Noname manuscript No.
(will be inserted by the editor)

Topology Optimization with Variable Loads and Supports Using a Super-Gaussian Projection Function

Lee Alacoque · Kai A. James

Received: date / Accepted: date

Abstract This work presents a new method for efficiently designing loads and supports simultaneously with material distribution in density-based topology optimization. We use a higher-order or super-Gaussian function to parameterize the shapes, locations, and orientations of mechanical loads and supports. With a distance function as an input, the super-Gaussian function projects smooth geometric shapes which can be used to model various types of boundary conditions using minimal numbers of additional design variables. As examples, we use the proposed formulation to model both concentrated and distributed loads and supports. We also model movable non-design regions of predetermined solid shapes using the same distance functions and design variables as the variable boundary conditions. Computing the design sensitivities using the adjoint sensitivity analysis method, we implement the technique in a 2D topology optimization algorithm with linear elasticity and demonstrate the improvements that the super-Gaussian projection method makes to some common benchmark problems. By allowing the optimizer to move the loads and supports throughout the design domain, the method produces significant enhancements to structures such as compliant mechanisms where the locations of the input load and fixed supports have a large effect on the magnitude of the output displacements.

Lee Alacoque
Department of Aerospace Engineering, University of Illinois at Urbana-Champaign
E-mail: leea2@illinois.edu

Kai A. James
Department of Aerospace Engineering, University of Illinois at Urbana-Champaign
E-mail: kaijames@illinois.edu

Keywords Topology Optimization · Compliant Mechanisms · Design of Loads · Design of Supports · Boundary Condition Optimization · Geometry Projection

1 Introduction

Topology optimization is a computational method used to automatically generate optimal layouts of material within a design domain (Bendsoe and Sigmund 2013), with common design problems including minimization of compliance for maximally stiff structures (Andreassen et al. 2011) or maximization of an output displacement for compliant mechanism design (Sigmund 1997; B. Zhu, X. Zhang, et al. 2020). It is currently standard practice for the boundary conditions in these problems to be predetermined by the user based only on intuition, where the locations of applied loads and rigid supports remain fixed and unchanging during the optimization process. In topology optimization studies such as multiple degree-of-freedom compliant mechanisms (Frecker, Kikuchi, and Kota 1999; Alonso, Ansola, and Querin 2014; B. Zhu, Chen, et al. 2018; Sigmund 2001), several input loads are applied which can have complex interactions between each other and desired output displacements. This makes it conceivable that the configuration of the boundary conditions could have a large effect on the final topology and performance of the optimized designs. In another application of topology optimization for the design of a bi-stable airfoil compliant mechanism (Bhattacharyya, Conlan-Smith, and James 2019), the input load and fixed supports were manually placed at particular locations on the domain which the authors noted required specific user inputs to determine. In examples such as these it is unlikely that the loads

and supports have been placed in optimal positions and orientations, suggesting that a more systematic way of determining their configuration would be useful. In this paper we therefore seek to develop an efficient method of including the boundary conditions as automatically optimized parameters in the topology optimization formulation.

Optimizing the locations of mechanical (fixed-displacement) supports is not a new idea and has been explored in other studies. An early work by Thomas Buhl (Buhl 2002) used an additional set of design variables to control the stiffness of support springs connected to each element in the finite element mesh of the structure. With its many design variables representing the supported areas, the method allowed for arbitrary support configurations to appear, but also required a penalization method to avoid supports with intermediate stiffness and extra constraint functions for the total area of the domain being supported. In a similar but more complex method by Zhu and Zhang (J. Zhu and W. Zhang 2010), supports were modeled using separate movable components subjected to boundary conditions, constraints to prevent component overlap, and a finite element mesh that adapted to the updating component locations. For the design of multi-component systems or assemblies, similar techniques to the mechanical support design methods have been applied. Rakotondrainibe et al. (Rakotondrainibe, Allaire, and Orval 2020) optimized rigid and bolted connections using a type of Robin boundary condition to model the locations, while using the topological derivative to allow for introduction of new connections during the optimization. Ambrozkiewicz and Kriegesmann (Ambrozkiewicz and Kriegesmann 2021) simultaneously optimized the joint locations and topologies of mechanical assemblies using spring connections to transfer the loads between parts. In another study for the design of multi-body mechanisms, Swartz and James (Swartz and James 2019) used spring connections between components which were parameterized by a Gaussian function to model the behavior of pin joints and to optimize their locations.

For applied loads that change during the course of the optimization, studies such as those by Lee at al. (Lee and Martins 2012; Lee, James, and Martins 2012) have implemented design-dependent pressure loads or self-weights, where the nodal forces were computed by using detected solid-void material interfaces for pressure or by using the weights of each element based on their density value. Other problems such as homogenization-based microstructure design and thermal structure design (Alacoque, Watkins, and Tamijani 2021) also feature design-dependent loads, but like with pressure and self-weight, these loads are calculated based on the current design and are not controlled by a design variable to find an optimal way of applying them.

In certain scenarios, such as for distributed loads, we will also need to apply continuously movable non-design regions to ensure that the varying loads are applied to completely solid surfaces, while for supports we may wish to have a predetermined solid shape for manufacturability reasons. In the paper by Ambrozkiewicz and Kriegesmann (Ambrozkiewicz and Kriegesmann 2021), they enforced circular and cylindrical non-design regions at the movable connections between components using parametric equations and "mask" vectors, which were combined with the structural topology. With a somewhat similar concept, Pollini and Amir (Pollini and Amir 2020) projected shapes onto the design domain from linear segmented profiles using super-Gaussian functions. They used the projections to control material properties or local constraints in specific parts of the domain that were movable by the optimizer. This method of geometry projection onto a domain discretized by a fixed mesh was first introduced by Norato et al. (J. Norato, Bell, and Tortorelli 2015), who defined the geometry of bars using distance functions and used design variables for the spatial locations of their endpoints. Since then, the geometry projection method has been used for other problems such as multi-material lattice structures (Kazemi, Vaziri, and Julián A Norato 2020) and for problems involving projection of more complex discrete shapes (S. Zhang, Julián A Norato, et al. 2016; S. Zhang, Gain, and Julián A Norato 2018; Jessee et al. 2020).

In this paper we take inspiration from and combine the concepts of spring connections from Buhl (Buhl 2002), distance functions from Norato et al. (J. Norato, Bell, and Tortorelli 2015), and projection using Gaussian functions from Swartz and James (Swartz and James 2019) and Pollini and Amir (Pollini and Amir 2020) to parameterize and optimize both loads and supports simultaneously with the structural topology. The Gaussian function method we develop using these concepts offers several attractive features: 1) Efficient modeling of variable boundary conditions which adds no additional degrees of freedom and requires no remeshing, 2) geometry projection characteristics that can be easily adjusted by choosing the values of a few scalar parameters in the Gaussian function, 3) projection shapes and topologies that can be changed in a modular way by using different distance functions as the input to the same Gaussian function, 4) overlapping of multiple loads or supports projected by a single Gaussian function which cause no issues because they merge together seamlessly without requiring any additional formulations, and 5) formulations and sensitivities which are simple to de-

rive and to implement in existing topology optimization codes.

We begin the paper with a description of the standard Solid Isotropic Material with Penalization (SIMP) topology optimization method in section 2. In section 3, we extend the finite element model by augmenting it with spring connections and nodal forces to allow for variable load and support boundary conditions. The general higher-order Gaussian function is introduced in section 4, and two different distance functions used for projecting point and line geometries into the domain are given. Sections 5 and 6 describe the specific Gaussian functions and formulations used to model the variable boundary conditions and the movable non-design regions, respectively. The main objective and constraint functions used for the topology optimization example problems are given in section 7 along with their adjoint sensitivity analysis formulations. Numerical examples showing the efficacy of the Gaussian function projection method are shown in section 8, and conclusions and potential for future applications of the method are discussed in section 9.

2 Topology Optimization

We base our method on standard SIMP (Solid Isotropic Material with Penalization) topology optimization, in which the linear elasticity problem is discretized into a uniform grid of rectangular 4-node bilinear finite elements. Each finite element e is assigned a design variable ρ_e which represents the density or volume fraction of material in the element ranging from $\rho_e = 0$ (void) to $\rho_e = 1$ (solid). To avoid checkerboard patterns and enforce a minimum length scale on the optimized designs, we use the density filtering method (Bruns and Tortorelli 2001). The filtered densities, representing the actual physical design to be manufactured, are given by

$$\bar{\rho}_e = \frac{\sum_{i=1}^{N_e} H_{ei}\rho_i}{\sum_{i=1}^{N_e} H_{ei}}, \tag{1}$$

$$H_{ei} = \max(0, r_{min} - \Delta(e,i)), \tag{2}$$

where N_e is the number of elements in the mesh, r_{min} is the filter radius specified by the user, and $\Delta(e,i)$ is the distance between element e and element i. The physical densities $\bar{\rho}_e$ are then used to calculate the Young's modulus of each element using the SIMP interpolation model:

$$E_e = E_{min} + \bar{\rho}_e^p(E_0 - E_{min}), \tag{3}$$

where E_{min} is a minimum value of Young's modulus given to void elements to avoid singular stiffness matrices in the finite element analyses, p is the SIMP penalization factor used to avoid intermediate densities by setting it to a value greater than 1, and E_0 is the Young's modulus of the solid material. The Poisson's ratio, ν, is a constant.

