Structural and Multidisciplinary Optimization (2018) 58:1297–1310
https://doi.org/10.1007/s00158-018-1939-x

EDUCATIONAL ARTICLE

A sequential element rejection and admission (SERA) topology optimization code written in Matlab

Rubén Ansola Loyola[1] · Osvaldo M. Querin[2] · Alain Garaigordobil Jiménez[3] · Cristina Alonso Gordoa[3]

Received: 3 February 2017 / Revised: 30 January 2018 / Accepted: 7 February 2018 / Published online: 19 February 2018

Abstract
This paper presents the Matlab implementation of the Sequential Element Rejection and Admission (SERA) method for topology optimization of structures and compliant mechanisms. The lines comprising this code include definition of design domain, finite element analysis, sensitivity analysis, mesh-independency filter, optimization algorithm and display of results. Extensions and changes in the algorithm are also included in order to solve multiple load cases, active and passive elements and compliant mechanisms design. The code is intended for educational purposes and introduces an alternative approach to traditional structural topology optimization algorithms. The complete code is provided in the Appendix.

Keywords Topology optimization · SERA method · Matlab · Compliant mechanisms

1 Introduction

Topology optimization is a computational approach that optimizes material distribution within a fixed design domain and for a given set of loads and boundary conditions such that the resulting layout meets a prescribed set of the design requirements. It is an expanding research field of computational mechanics which has been growing very rapidly and has attracted the interest of numerous applied mathematicians and engineering designers, becoming extremely popular in the last years. Topology optimization has interesting applications in mechanics, multiphysics and micro- and nanotechnologies, allowing for efficient designs with minimal preconceived decisions.

The first work on topology optimization was published over a century ago by Michell (1904). Since the landmark paper of Bendsoe and Kikuchi (1988), where a so-called microstructure and homogenization based approach was used, numerical methods for topology optimization have been investigated extensively. At present the most popular topology optimization method is the SIMP method, which stands for "Solid Isotropic Material with Penalization", proposed in the late eighties by Bendsoe (1989). In this approach, the variables are the element relative densities which are assumed to be constant within each element of the discretized design domain. It is well known that the optimal solution of the topology optimization problem depends on the discretization level, as observed in many applications based on the finite element method. In order to avoid checkerboards and mesh-dependencies some sort of restriction on the resulting design must be introduced, combining the power law approach with, e.g., a perimeter constraint (Haber et al. 1996), a gradient constraint (Borrvall 2001) or with filtering techniques (Sigmund 1994). A number of papers have also appeared on solving the topology optimization problem as an integer problem (Beckers 1999) and other non-gradient or semi-random methods like genetic algorithms (Hajela and Lee 1995). During last years Level Set Methods have emerged as an attractive and promising alternative to perform structural shape and topology optimization, inspired in the work on topological derivatives by Sokolowski and Zochowski (1999) and the paper by Sethian and Wiegman (2000).

Responsible Editor: Anton Evgrafov

✉ Rubén Ansola Loyola
ruben.ansola@ehu.es

[1] Department of Mechanical Engineering, Faculty of Engineering, University of The Basque Country, Alda, Urquijo s/n, 48013 Bilbao, Spain

[2] School of Mechanical Engineering, University of Leeds, Leeds LS2 9JT, UK

[3] Department of Mechanical Engineering, University of The Basque Country, Alda, Urquijo s/n, 48013 Bilbao, Spain

