optimization for minimizing frequency response. J Sound Vib 330(24):5820–5834
Stolpe M, Svanberg K (2001) An alternative interpolation scheme for minimum compliance topology optimization. Struct Multidiscip Optim 22(2):116–124
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
Turteltaub S (2005) Optimal non-homogeneous composites for dynamic loading. Struct Multidiscip Optim 30(2):101–112
Verbart A, Stolpe M (2018) A working-set approach for sizing optimization of frame-structures subjected to time-dependent constraints. Struct Multidiscip Optim 58(4):1367–1382
Wang F, Lazarov BS, Sigmund O (2011) On projection methods, convergence and robust formulations in topology optimization. Struct Multidiscip Optim 43(6):767–784
Yoon GH (2010) Structural topology optimization for frequency response problem using model reduction schemes. Comput Methods Appl Mech Eng 199(25-28):1744–1763
Zhang X, Paulino GH, Ramos Jr AS (2018) Multi-material topology optimization with multiple volume constraints: A general approach applied to ground structures with material nonlinearity. Struct Multidiscip Optim 57(1):161–182
Zhao J, Wang C (2016) Dynamic response topology optimization in the time domain using model reduction method. Struct Multidiscip Optim 53(1):101–114
Zhao J, Wang C (2017) Topology optimization for minimizing the maximum dynamic response in the time domain using aggregation functional method. Comput Struct 190:41–60
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