3 Finite Element Formulation

Expanding on the standard SIMP method to allow for simultaneous optimization of the structural topology, mechanical supports, and applied forces, the finite element mesh is connected to a system of spring elements. Like in the method introduced by Buhl (Buhl 2002), all degrees of freedom in the continuum part of the mesh are connected to rigid fixtures by a spring. The stiffness of the support springs are based on a value k_e^s assigned to each element e, making the horizontal and vertical degrees of freedom of each node equally stiff. Extending Buhl's method, we also apply forces to every node of the continuum mesh at an angle θ with a magnitude based on a value f_e associated with each of the continuum elements. For the design of compliant mechanisms using the spring model, springs opposing the input forces are also placed at every node at the same angle θ and with element stiffness values k_e^{in}. The general mesh setup is illustrated in Figure 1. In the following sections of this paper, we explain how to vary the distributions of the spring stiffnesses and force magnitudes in order to control the effective shapes and locations of the boundary conditions.

The finite element equations for the mesh setup are assembled as follows, where the symbol Λ is used to denote the finite element assembly operator. The global stiffness matrix $\boldsymbol{K}$ of the continuum mesh is assembled in the usual way as

$$\boldsymbol{K} = \bigwedge_{e=1}^{N_e} E_e \int_{V_e} \boldsymbol{B}_e^T \boldsymbol{C}_0 \boldsymbol{B}_e dV_e = \bigwedge_{e=1}^{N_e} E_e \boldsymbol{k}_e^0, \tag{4}$$

where e is the element number, V_e is the element volume, $\boldsymbol{B}_e$ is the element strain-displacement matrix, and $\boldsymbol{C}_0$ is the constitutive matrix for a unit Young's modulus. The integral in equation (4) is the element stiffness matrix for a unit Young's modulus and is written more simply as $\boldsymbol{k}_e^0$. The support and input load springs add no additional degrees of freedom to the model, so their contributions to the global stiffness matrix can be simply added to $\boldsymbol{K}$. The contribution of the support springs, $\boldsymbol{K}^s$, is the assembly of spring element stiffness matrices

$$\boldsymbol{K}^s = \bigwedge_{e=1}^{N_e} k_e^s \boldsymbol{I}_8, \tag{5}$$

where $\boldsymbol{I}_8$ is the 8x8 (8 degrees of freedom per element) identity matrix. For the global force vector, the applied

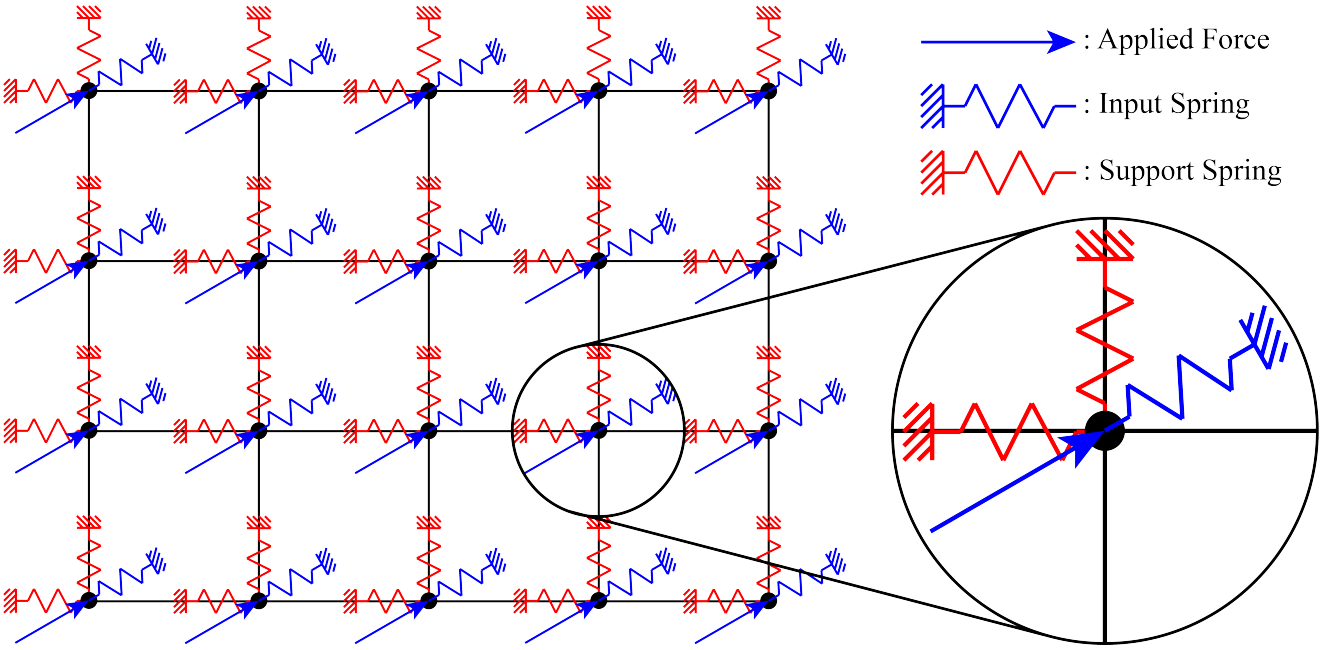


Fig. 1: The general finite element mesh setup with forces, input force springs, and support springs applied to each node

forces at each element are assembled as

$$\boldsymbol{F} = \bigwedge_{e=1}^{N_e} f_e \boldsymbol{\Theta}, \tag{6}$$

where $\boldsymbol{\Theta}$ is an 8x1 rotation vector that converts the nodal force magnitudes into their corresponding components in the x and y directions based on the global reference frame:

$$\boldsymbol{\Theta} = \begin{bmatrix} \cos\theta & \sin\theta & \cdots \end{bmatrix}^T. \tag{7}$$

The contribution of the input springs to the global stiffness matrix is assembled in a similar way to the force vector:

$$\boldsymbol{K}^{in} = \bigwedge_{e=1}^{N_e} k_e^{in} \boldsymbol{I}_8 \langle \boldsymbol{\Theta} \rangle. \tag{8}$$

Here, the vector $\langle \boldsymbol{\Theta} \rangle$ is a smooth approximation of the absolute values of $\boldsymbol{\Theta}$ in order to avoid negative values of spring stiffness while keeping the function differentiable at all values of θ:

$$\langle \boldsymbol{\Theta} \rangle = \begin{bmatrix} \sqrt{\cos^2\theta + \epsilon} & \sqrt{\sin^2\theta + \epsilon} & \cdots \end{bmatrix}^T, \tag{9}$$

where ϵ is a small positive number. The output spring for compliant mechanism problems is modeled by adding a single spring of stiffness k^{out} to the output degree of freedom on the main diagonal of the total global stiffness matrix. The global vector of nodal displacements $\boldsymbol{U}$ is then found by solving the finite element equilibrium equations, where the spring stiffness matrices are added to the continuum stiffness matrix:

$$\left(\boldsymbol{K} + \boldsymbol{K}^s + \boldsymbol{K}^{in}\right)\boldsymbol{U} = \boldsymbol{F}. \tag{10}$$

4 Gaussian Function Parameterization

To control the distributions of force magnitude and support stiffness using only a small number of parameters, we use a super-Gaussian function of the form

$$G(x,y) = Ab^{-\left(\frac{d(x,y)^2}{r^2}\right)^P}. \tag{11}$$

The super-Gaussian function has a flat plateau-shaped top with a smooth Gaussian fall-off in the directions of increasing distance represented by the function $d(x,y)$. The parameter P controls the sharpness of the plateau, the coefficient A is the height, and the radius parameter r determines the length from the center of the plateau to a point in the fall-off region with height A divided by the base parameter b:

$$G(d=r) = \frac{A}{b}. \tag{12}$$

The properties of the super-Gaussian function are illustrated in Figure 2, where we use a simple one-dimensional distance function describing the distance to the origin point.

In two dimensions, we have made use of two different types of distance functions. The first is used to model concentrated loads and small circular supports and is the minimum distance to a number of zero dimensional points, where the positions of the N points are described by the sets of design variables $\boldsymbol{x}_D$ and $\boldsymbol{y}_D$, where $\boldsymbol{x}_D = \begin{bmatrix} x_D^{(1)} & x_D^{(2)} & \cdots & x_D^{(N)} \end{bmatrix}$ and $\boldsymbol{y}_D = \begin{bmatrix} y_D^{(1)} & y_D^{(2)} & \cdots & y_D^{(N)} \end{bmatrix}$. The

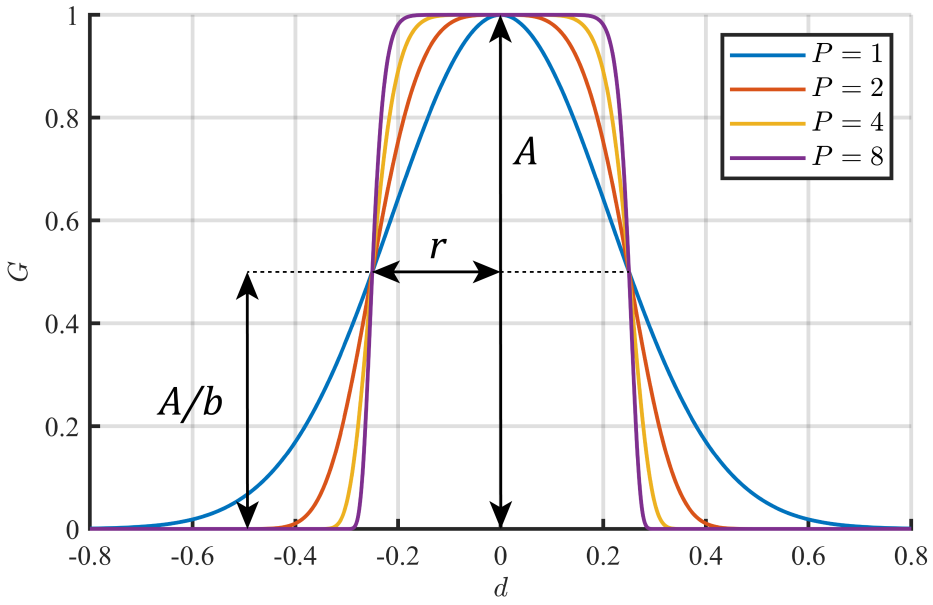
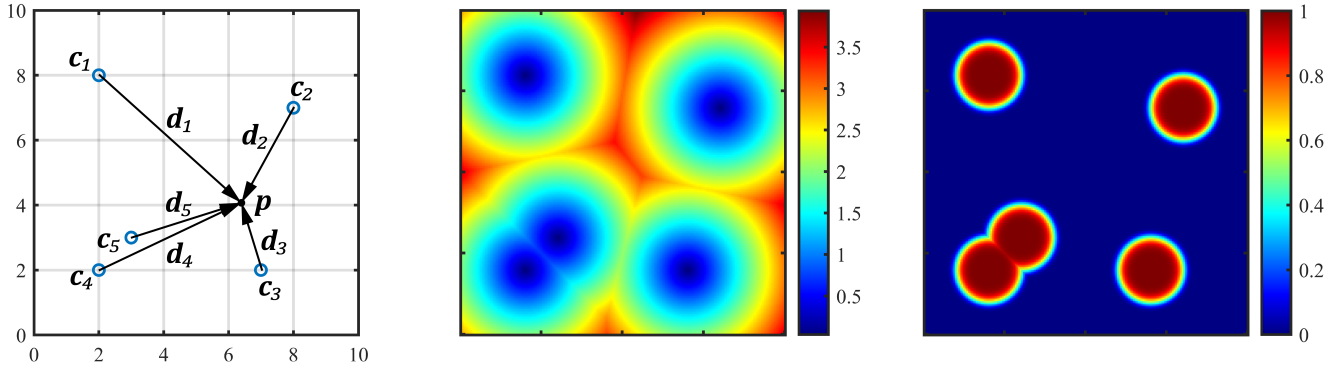