Apart from above mentioned approaches, a number of heuristic or intuition based approaches have effectively addressed a variety of size, shape and topology optimization problems. An important branch of these approaches for topology optimization is the evolutionary structural optimization approach (ESO) by Xie and Steven (1993). The initial concept was that by systematically removing inefficient materials (elements with lowest strain energy density), the structure evolves towards an optimum. Its application in topology optimization of continuum media is quite extensive, see e.g., Xie and Steven (1997). Although initially solely based on intuition, this basic idea has developed from simple hard-kill strategies to more efficient soft-kill bi-directional schemes (BESO), which allow efficient materials to be added in addition to the inefficient ones being removed (Querin 1997). The newer BESO method has demonstrated its strength in solving a variety of topology optimization problems (Querin et al. 1998; Yang et al. 1999), but as it is presently defined, it uses a power law (SIMP) parametrization strategy and standard filtering techniques similar to those used in the density approach in order to stabilize results (Huang and Xie 2010), so it could be categorized as a discrete update version of the standard SIMP scheme (Sigmund and Maute 2013). On the other hand, the ESO/BESO approaches have been criticized for failing in certain situations and lead to entirely non-optimal solutions (Zhou and Rozvany 2001; Rozvany 2009). Rozvany and Querin proposed some improvements of this method under the term SERA (Sequential Element Rejection and Admission) where a “virtual material” was introduced, without the use of any intermediate densities or power law interpolations (Rozvany and Querin 2004). Additionally, two separate criteria are considered in the topology optimization process by SERA method, where the sensitivity numbers of ‘real’ and ‘virtual’ material present in the domain are sorted out separately. It was demonstrated that elements that are added or removed are not the same when a single list of sensitivity numbers is used to perform the optimization, because elements actual material status is not taken in consideration. These ideas were developed for fully stressed design (Brodie 2007) and extended to most of the classical problems in structural topology optimization and compliant mechanisms design (Alonso et al. 2013, 2014a, b).

Concerning educational articles with implementations of topology optimization algorithms, there have been a number of readily available educational computer tools for MATLAB. The popular 99-line Matlab code published by Sigmund (2001) played a very important role in general acceptance of topology optimization methods. This code has been optimized later for speed and compactness by Andreassen et al. (2011), who presented a faster 88-line program with improved assembly and filtering strategies. Also for Matlab, Allaire (2012) and Challis (2010) implemented programs making use of the level-set method with continuous and discrete variables, respectively. Three-dimensional topology optimization codes can be found in the work presented by Liu and Tovar (2014), including extensions for multiple load cases, active and passive elements, continuation strategy, synthesis of compliant mechanisms and heat conduction problems. Changing a handful of lines in the aforementioned 99-line code, the BESO scheme was implemented in Matlab by Huang and Xie (2010). Recently Matlab codes with extensions to Pareto strategies (Suresh 2010) and alternative element discretizations using polygonal finite elements (Talischi et al. 2012a, b) have been presented. Polygonal finite elements were used by Pereira et al. (2016) for fluid flow topology optimization. Recently, a methodology for ground structure based topology optimization in arbitrary 2D and 3D domains have been implemented using Matlab (Zegard and Paulino 2014, 2015). Finally, bridging topology optimization and additive manufacturing technologies, a new tool named TOPslicer was developed in Matlab to generate suitable outputs for additive manufacturing (Zegard and Paulino 2016).

The present paper explains the implementation and use of a Matlab program that incorporates the strategies for topology optimization based on the Sequential Element Rejection and Admission (SERA) method. The proposed code is very similar to the 99-line paper except for the material update subroutine, where the optimality criterion has been replaced by the SERA algorithm. This program can be effectively used in personal computers for educational purposes of engineering students or newcomers interested in the field of topology optimization, both as an educational tool in courses on topology optimization, but also as a platform for research and development of alternative topology optimization approaches. The rest of this paper is organized as follows. Section 2 briefly reviews the mathematical formulation of the topology optimization problem for compliance minimization of statically loaded structures. The optimization problem is solved using the Sequential Element Rejection and Admission (SERA) procedure with the addition of a mesh independent filter. Section 3 discusses details of the Matlab implementation for the algorithm and the use of the code is demonstrated through several benchmark examples in section 4. These examples include problems with different boundary conditions, multiple load cases, passive elements and compliant mechanisms design. Section 5 offers some closing thoughts. Finally, the Appendix provides the complete Matlab code for the proposed approach.

2 The topology optimization problem

2.1 Problem formulation

The topology optimization problem for maximum stiffness structural design is defined as the minimization of the compliance, where the objective is to find the material density distribution that minimizes the structure's deformation under the prescribed support and loading conditions, subjected to a volume constraint. This optimization problem can be written as:

$$\begin{aligned} \min_{\boldsymbol{\rho}} : \quad & c(\boldsymbol{\rho})\mathbf{U}^{\mathbf{T}}\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} \mathbf{U}_e^T \mathbf{K}_e \mathbf{U}_e \\ \textit{subject} \quad \textit{to}: \quad & \frac{V(\boldsymbol{\rho})}{V_0} = \sum_{e=1}^{N} \frac{\rho_e V_e}{V_0} \leq f \rho_e \\ = \{\rho_{\min}, 1\} e = 1, \ldots, N \quad & \mathbf{KU} = \mathbf{F} \end{aligned} \tag{1}$$