Fig. 2: Super-Gaussian function with a 1D distance function for several values of the exponent P



(a) Points $\boldsymbol{c}_i$ and their positions relative to a point in the domain $\boldsymbol{p}_e$

(b) Minimum distance function to points $\boldsymbol{c}_i$

(c) Super-Gaussian function with $A = 1$, $b = 2$, $r = 1$, and $P = 4$

Fig. 3: Using the Gaussian function to project circular shapes from zero dimensional points

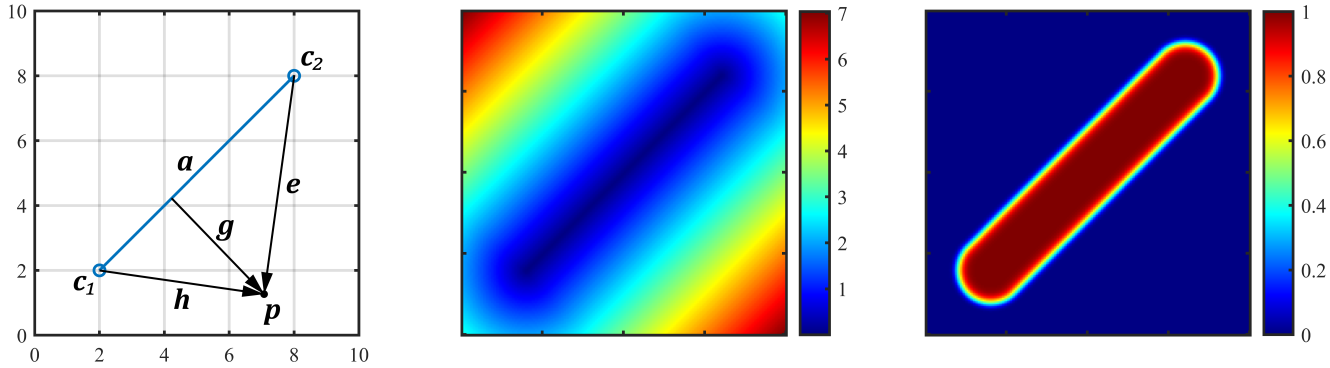
subscript D indicates the coordinate is a design variable and not just a spatial point in the domain. The minimum distance from an element centroid coordinate $\boldsymbol{p}_e = \begin{bmatrix} x^{(e)} & y^{(e)} \end{bmatrix}$ to the N points $\boldsymbol{c}_i = \begin{bmatrix} x^{(i)} & y^{(i)} \end{bmatrix}$ is

$$d_e = \min(\|\boldsymbol{d}_1\|, \|\boldsymbol{d}_2\|, \cdots, \|\boldsymbol{d}_N\|), \tag{13}$$

where $\boldsymbol{d}_i = \boldsymbol{p}_e - \boldsymbol{c}_i$. An example of the minimum distance function to five different points and the resulting circular shapes with radius r projected into the domain by using the Gaussian function is shown in Figure 3. Notice that as the circular regions overlap, the value of the Gaussian function is always as if we took the maximum value of separate Gaussian functions applied to each point individually. Although this produces sharp transitions, the function is still fully differentiable with respect to each individual design variable and no numerical issues are caused in the optimization. If multiple points occupy identical coordinates, the Gaussian function produces a distribution exactly as though a single point were at that position.

The second distance function we use is to model distributed loads and supports. This function is the minimum distance to a one-dimensional line, defined by two endpoints $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ as shown in Figure 4 and described by the piecewise function (J. Norato, Bell, and Tortorelli 2015)

$$d_e = \begin{cases} \|\boldsymbol{h}\| & \text{if } \boldsymbol{a} \cdot \boldsymbol{h} \leq 0, \\ \|\boldsymbol{g}\| & \text{if } 0 < \boldsymbol{a} \cdot \boldsymbol{h} < \boldsymbol{a} \cdot \boldsymbol{a}, \\ \|\boldsymbol{e}\| & \text{if } \boldsymbol{a} \cdot \boldsymbol{h} \geq \boldsymbol{a} \cdot \boldsymbol{a}, \end{cases} \tag{14}$$



(a) Geometry of a line $\boldsymbol{a}$ defined by two points $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ with positions relative to a point $\boldsymbol{p}$ in the domain

(b) Minimum distance function to line $\boldsymbol{a}$

(c) Super-Gaussian function with $A = 1$, $b = 2$, $r = 1$, and $P = 4$

Fig. 4: Using the Gaussian function to project a bar shape from a one dimensional line

where

$$\boldsymbol{a} = \boldsymbol{c}_2 - \boldsymbol{c}_1, \tag{15}$$

$$\boldsymbol{h} = \boldsymbol{p}_e - \boldsymbol{c}_1, \tag{16}$$

$$\boldsymbol{e} = \boldsymbol{p}_e - \boldsymbol{c}_2, \tag{17}$$

$$\boldsymbol{g} = \left[\boldsymbol{I} - \frac{1}{\|\boldsymbol{a}\|^2}\boldsymbol{a} \otimes \boldsymbol{a}\right] \boldsymbol{h}. \tag{18}$$

After passing the line distance function through the Gaussian function, a rounded bar shape of radius r is projected onto the domain.

5 Variable Loads and Supports

The locations of the supports are allowed to vary by assigning point coordinates as design variables for the optimizer. We define the vector of all design variables, $\boldsymbol{z}$, by concatenating the vector of element densities $\boldsymbol{\rho}$ with the vectors of support point coordinates $\boldsymbol{x}_s$ and $\boldsymbol{y}_s$:

$$\boldsymbol{z} = \left[\boldsymbol{\rho}\ \boldsymbol{x}_s\ \boldsymbol{y}_s\right]. \tag{19}$$

The support points are then used to define a distance function using either of equations (13) or (14). Using the Gaussian function to define the projection based on the distance function, we obtain the distribution of support spring stiffness throughout the design domain. In the Gaussian function (equation 11), the coefficient A is assigned a value of spring stiffness, k_0, chosen by the user which should be large enough to adequately simulate rigid supports but low enough to avoid numerical problems:

$$k_e^s = k_0 b^{-\left(\frac{d_e^2}{r^2}\right)^P}. \tag{20}$$

The location and orientation of loads are optimized by including the coordinates $\boldsymbol{x}_f$ and $\boldsymbol{y}_f$ and the orientation of the forces θ in the design variable vector. Appending these parameters to the vector of design variables gives

$$\boldsymbol{z} = \left[\boldsymbol{\rho}\ \boldsymbol{x}_s\ \boldsymbol{y}_s\ \boldsymbol{x}_f\ \boldsymbol{y}_f\ \theta\right]. \tag{21}$$

The design variables for the forces are used to define another distance function, which is passed to the Gaussian function to obtain a force value in each element:

$$f_e = A_f b^{-\left(\frac{d_e^2}{r^2}\right)^P}. \tag{22}$$

We define the coefficient A_f such that the user can specify the approximate total load applied in the design domain as a single constant f_0. As the Gaussian function superscript P approaches infinity, the total load in the domain becomes equal to the total load under the projected shape of radius r. For concentrated loads, this is written as

$$A_f = \frac{f_0 V_e}{n_n \pi r^2}, \tag{23}$$

while for distributed loads it is written as a function of the line length $\|\boldsymbol{a}\|$, given by

$$A_f = \frac{f_0 V_e}{n_n(\pi r^2 + 2r\|\boldsymbol{a}\|)}. \tag{24}$$

where n_n is the number of overlapping nodes between elements (four in the case of 2D rectangular bilinear elements). For the design of compliant mechanisms using the spring model, we scale the stiffness of the input springs proportionally to the force magnitudes using a user-specified constant k_0^{in}:

$$k_e^{in} = \frac{k_0^{in}}{A_f} f_e. \tag{25}$$

6 Variable Non-Design Regions

To create movable non-design regions that follow the locations of the variable boundary conditions, we use the Gaussian function to project a density distribution onto the domain which is then combined with the filtered densities in a smooth and differentiable way. The filtered element densities are calculated from the density design variables in the same way as before:

$$\tilde{\rho}_e = \frac{\sum_{i=1}^{N_e} H_{ei}\rho_i}{\sum_{i=1}^{N_e} H_{ei}}. \tag{26}$$

The projected density distribution is given by the Gaussian function of unit height ($A = 1$) with a distance function based on load and support point variables:

$$\hat{\rho}_e = b^{-\left(\frac{\mathrm{d}_e^2}{r^2}\right)^P}. \tag{27}$$

The filtered densities $\tilde{\rho}_e$ and the projected densities $\hat{\rho}_e$ are then combined into a physical density field using a generalized mean to take the maximum of the two fields:

$$\bar{\rho}_e = \left(\frac{\tilde{\rho}_e^Q + \hat{\rho}_e^Q}{2}\right)^{\frac{1}{Q}}. \tag{28}$$

With $Q = 1$, $\bar{\rho}_e$ is the average of $\tilde{\rho}_e$ and $\hat{\rho}_e$, and as Q approaches infinity, $\bar{\rho}_e$ approaches the maximum of the two values from below. Thus, we can set Q to a finite value to approximate the maximum of the filtered and projected density fields in a smooth and differentiable way.

7 Optimization Problems and Sensitivity Analysis

The optimization problem considered in this paper is a minimization of an objective function f_{obj} subjected to constraints on the design variable upper and lower limits (denoted by the subscripts U and L, respectively, with n_s representing the number of *support* design variables and n_f representing the number of *load* variables), the amount of material in the domain or overall volume fraction V_f, and any additional number, n_c, of functions $h_i(\boldsymbol{z})$ that may be desired in particular problem setups:

$$\begin{aligned}
\min_{\boldsymbol{z}} \quad & f_{obj}(\boldsymbol{z}) \\
\text{s.t.} \quad & 0 \le \rho_e \le 1, && e = 1,\dots,N_e, \\
& x_{sL}^{(i)} \le x_s^{(i)} \le x_{sU}^{(i)}, && i = 1,\dots,n_s, \\
& y_{sL}^{(i)} \le y_s^{(i)} \le y_{sU}^{(i)}, && i = 1,\dots,n_s, \\
& x_{fL}^{(i)} \le x_f^{(i)} \le x_{fU}^{(i)}, && i = 1,\dots,n_f, \\
& y_{fL}^{(i)} \le y_f^{(i)} \le y_{fU}^{(i)}, && i = 1,\dots,n_f, \\
& \theta_L \le \theta \le \theta_U, \\
& \frac{\sum_{e=1}^{N_e} \bar{\rho}_e}{N_e} \le V_f, \\
& h_i(\boldsymbol{z}) \le 0, && i = 1,\dots,n_c.
\end{aligned} \tag{29}$$

This problem is solved using the method of moving asymptotes (MMA) (Svanberg 1987), which as a gradient-based numerical optimization method requires the objective and constraint function values along with their first derivatives as inputs. For compliance minimization problems, the objective function $f_{obj}(\boldsymbol{z})$ takes the form

$$f_{obj}(\boldsymbol{z}) = C(\boldsymbol{z}) = \boldsymbol{F}(\boldsymbol{z})^T \boldsymbol{U}(\boldsymbol{z}). \tag{30}$$

In the case of compliant mechanism design, the objective is the specified output displacement

$$f_{obj}(\boldsymbol{z}) = U_{out}(\boldsymbol{z}) = \boldsymbol{L}^T \boldsymbol{U}(\boldsymbol{z}), \tag{31}$$

where $\boldsymbol{L}^T$ is a constant vector of all zeros except at the output degree of freedom, where it has a value of one.

To determine the derivative of the compliance with respect to each of the design variables, we use adjoint sensitivity analysis to find:

$$\frac{\partial C(\boldsymbol{z})}{\partial z_i} = \frac{\partial \boldsymbol{F}^T}{\partial z_i}\boldsymbol{U} + \boldsymbol{U}^T \cdot \left(\frac{\partial \boldsymbol{F}}{\partial z_i} - \left(\frac{\partial \boldsymbol{K}}{\partial z_i} + \frac{\partial \boldsymbol{K}^s}{\partial z_i} + \frac{\partial \boldsymbol{K}^{in}}{\partial z_i}\right)\boldsymbol{U}\right). \tag{32}$$

Similarly for the output displacement objective function, we find the following form:

$$\frac{\partial U_{out}(\boldsymbol{z})}{\partial z_i} = \boldsymbol{\lambda}^T \cdot \left(\frac{\partial \boldsymbol{F}}{\partial z_i} - \left(\frac{\partial \boldsymbol{K}}{\partial z_i} + \frac{\partial \boldsymbol{K}^s}{\partial z_i} + \frac{\partial \boldsymbol{K}^{in}}{\partial z_i}\right)\boldsymbol{U}\right). \tag{33}$$

where the adjoint vector is first computed as:

$$\boldsymbol{\lambda} = \left(\boldsymbol{K} + \boldsymbol{K}^s + \boldsymbol{K}^{in}\right)^{-1} \boldsymbol{L}. \tag{34}$$

The adjoint sensitivity equations then take on different forms for each design variable type $\boldsymbol{\rho}$, $\boldsymbol{x}_s$ or $\boldsymbol{y}_s$, $\boldsymbol{x}_f$ or $\boldsymbol{y}_f$, and θ. For the sensitivity with respect to the

Table 1: Adjoint sensitivity equations for the derivative of the compliance and output displacement functions with respect to each design variable

z_i	$\frac{\partial C(\boldsymbol{z})}{\partial z_i}$	$\frac{\partial \boldsymbol{U}_{out}(\boldsymbol{z})}{\partial z_i}$
$\bar{\rho}_i$	$-\boldsymbol{U}_e^T \frac{\partial \boldsymbol{K}_e}{\partial \bar{\rho}_e} \boldsymbol{U}_e$	$-\boldsymbol{\lambda}_e^T \frac{\partial \boldsymbol{K}_e}{\partial \bar{\rho}_e} \boldsymbol{U}_e$
$z_s^{(i)}$	$-\boldsymbol{U}^T (\frac{\partial \boldsymbol{K}}{\partial z_s^{(i)}} + \frac{\partial \boldsymbol{K}^s}{\partial z_s^{(i)}}) \boldsymbol{U}$	$-\boldsymbol{\lambda}^T (\frac{\partial \boldsymbol{K}}{\partial z_s^{(i)}} + \frac{\partial \boldsymbol{K}^s}{\partial z_s^{(i)}}) \boldsymbol{U}$
$z_f^{(i)}$	$\frac{\partial \boldsymbol{F}^T}{\partial z_f^{(i)}} \boldsymbol{U} + \boldsymbol{U}^T \left(\frac{\partial \boldsymbol{F}}{\partial z_f^{(i)}} - \left(\frac{\partial \boldsymbol{K}}{\partial z_f^{(i)}} + \frac{\partial \boldsymbol{K}^{in}}{\partial z_f^{(i)}} \right) \boldsymbol{U} \right)$	$\boldsymbol{\lambda}^T \left(\frac{\partial \boldsymbol{F}}{\partial z_f^{(i)}} - \left(\frac{\partial \boldsymbol{K}}{\partial z_f^{(i)}} + \frac{\partial \boldsymbol{K}^{in}}{\partial z_f^{(i)}} \right) \boldsymbol{U} \right)$
θ	$\frac{\partial \boldsymbol{F}^T}{\partial \theta} \boldsymbol{U} + \boldsymbol{U}^T \left(\frac{\partial \boldsymbol{F}}{\partial \theta} - \frac{\partial \boldsymbol{K}^{in}}{\partial \theta} \boldsymbol{U} \right)$	$\boldsymbol{\lambda}^T \left(\frac{\partial \boldsymbol{F}}{\partial \theta} - \frac{\partial \boldsymbol{K}^{in}}{\partial \theta} \boldsymbol{U} \right)$

density variable at each element e ($z_i = \rho_i$), the derivative with respect to the physical density $\bar{\rho}_e$ is taken first and the chain rule is subsequently used to find the sensitivity with respect to the base design variable:

$$\frac{\partial f_{obj}(\boldsymbol{z})}{\partial \rho_i} = \sum_{e=1}^{N_e} \frac{\partial f_{obj}(\boldsymbol{z})}{\partial \bar{\rho}_e} \frac{\partial \bar{\rho}_e}{\partial \tilde{\rho}_e} \frac{\partial \tilde{\rho}_e}{\partial \rho_i}, \tag{35}$$

where:

$$\frac{\partial \bar{\rho}_e}{\partial \tilde{\rho}_e} = \frac{1}{2} \left(\frac{\tilde{\rho}_e^Q + \hat{\rho}_e^Q}{2} \right)^{\frac{1}{Q}-1} \tilde{\rho}_e^{Q-1}, \tag{36}$$

$$\frac{\partial \tilde{\rho}_e}{\partial \rho_i} = \frac{H_{ie}}{\sum_{j=1}^{N_e} H_{ij}}. \tag{37}$$

The final adjoint sensitivity equations are summarized in Table 1.

We now list the partial derivatives of the stiffness matrices and force vectors. The sensitivity of the global stiffness matrix with respect to the physical densities is given by

$$\frac{\partial \boldsymbol{K}_e}{\partial \bar{\rho}_e} = p \bar{\rho}_e^{p-1} (E_0 - E_{min}) \boldsymbol{k}_e^0. \tag{38}$$

where the subscript e on the global matrix indicates that we only consider entries within the global stiffness matrix that correspond to degrees of freedom associated with the element e, since the derivatives of all other entries vanish.