where U and F are the global displacement and force vectors, K is the global stiffness matrix, U_e and K_e are the element displacement vector and stiffness matrix, respectively, ρ is the vector of design variables, N is the number of elements used to discretize the design domain, and $V(\rho)$ and V_0 are the material volume and design domain volume, respectively. V_e corresponds to each finite element volume and f is the prescribed volume fraction. The global stiffness matrix K is assembled from the element stiffness matrices K_e, which are obtained multiplying the element isotropic stiffness matrix K_0 by the density of the element, since Young's moduli are assumed to depend linearly on the density variable, i.e., $K_e(\rho_e)= \rho_e K_0$. These design variables are discrete in the SERA method, so density can only be zero or one. Nevertheless, in order to avoid obtaining a singular stiffness matrix, a non-zero lower bound is assigned to density (ρ_{min}) as shown in (1). The minimum non-zero relative density has been set to $\rho_{min} = 10^{-9}$, in order to avoid the optimizer in taking advantage of low density regions and to comply with the stiffness ratio between solid and void elements applied in the 99 line code.

2.2 Sequential element rejection and admission procedure

The SERA topology optimization method is bi-directional in nature and considers two separate material models: 1) 'Real' material and 2) 'Virtual' material with negligible stiffness (Rozvany and Querin 2002a, b). Two separate criterions of rejection and admission of elements allow material to be introduced and removed from the design domain by changing its status from 'virtual' to 'real' and vice versa (Fig. 1), so that the final topology is made of all the 'real' material present at the end of the optimization. A sensitivity analysis is performed and the resulting elemental sensitivity numbers obtained are the ones that define the elements rejection and admission criteria. 'Real' and 'virtual' elements are sorted according to their sensitivity value from highest to lowest. 'Real' elements are changed to 'virtual' if their sensitivity number is below a

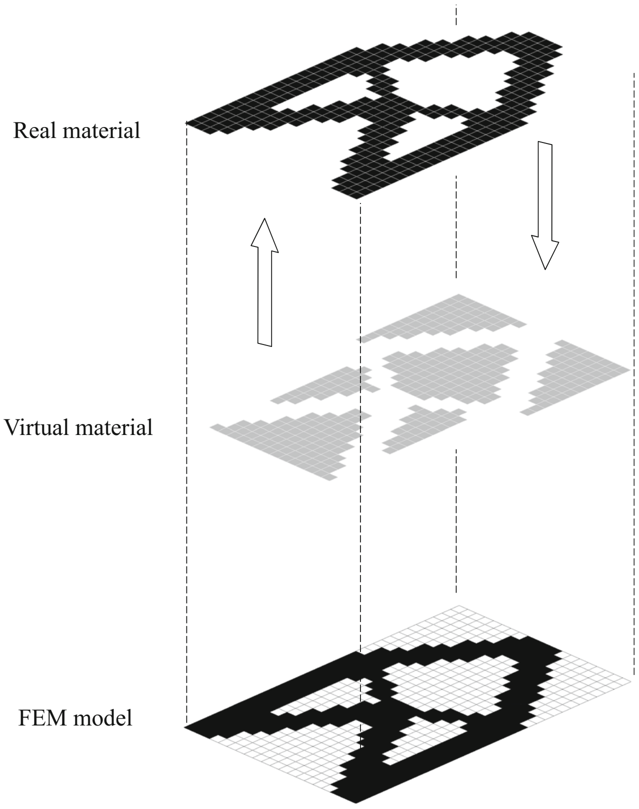

Fig. 1 'Real' and 'virtual' material models

defined threshold rejection value and 'virtual' elements are transformed into 'real' if their sensitivity is higher than a defined threshold admission value. The steps that drive the SERA method are given below.