For the derivative of the stiffness matrix of the continuum structure with respect to each support variable $z_i = x_s^{(i)}$ or $z_i = y_s^{(i)}$ (and similarly with respect to the force location variables), the derivative is:

$$\frac{\partial \boldsymbol{K}}{\partial z_s^{(i)}} = \bigwedge_{e=1}^{N_e} p \bar{\rho}_e^{p-1} \frac{\partial \bar{\rho}_e}{\partial z_s^{(i)}} (E_0 - E_{min}) \boldsymbol{k}_e^0, \tag{39}$$

where

$$\frac{\partial \bar{\rho}_e}{\partial z_s^{(i)}} = \frac{1}{2} \left(\frac{\tilde{\rho}_e^Q + \hat{\rho}_e^Q}{2} \right)^{\frac{1}{Q}-1} \tilde{\rho}_e^{Q-1} \frac{\partial \hat{\rho}_e}{\partial z_s^{(i)}}, \tag{40}$$

and

$$\frac{\partial \hat{\rho}_e}{\partial z_s^{(i)}} = -\frac{2P}{r^2} b^{-\left(\frac{d_e^2}{r^2}\right)^P} \ln(b) \left(\frac{d_e^2}{r^2} \right)^{P-1} d_e \frac{\partial d_e}{\partial z_s^{(i)}}. \tag{41}$$

For the support spring component of the global stiffness matrix, it is:

$$\frac{\partial \boldsymbol{K}^s}{\partial z_s^{(i)}} = \bigwedge_{e=1}^{N_e} \frac{\partial k_e^s}{\partial z_s^{(i)}} \boldsymbol{I}_8, \tag{42}$$

where

$$\frac{\partial k_e^s}{\partial z_s^{(i)}} = - \frac{2k_0 P}{r^2} b^{-\left(\frac{d_e^2}{r^2}\right)^P} \ln(b) \left(\frac{d_e^2}{r^2} \right)^{P-1} d_e \frac{\partial d_e}{\partial z_s^{(i)}}. \tag{43}$$

For the derivative with respect to each force variable $z_i = x_f^{(i)}$ or $z_i = y_f^{(i)}$, The derivative of the input spring stiffness matrix is:

$$\frac{\partial \boldsymbol{K}^{in}}{\partial z_f^{(i)}} = -\frac{k_0^{in}}{A_f^2} \frac{\partial A_f}{\partial z_f^{(i)}} \bigwedge_{e=1}^{N_e} f_e \boldsymbol{I}_8 \langle \boldsymbol{\Theta} \rangle + \frac{k_0^{in}}{A_f} \bigwedge_{e=1}^{N_e} \frac{\partial f_e}{\partial z_f^{(i)}} \boldsymbol{I}_8 \langle \boldsymbol{\Theta} \rangle. \tag{44}$$

The force vector is also a function of the force design variables and its derivative is given by:

$$\frac{\partial \boldsymbol{F}}{\partial z_f^{(i)}} = \bigwedge_{e=1}^{N_e} \frac{\partial f_e}{\partial z_f^{(i)}} \boldsymbol{\Theta}, \tag{45}$$

where, for the case where the coefficient A_f is a function of the line length:

$$\frac{\partial f_e}{\partial z_f^{(i)}} = \frac{\partial A_f}{\partial z_f^{(i)}} b^{-\left(\frac{d_e^2}{r^2}\right)^P} - \frac{2A_f P}{r^2} b^{-\left(\frac{d_e^2}{r^2}\right)^P} \ln(b) \left(\frac{d_e^2}{r^2} \right)^{P-1} d_e \frac{\partial d_e}{\partial z_f^{(i)}}. \tag{46}$$

The derivative of the coefficient is given by

$$\frac{\partial A_f}{\partial z_f^{(i)}} = \frac{2 f_0 V_e r}{n_n(\pi r^2 + 2r\|\boldsymbol{a}\|)^2} \frac{\boldsymbol{a}^{(i)}}{\|\boldsymbol{a}\|}, \tag{47}$$

for the case of a line load or else is equal to zero for the case of a point load.

For the derivative of the global stiffness matrix with respect to the force angle variable θ, the input spring stiffness matrix has dependence:

$$\frac{\partial \boldsymbol{K}^{in}}{\partial \theta} = \frac{k_0^{in}}{A_f} \overset{N_e}{\underset{e=1}{\Lambda}} f_e \boldsymbol{I}_8 \frac{\partial \langle \boldsymbol{\Theta} \rangle}{\partial \theta}, \tag{48}$$

where

$$\frac{\partial \langle \boldsymbol{\Theta} \rangle}{\partial \theta} = \left[-\frac{\sin\theta\cos\theta}{\sqrt{\cos^2(\theta)+\epsilon}} \;\; \frac{\sin\theta\cos\theta}{\sqrt{\sin^2(\theta)+\epsilon}} \;\; \cdots \right]^T. \tag{49}$$

Finally, the derivative of the global force vector with respect to the load orientation is given by:

$$\frac{\partial \boldsymbol{F}}{\partial \theta} = \overset{N_e}{\underset{e=1}{\Lambda}} f_e \frac{\partial \boldsymbol{\Theta}}{\partial \theta}, \tag{50}$$

where

$$\frac{\partial \boldsymbol{\Theta}}{\partial \theta} = \left[-\sin\theta \;\; \cos\theta \;\; \cdots \right]^T. \tag{51}$$

When zero dimensional points are being used to construct the minimum distance function, the derivative of the distance function is:

$$\frac{\partial d_e}{\partial \begin{bmatrix} x_D^{(i)} & y_D^{(i)} \end{bmatrix}^T} = \begin{cases} -\frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|} & \text{if } \|\boldsymbol{d}_i\| = \min(\|\boldsymbol{d}_1\|, \|\boldsymbol{d}_2\|, \cdots, \|\boldsymbol{d}_N\|), \\ \boldsymbol{0} & \text{if } \|\boldsymbol{d}_i\| \neq \min(\|\boldsymbol{d}_1\|, \|\boldsymbol{d}_2\|, \cdots, \|\boldsymbol{d}_N\|), \end{cases} \tag{52}$$

and when one dimensional lines are being used, it is:

$$\frac{\partial d_e}{\partial \begin{bmatrix} x_D^{(1)} & y_D^{(1)} \end{bmatrix}^T} = \begin{cases} -\frac{\boldsymbol{h}}{\|\boldsymbol{h}\|} \\ \text{if } \boldsymbol{a}\cdot\boldsymbol{h} \leq 0, \\ \frac{1}{\|\boldsymbol{g}\|}\left[\frac{1}{\|\boldsymbol{a}\|^2}\left((\boldsymbol{a}\otimes\boldsymbol{h})^T + (\boldsymbol{a}\cdot\boldsymbol{h})\boldsymbol{I}\right) - \boldsymbol{I}\right]\boldsymbol{g} \\ \text{if } 0 < \boldsymbol{a}\cdot\boldsymbol{h} < \boldsymbol{a}\cdot\boldsymbol{a}, \\ \boldsymbol{0} \\ \text{if } \boldsymbol{a}\cdot\boldsymbol{h} \geq \boldsymbol{a}\cdot\boldsymbol{a}. \end{cases} \tag{53}$$

for the first point of the line and

$$\frac{\partial d_e}{\partial \begin{bmatrix} x_D^{(2)} & y_D^{(2)} \end{bmatrix}^T} = \begin{cases} \boldsymbol{0} \\ \text{if } \boldsymbol{a}\cdot\boldsymbol{h} \leq 0, \\ -\frac{1}{\|\boldsymbol{g}\|\|\boldsymbol{a}\|^2}\left((\boldsymbol{a}\otimes\boldsymbol{h})^T + (\boldsymbol{a}\cdot\boldsymbol{h})\boldsymbol{I}\right)\boldsymbol{g} \\ \text{if } 0 < \boldsymbol{a}\cdot\boldsymbol{h} < \boldsymbol{a}\cdot\boldsymbol{a}, \\ -\frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \\ \text{if } \boldsymbol{a}\cdot\boldsymbol{h} \geq \boldsymbol{a}\cdot\boldsymbol{a}. \end{cases} \tag{54}$$

for the second point of the line.

8 Numerical Examples

In this section we demonstrate the performance of the algorithm for several benchmark problems. To avoid having the loads and supports settle into local minima too early, we use a continuation strategy on the SIMP penalty parameter p where the optimization runs with $p = 1$ until the average change in the density design variables from the previous iteration is less than 10^{-3}, after which p is increased by 0.5. This process repeats until $p = 3$, and at this point the optimization continues until the average change in density variables is less than 10^{-4}, at which point we consider the optimization fully converged and stop the program. We use the average change, rather than the maximum change that is often used in topology optimization (Andreassen et al. 2011), since it is less sensitive to localized changes in the density caused by small oscillations in the load and support design variables (Ferrari and Sigmund 2020).

Table 2: Optimization parameters common to all example problems

E_0	1 Pa
E_{min}	$E_0 \times 1e^{-9}$ Pa
ν	0.3
r_{min}	2.5 Elements
ϵ	0.1
k_0	$E_0 \times 1e^{-3}$ N/m
f_0	1 N
k_0^{in}	$E_0 \times 1e^{-3}$ N/m
k^{out}	$E_0 \times 1e^{-3}$ N/m
b	2
r	5 mm
P	4
Q	10

In the design plots shown in the following sections, red contour lines represent the support geometry projected by the Gaussian function at a radius r. Loads

are similarly shown by blue contour lines, with an arrow pointing in the direction θ and originating at the design variable coordinates (x_f, y_f).

The common parameters used for all of the following examples are summarized in Table 2. The MMA move limits are set to plus or minus 20% of the current values for the densities, $2r$ for the load and support locations, and 2 degrees for the load orientation.

8.1 Minimum Compliance Design

We begin by validating the framework for a simple minimum compliance cantilever beam problem which has a well known solution with obvious optimal locations for the loads and supports. A rectangular design domain of dimensions 30x7.5 centimeters and discretized by a grid of 200x50 elements is initialized with a uniform distribution of 20% density, a distributed line support at the left side with both endpoints at the same position, and a concentrated load at the right side. We note that although the structure is initially supported by only a single point, the Gaussian function projects a circle of finite radius r which provides rotational stiffness. The design variables included are the densities, support line endpoint coordinates, and load point coordinates:

$$\boldsymbol{z} = \left[\boldsymbol{\rho}\ \boldsymbol{x}_s\ \boldsymbol{y}_s\ \boldsymbol{x}_f\ \boldsymbol{y}_f\right]. \tag{55}$$

The load orientation is set to a constant value of $\theta = -90°$ (pointing straight downwards). The design domains for the support and load coordinates are restricted to one fourth of the length of domain from the left and right ends as shown in Figure 5a. Minimum and maximum values of the points are set such that they must remain at least a distance of the Gaussian function radius, r, from the overall domain boundaries.

Running the optimization, we get the design we would expect with fast convergence. The loads and supports move horizontally as close together as possible to minimize the moment arm, and the two endpoints of the distributed support move vertically as far apart as possible to maximize the second moment of area. The density resolves to a cantilever beam design that is typical with the standard SIMP method. The optimized design is shown in Figure 5b and the convergence history is shown in Figure 5c. The sharp kinks in the objective function history starting near 25 iterations correspond to the point where the loads and supports reach their vertical and horizontal limits, and the small upward jumps afterwards are caused by the the continuation scheme when the SIMP penalty parameter p increases by 0.5.

As a second compliance minimization problem we optimize a bridge structure. We initialize a 20x20 centimeter design domain with 200x200 elements and 15% uniform density as shown in Figure 6a. One point support is placed in each of the top corners of the domain, and two overlapping supports are placed at each of the two bottom corners for a total of six support points. A distributed line load with a solid non-design region projected underneath is placed across the width of the center of the domain with the orientation initially pointing downwards. These parameters are represented by the following vector of design variables:

$$\boldsymbol{z} = \left[\boldsymbol{\rho}\ \boldsymbol{x}_s\ \boldsymbol{y}_s\ \boldsymbol{x}_f\ \boldsymbol{y}_f\ \theta\right]. \tag{56}$$

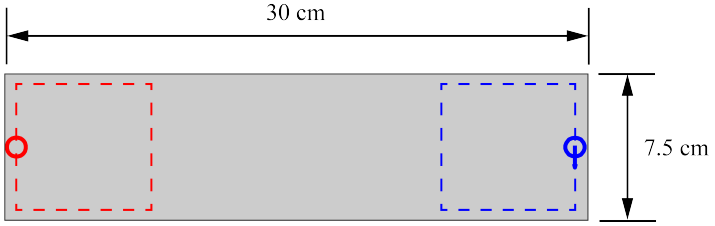
The upper bound of the material volume fraction is constrained to 15%, and the supports are allowed to move only along the edges of the domain as shown in Figure 6a by the red dashed lines. The distributed load is allowed to move in the middle third of the domain as shown in Figure 6a by the blue dashed region.

The results of the optimization are shown in 6b. As would be expected, the distributed load moves as close as possible to the supports along the bottom edge, which distribute themselves underneath it. The load also remains distributed across the entire domain and the orientation does not change from its initial downward direction. The supports allowed to move along the vertical edges place themselves at the ends of the bridge to directly support it.

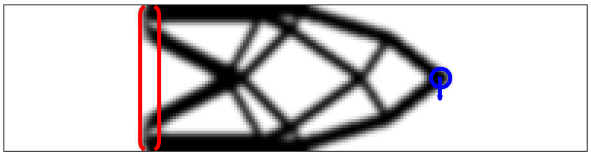
8.2 Compliant Mechanism Design

While for the simple compliance minimization problems in the previous section the optimal locations of the loads and supports were somewhat obvious and could be guessed intuitively, typically the same cannot be said for compliant mechanism design problems. The positions and orientations of the boundary conditions have a significant effect on the motion of the output degrees of freedom and initial guesses based on intuition are likely suboptimal.

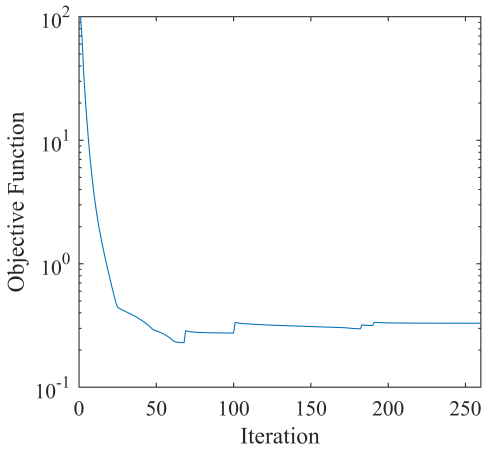
To demonstrate this, we use the standard benchmark compliant mechanism problem of a displacement inverter. The domain is initialized as a 20x20 centimeter square with a grid of 200x200 elements, with a uniform 20% material density. This same 20% value was used for the constraint on the volume fraction upper bound, and the load and support locations are placed in the positions shown in 7a, which are the typical locations in the inverter mechanism problem. A spring of stiffness k^{out} is placed on the degree of freedom for the horizontal displacement at the center of the right edge, which



(a) Initial design with allowable load (blue) and support (red) regions

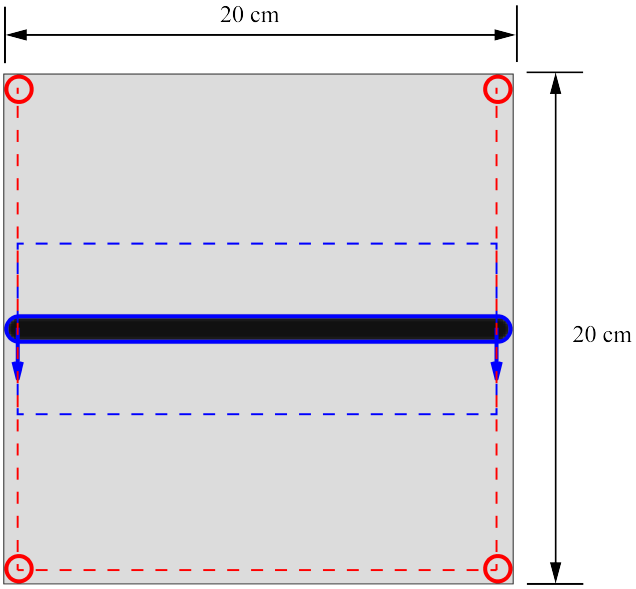


(b) Final optimized design

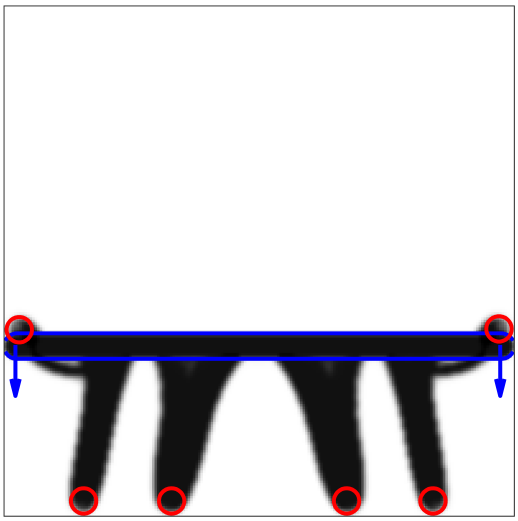


(c) Convergence history of a scaled value of the compliance

Fig. 5: Results of the cantilever beam problem



(a) Initial design with allowable load (blue) and support (red) regions



(b) Final optimized design

Fig. 6: Results for the bridge design problem

is the displacement being minimized as the objective function (to maximize the displacement in the leftward direction). This initial design and volume constraint is the same for each of the following examples. First, we set only the material densities as the design variables:

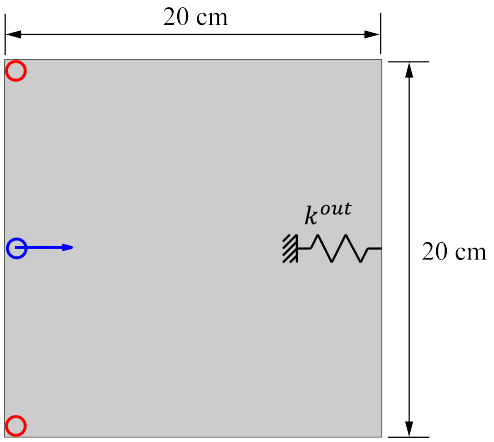
$$\boldsymbol{z} = \left[\boldsymbol{\rho}\right]. \tag{57}$$

The results of this optimization gives a familiar displacement inverter design, shown in Figure 7b.