1. Define the design problem. The maximum design domain must be defined and discretized for finite element analysis. All boundary constraints, loads and the target volume fraction f must also be specified.
2. Calculate the variation of the volume fraction in the ith iteration which consists of the volume fraction to be added and removed.
3. Carry out a finite element analysis to compute elemental and global stiffness matrices as well as the displacement vector.
4. Calculate the elemental sensitivity numbers.
5. Apply a mesh independent filtering to the sensitivity numbers.
6. Separate the sensitivity numbers into 'real' and 'virtual' materials.
7. Define the threshold values for 'real' and 'virtual'material.
8. Remove and add elements.
9. Calculate the volume of the 'real' material in the domain.
10. Calculate the stopping criterion
11. Repeat steps (2) through (10) until the target volume is reached and the optimization converges. The final

topology is represented by the 'real' material in the design domain.

2.3 Material rejection, addition and re-distribution

Material is added and removed from the design domain in a two stage process: 1) Different amounts of material are added and removed in each iteration until the target volume fraction f is reached. 2) Once the target volume fraction is reached, material re-distribution takes place by both adding and removing the same amount of material until the problem converges.

The target volume fraction $VF(i)$ of stage 1) can be calculated using (2). The fraction of material to be removed in the ith iteration is then given by (3). This value is then separated into the volume fraction that will be added $\Delta VF_{\mathrm{add}}(i)$ and the volume fraction that will be removed $\Delta VF_{\mathrm{rem}}(i)$. These terms are given in (4) and (5). Figure 2 describes the scheme of the material removal and addition starting from a full design domain, where each iteration consists of two sub-steps which add and remove material from the design domain. As it can be shown in Fig. 2, initially a larger amount than the strictly required is removed. The second step adds material so that the final change is exactly the amount given in (3).

$$VF(i) = \max(VF(i-1)\cdot(1-PR), f) \tag{2}$$

$$\Delta VF(i) = |VF(i)-VF(i-1)| \tag{3}$$

$$\Delta VF_{add}(i) = \Delta VF(i)\cdot(SR-1) \tag{4}$$

$$\Delta VF_{rem}(i) = \Delta VF(i)\cdot SR \tag{5}$$

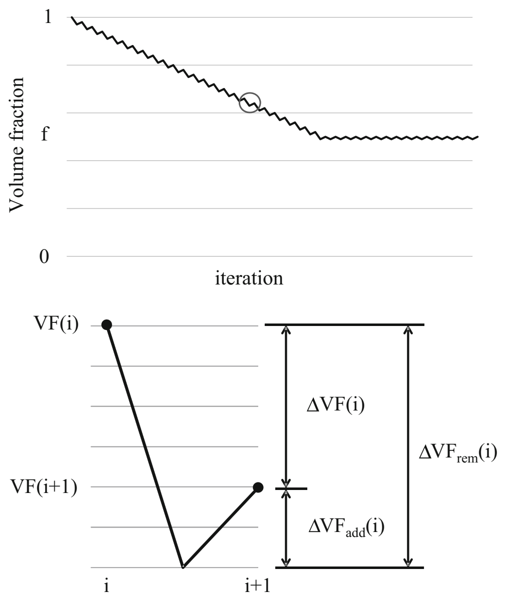

Fig. 2 Material removal and addition scheme

where PR is the Progression Rate, with typical values ranking between 0.01 and 0.05 and SR is the Smoothing Ratio, with typical values in the range between 1.2 and 1.4.

Both PR and SR control the changes that can happen at each iteration step and are adjustable for efficiency of the method. Higher values can speed up the optimization but convergence and optimum solution could be compromised.

The process of material re-distribution that takes place in stage 2) consists of both adding and removing the same amount of material from the design domain, and can be obtained with the following equation:

$$\Delta VF_{add}(i) = \Delta VF_{rem}(i) = B\cdot f \tag{6}$$

In (6) B is the material re-distribution fraction, with typical values ranging between 0.001 and 0.005. In order to remove or add the amount of volume defined for each iteration, threshold values $\alpha^{R}{}_{th}$ and $\alpha^{V}{}_{th}$ of sensitivity numbers are calculated (Fig. 3). Material redistribution stage only starts when the volume fraction constraint is accurately satisfied. Therefore, if the volume fraction of the structure differs by more than 0.001 from the target volume fraction, the Progression Rate value is reduced by 70% and stage 1) is repeated until the right volume fraction is reached.