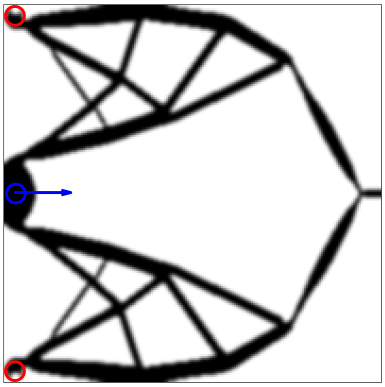
As a second problem, we add the positions of the load and supports as design variables:

$$\boldsymbol{z} = \left[\boldsymbol{\rho}\ \boldsymbol{x}_s\ \boldsymbol{y}_s\ \boldsymbol{x}_f\ \boldsymbol{y}_f\right]. \tag{58}$$

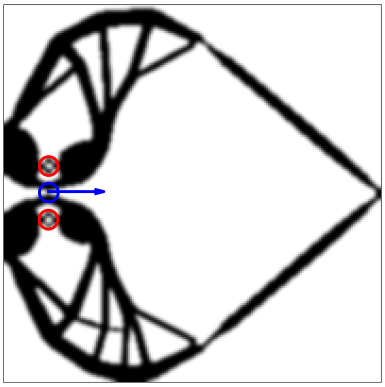
These positions are unconstrained and can move anywhere in the design domain, with the exception that they must remain at least a distance of r from the edges of the domain. Since asymmetry in the design was observed, we also include an additional constraint function to prevent the output displacement from deviating



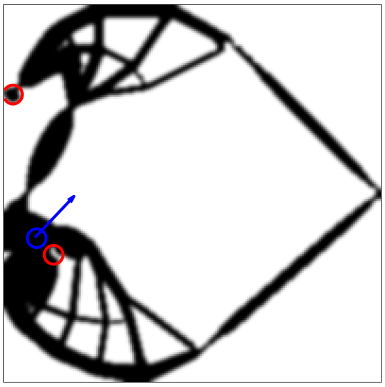
(a) Initial design



(b) Optimized design for load and support locations fixed at their conventional locations at the edge of the domain



(c) Optimized design for freely movable load and support locations



(d) Optimized design for freely moving boundary condition locations with rotating load orientation

Fig. 7: Results for the displacement inverter design problem

significantly in the vertical direction:

$$(U^y_{out})^2 \leq (0.05U^x_{out})^2. \tag{59}$$

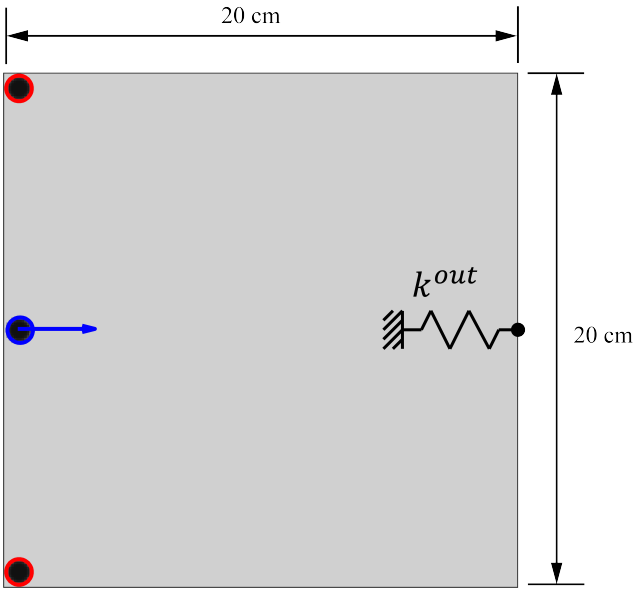
The problem results in a different design, shown in Figure 7c, where the supports have moved very close to the input load, which shifts to a position further inside the domain. As a result of this change in the boundary condition locations, the output displacement increases by 123% compared to the conventional design that had predetermined boundary conditions.

For a third problem, we include the orientation of the load as a design variable:

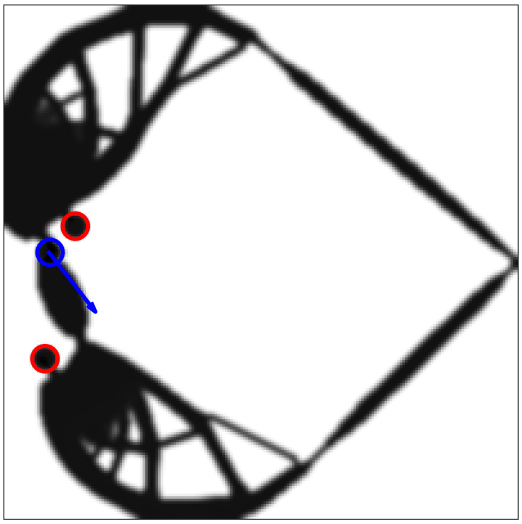
$$\boldsymbol{z} = \left[\boldsymbol{\rho}\ \boldsymbol{x}_s\ \boldsymbol{y}_s\ \boldsymbol{x}_f\ \boldsymbol{y}_f\ \theta\right]. \tag{60}$$

Keeping the constraint function for controlling the unwanted vertical output displacement, equation (59), this results in an asymmetrical design, shown in Figure 7d, where the load is applied at an oblique angle relative to the direction of the output displacement. A displacement 151% larger than the conventional fixed boundary condition design is achieved for the same input force magnitude, which shows that the optimizer is able to exploit the additional design freedom to obtain better objective function values. The increase in performance and the counterintuitive, asymmetrical design when the boundary conditions are included as design variables shows the effectiveness of allowing them to be determined automatically by the optimizer.

The asymmetrical design of Figure 7d performs well based on the finite element analysis, however the support closest to the applied load is somewhat difficult to interpret as a manufacturable structure. It is surrounded by solid material with a region of soft intermediate density in the middle, making the support rotate and act more like a pin joint than a compliant hinge. To get a fully compliant mechanism design with no need for bearings or significant post-processing, we run the asymmetrical inverter problem once more and include variable non-design solid regions projected on both the load and the supports. We implement this by defining a



(a) Initial design



(b) Optimized design

Fig. 8: Results for the asymmetric displacement inverter problem with variable non-design regions included

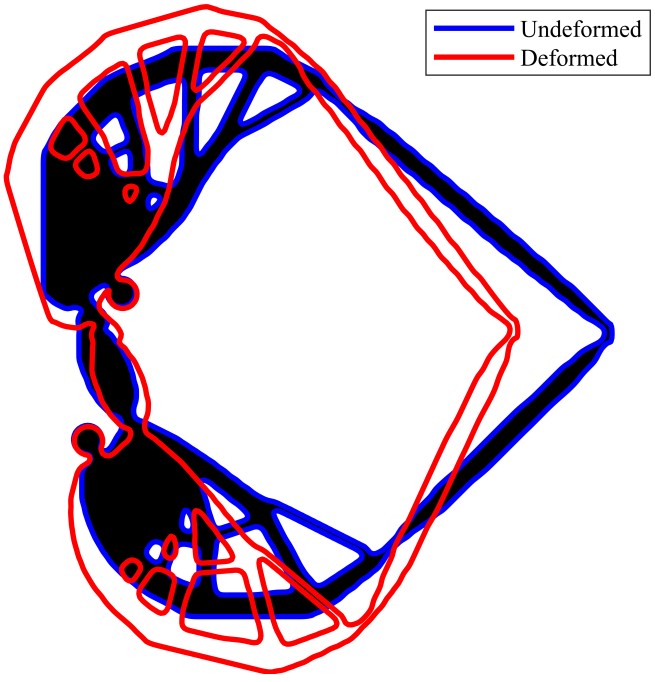


Fig. 9: Deformation of the asymmetrical inverter design superimposed on the undeformed design

new distance function of three points, $d(\boldsymbol{x}_s, \boldsymbol{y}_s, x_f, y_f)$, and using it in the equations of section 6. The initial design and optimized results are shown in Figure 8, where there are now clearly formed compliant hinges for each boundary condition point. By forcing the material to be solid at the load and support locations, the optimizer was no longer able to take advantage of the soft intermediate density material to make a pin joint. This came at a small cost to the overall performance, with the design achieving 1.7% less displacement at the output point compared to the asymmetrical inverter without the non-design regions. The deformation is visualized in Figure 9, where the bending of compliant hinges and a substantial geometric advantage can be seen. The locations of the supports translate very little in relation to the input load and the output point, showing that the stiffness of the support springs is adequately high.

To validate our methodology, we manufactured a half-scale model of the design of Figure 8 on an Objet260 Connex3 3D printer using the digital material FLX9885-DM, a blend of VeroWhite and TangoBlack+ polymers. The mechanism's supports were inserted into a base plate, 3D printed from VeroWhite, with a cutout included to guide the input actuation handle at the correct angle. Figure 10 shows the 3D printed model as it is actuated through a large displacement. While the numerical modeling was only based on linear elasticity, the physical prototype is still able to maintain a small vertical output displacement through the large actuation shown in Figure 10. In both the partially actuated and fully actuated states shown in Figure 10, the vertical displacement of the output point is about 8% the magnitude of the horizontal displacement based on measurements of the image. This can be compared to the 5% constraint imposed on the design in the topology optimization by equation (59).

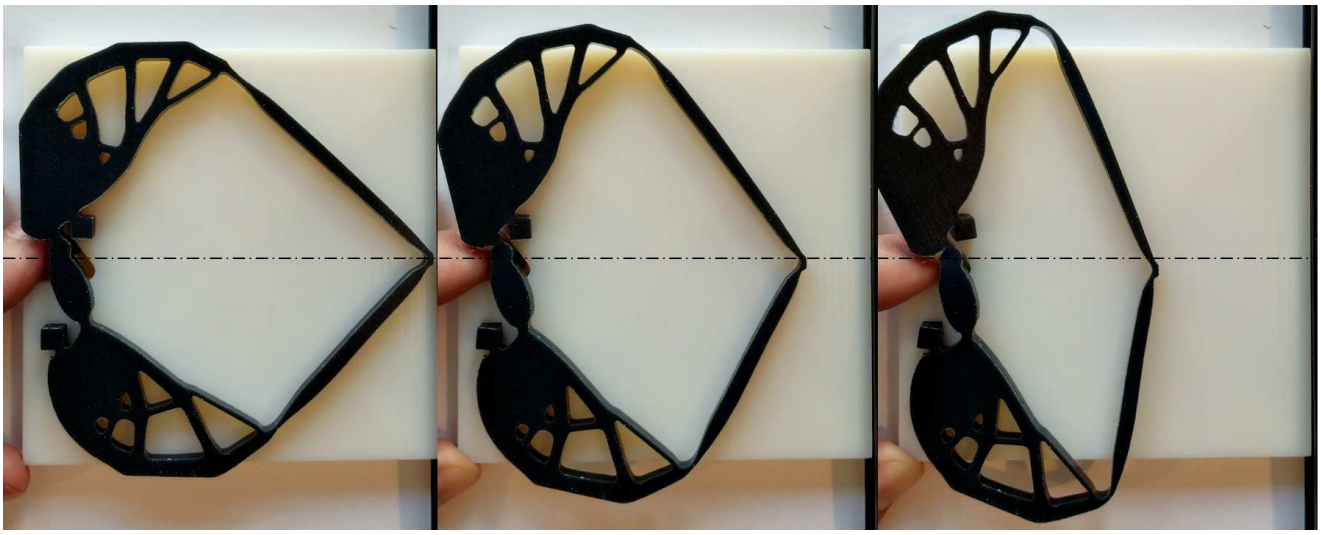


Fig. 10: 3D printed model of the displacement inverter design

9 Conclusions

In this paper, we introduced a framework for including variable load and support boundary conditions in topology optimization. Starting with the standard SIMP method with linear elasticity, we extended it to use a system of spring elements to model elastic supports and loads. The stiffness of the springs and the magnitudes of input forces applied to every structural element were parameterized and controlled by a higher-order Gaussian function. By using the distance functions of simple points and lines, the Gaussian function was used to model the effective location and orientation of different boundary conditions in a smooth, differentiable, and optimizable way with minimal numbers of additional design variables.

Two examples of compliance minimization problems were shown, demonstrating the effectiveness and efficiency of the Gaussian function approach in automatically finding optimal placements of the boundary conditions. Several examples of compliant mechanism problems were then presented, resulting in significantly increased performance over designs in which the boundary conditions were defined a priori. Using our method to design displacement inverters, we produced several designs with more than double the performance of the design with conventionally predetermined boundary conditions. The relatively counterintuitive design of these mechanisms shows the usefulness of allowing a numerical optimizer to automatically find the optimal boundary conditions, rather than relying only on experience or trial and error methods.

The super-Gaussian projection method proposed here was applied to linear elasticity problems with the boundary conditions modeled by simple points and straight lines. However, it should be extendable to more complex problems such as those that include geometric nonlinearity, multiple physics disciplines, or more complex boundary condition geometries. Extension to three dimensions may present some challenges with maintaining manufacturable supports that do not become entirely enclosed in material, but otherwise should be straightforward. Future work will utilize the Gaussian function method developed in this paper for some of these more complex problems.

Acknowledgements The authors would like to thank Professor Krister Svanberg for providing the MATLAB MMA code.

Declarations

Funding

This research was funded by the National Science Foundation through grant #1752045.

Conflict of Interest

The authors declare that there is no conflict of interest.

Code availability (software application or custom code)

The MATLAB code used to produce the results in this work can be requested from the corresponding author Lee Alacoque at leea2@illinois.edu.

Authors' contributions

Lee Alacoque: Conceptualization, derivations, coding, writing, artwork.
Kai A. James: Conceptualization, advising, writing, proofreading and review.

Replication of Results

Readers can replicate the results by implementing the formulations presented in this paper in a standard density-based topology optimization program. All results presented in the study were generated using a modified version of the top88 MATLAB code written by Andreassen et al. (Andreassen et al. 2011) with the method of moving asymptotes MATLAB function provided by Professor Krister Svanberg (Svanberg 1987). If readers wish to obtain a copy of the code, it can be requested from the corresponding author Lee Alacoque at leea2@illinois.edu.

References

Alacoque, Lee, Ryan T Watkins, and Ali Y Tamijani (2021). "Stress-based and robust topology optimization for thermoelastic multi-material periodic microstructures". In: *Computer Methods in Applied Mechanics and Engineering* 379, p. 113749. DOI: https://doi.org/10.1016/j.cma.2021.113749.

Alonso, Cristina, Ruben Ansola, and Osvaldo M Querin (2014). "Topology synthesis of multi-input–multi-output compliant mechanisms". In: *Advances in Engineering Software* 76, pp. 125–132. DOI: https://doi.org/10.1016/j.advengsoft.2014.05.008.

Ambrozkiewicz, Olaf and Benedikt Kriegesmann (2021). "Simultaneous topology and fastener layout optimization of assemblies considering joint failure". In: *International Journal for Numerical Methods in Engineering* 122.1, pp. 294–319. DOI: https://doi.org/10.1002/nme.6538.

Andreassen, Erik et al. (2011). "Efficient topology optimization in MATLAB using 88 lines of code". In: *Structural and Multidisciplinary Optimization* 43.1, pp. 1–16. DOI: https://doi.org/10.1007/s00158-010-0594-7.

Bendsoe, Martin Philip and Ole Sigmund (2013). *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.

Bhattacharyya, Anurag, Cian Conlan-Smith, and Kai A James (2019). "Design of a bi-stable airfoil with tailored snap-through response using topology optimization". In: *Computer-Aided Design* 108, pp. 42–55. DOI: https://doi.org/10.1016/j.cad.2018.11.001.

Bruns, Tyler E and Daniel A Tortorelli (2001). "Topology optimization of non-linear elastic structures and compliant mechanisms". In: *Computer methods in applied mechanics and engineering* 190.26-27, pp. 3443–3459. DOI: https://doi.org/10.1016/S0045-7825(00)00278-4.

Buhl, Thomas (2002). "Simultaneous topology optimization of structure and supports". In: *Structural and multidisciplinary optimization* 23.5, pp. 336–346. DOI: https://doi.org/10.1007/s00158-002-0194-2.

Ferrari, Federico and Ole Sigmund (2020). "A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D". In: *Structural and Multidisciplinary Optimization* 62.4, pp. 2211–2228. DOI: https://doi.org/10.1007/s00158-020-02629-w.

Frecker, M, Noboru Kikuchi, and S Kota (1999). "Topology optimization of compliant mechanisms with multiple outputs". In: *Structural optimization* 17.4, pp. 269–278.

Jessee, Alex et al. (2020). "Simultaneous packing and routing optimization using geometric projection". In: *Journal of Mechanical Design* 142.11. DOI: https://doi.org/10.1115/1.4046809.

Kazemi, Hesaneh, Ashkan Vaziri, and Julián A Norato (2020). "Multi-material topology optimization of lattice structures using geometry projection". In: *Computer Methods in Applied Mechanics and Engineering* 363, p. 112895. DOI: https://doi.org/10.1016/j.cma.2020.112895.

Lee, Edmund, Kai A James, and Joaquim RRA Martins (2012). "Stress-constrained topology optimization with design-dependent loading". In: *Structural and Multidisciplinary Optimization* 46.5, pp. 647–661. DOI: https://doi.org/10.1007/s00158-012-0780-x.

Lee, Edmund and Joaquim RRA Martins (2012). "Structural topology optimization with design-dependent pressure loads". In: *Computer Methods in Applied Mechanics and Engineering* 233, pp. 40–48. DOI: https://doi.org/10.1016/j.cma.2012.04.007.

Norato, JA, BK Bell, and Daniel A Tortorelli (2015). "A geometry projection method for continuum-based topology optimization with discrete elements". In: *Computer Methods in Applied Mechanics and Engineering* 293, pp. 306–327. DOI: https://doi.org/10.1016/j.cma.2015.05.005.

Pollini, Nicolò and Oded Amir (2020). "Mixed projection- and density-based topology optimization with applications to structural assemblies". In: *Structural and Multidisciplinary Optimization* 61.2, pp. 687–710. DOI: https://doi.org/10.1007/s00158-019-02390-9.

Rakotondrainibe, Lalaina, Grégoire Allaire, and Patrick Orval (2020). "Topology optimization of connections in mechanical systems". In: *Structural and Multidisciplinary Optimization*, pp. 1–17. DOI: https://doi.org/10.1007/s00158-020-02511-9.

Sigmund, Ole (1997). "On the design of compliant mechanisms using topology optimization". In: *Journal of Structural Mechanics* 25.4, pp. 493–524. DOI: https://doi.org/10.1080/08905459708945415.

— (2001). "Design of multiphysics actuators using topology optimization–Part I: One-material structures". In: *Computer methods in applied mechanics and engineering* 190.49-50, pp. 6577–6604. DOI: https://doi.org/10.1016/S0045-7825(01)00251-1.

Svanberg, Krister (1987). "The method of moving asymptotes—a new method for structural optimization". In: *International journal for numerical methods in engineering* 24.2, pp. 359–373. DOI: https://doi.org/10.1002/nme.1620240207.

Swartz, Kenneth E and Kai A James (2019). "Gaussian layer connectivity parameterization: a new approach to topology optimization of Multi-Body mechanisms". In: *Computer-Aided Design* 115, pp. 42–51. DOI: https://doi.org/10.1016/j.cad.2019.05.008.

Zhang, Shanglong, Arun L Gain, and Julián A Norato (2018). "A geometry projection method for the topology optimization of curved plate structures with placement bounds". In: *International Journal for Numerical Methods in Engineering* 114.2, pp. 128–146. DOI: https://doi.org/10.1002/nme.5737.

Zhang, Shanglong, Julián A Norato, et al. (2016). "A geometry projection method for the topology optimization of plate structures". In: *Structural and Multidisciplinary Optimization* 54.5, pp. 1173–1190. DOI: https://doi.org/10.1007/s00158-016-1466-6.

Zhu, Benliang, Qi Chen, et al. (2018). "Design of fully decoupled compliant mechanisms with multiple degrees of freedom using topology optimization". In: *Mechanism and Machine Theory* 126, pp. 413–428. DOI: https://doi.org/10.1016/j.mechmachtheory.2018.04.028.

Zhu, Benliang, Xianmin Zhang, et al. (2020). "Design of compliant mechanisms using continuum topology optimization: a review". In: *Mechanism and Machine Theory* 143, p. 103622. DOI: https://doi.org/10.1016/j.mechmachtheory.2019.103622.

Zhu, JH and WH Zhang (2010). "Integrated layout design of supports and structures". In: *Computer Methods in Applied Mechanics and Engineering* 199.9-12, pp. 557–569. DOI: https://doi.org/10.1016/j.cma.2009.10.011.