2.4 Sensitivity analysis

The sensitivity number α_e in each element that determines which elements are removed or added so that the objective function gets minimized is obtained with the following expression (Alonso et al. 2013):

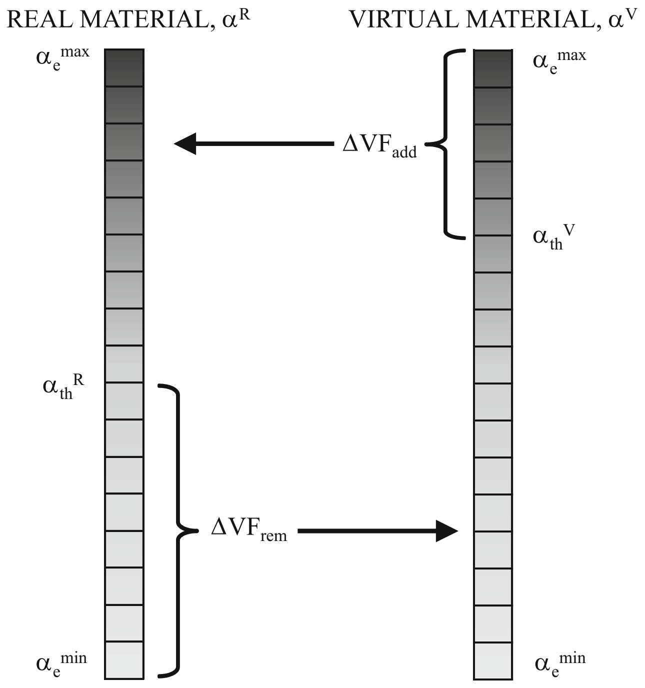

Fig. 3 Lists with sensitivity numbers of 'real' and 'virtual' materials

$$\alpha_e = -\mathbf{U}_e^T \Delta\mathbf{K}_e \mathbf{U}_e \tag{7}$$

where U_e is the displacement vector of element e and $\Delta \mathrm{K}_e$ is the variation of the elemental stiffness matrix. The sensitivity number for the eth finite element α_e (7) is a function of the variation between two iterations in the stiffness matrix of that element (8).

$$\Delta\mathbf{K}_e = \mathbf{K}_e(i) - \mathbf{K}_e(i-1) \tag{8}$$

where $\mathrm{K}_e(i)$ is the stiffness matrix in the ith iteration for the eth finite element and $\mathrm{K}_e(i-1)$ is the stiffness matrix in the $(i-1)$th iteration for the same finite element. If an element is added, $\mathrm{K}_e(i) = \mathrm{K}_e$ and $\mathrm{K}_e(i-1) \approx 0$, so the variation of the elemental stiffness matrix is $\Delta\mathrm{K}_e = \mathrm{K}_e$. But if an element is removed, $\mathrm{K}_e(i) \approx 0$ and $\mathrm{K}_e(i-1) \approx \mathrm{K}_e$, so $\Delta\mathrm{K}_e = -\mathrm{K}_e$. Therefore the elemental sensitivity numbers for the ‘real’ and ‘virtual’ material are given by (9) and (10), respectively:

$$a_e^R = \mathbf{U}_e^T \mathbf{K}_e \mathbf{U}_e \tag{9}$$

$$a_e^V = -\mathbf{U}_e^T \mathbf{K}_e \mathbf{U}_e \tag{10}$$

As the objective is to minimize the compliance of the structure, the elements with the lower values of sensitivity number are the ones to be added and removed. It must be noted that the addition of elements with lower values of sensitivities is equivalent to the addition of elements with higher values if the sign of the corresponding sensitivity number is inverted in (7). In this case, the sensitivity number can be calculated using the same (9) regardless of the material model, which simplifies its Matlab implementation. This material redistribution strategy and the lists of ‘real’ and ‘virtual’ material sensitivities are shown in Fig. 3.

2.5 Filtering of sensitivities and stopping criterion

The mesh independent filter is based on the technique proposed by Sigmund and Petersson (1998) and modifies the sensitivity number of each element based on a weighted average of the element sensitivities (11) in a fixed neighbourhood defined by a minimum radius $r_{\min}$:

$$\alpha'_e = \frac{\sum_{i=1}^{N} \rho_i \cdot \omega_i \cdot \alpha_i}{\rho_e \sum_{i=1}^{n} \omega_i} \quad \omega_i = r_{\min} - \mathbf{dist}(e,i), \quad \{i \in n / \mathbf{dist}(e,i) \le r_{\min}\} \tag{11}$$

where α_e is the eth element filtered sensitivity number, n is the number of elements which are inside of the filter radius, ρ_i is the density of element i. The weighting factor ω_i for element i decreases linearly the further element i is from element e, and for all elements outside the filter radius its value is equal to zero. Finally, α_i is the ith element sensitivity value, $r_{\min}$ is the filter radius specified by the user and dist(e, i) represents the distance between the centers of both elements.

The stopping criterion is defined as the change in the objective function in the last 20 iterations (12), which is considered an adequate number of iterations for the convergence study. It implies that the process will have a minimum of 20 iterations as the criterion is not applied until the iteration number has reached 20.

$$\varepsilon_i = \frac{\sum_{i-19}^{i-10} c_i - \sum_{i-9}^{i} c_i}{\sum_{i-9}^{i} c_i} \tag{12}$$

where ε_i is the stopping criterion, with typical values ranging between 0.0001 and 0.001.

3 Matlab implementation

The main program of the code in Appendix can be called from the Matlab prompt with the following line

sera(nelx,nely,volfrac,rmin)

where nelx and nely are the number of elements in the horizontal and vertical directions, respectively, volfrac is the volume fraction and rmin represents the filter size. Additional variables like termination criteria, material model, boundary conditions and plot style are defined in the code itself and can be edited if needed. The input line is similar to the original 99 line code by Sigmund (2001), except that the penalization power can be ignored in this case. These details of the Matlab code are discussed in the following subsections and several examples are shown in section 4 to demonstrate the application of the code with its extension to multiple load cases, passive elements and compliant mechanism design.

3.1 Main program

The main program starts initializing the iteration number and the variable that takes account of the change in the objective function. Then a full rectangular design domain (initially all elements are “real” material) with nelx elements in the horizontal direction and nely elements in the vertical direction is defined. The Progression Ratio PR, the Smoothing Ratio SR, and the re-distribution ratio B are set at the beginning of the main program as well. The main loop stars calculating the target volume fraction of the next iteration and the finite element subroutine is called in line 12. This function returns the displacement vector U, the global stiffness matrix K and the element stiffness matrix Ko, which is the same for all ‘real’ material elements. The displacement vector and the global stiffness matrix are used to determine the objective function (line 13) and sensitivity numbers for all elements are obtained extracting the element displacement vector Ue from the global displacement vector U (line 19). Following this loop there is a call to the mesh-independency filter (line 23) and the SERA optimization algorithm (line 25). Finally, in line 27 a new volume fraction is calculated for the current iteration and the change in the objective function is obtained (line 30). The intermediate results are printed and plotted (lines 33–36) in the same way as in the previously cited 99 line code. The change in the objective function and the number of added and removed elements are also displayed. The optimization loop is terminated if the relative change of the objective function in the last 20 iterations is less than 0.0001 (variable Change is computed in line 30), otherwise above steps are repeated. The proposed code includes additional lines as described in the 99 line code to account for different boundary conditions, multiple load cases, and passive elements. Load and support conditions are changed in chapter 4 in order to solve several optimization problems. Finally the compliance minimization problem is converted to a compliant mechanism synthesis case by changing a few lines of the code.

The finite element routine included in the 99-line code may be indeed slow for very fine discretizations, since it uses a sparse assembly strategy that is easy to read but highly inefficient for large problem. This code was much improved for speed in the 88-line code by Andreassen et al. (2011). In this paper, the analysis routine in the 99-line has been replaced by a very efficient code where assembly is also performed using a list of triplets (Nobel-Jørgensen and Bærentzen 2016). A dramatic speed-up is obtained with this substitution.

3.2 Optimization algorithm

Obviously the optimization algorithm is the most important difference as compared to the original 99 line paper, where the classical SIMP approach was used. The optimization loop begins in line 25 with a call to the SERA_Update function. The algorithm starts by sorting the "real" and "virtual" elements, whose sensitivities are stored in two different matrixes, alfa_R and alfa_V, respectively. The next step depends on the actual volume fraction of the design. If "real" material volume fraction is larger than the specified volume fraction *f*, the new target volume fraction DeltaV(i) is calculated as well as the number of elements to be removed, NumElem_Rem (line 114). If iteration counter is bigger than 2 the number of elements to add, NumElem_Add (line 117), is also obtained. If "real" material has already reached the specified volume fraction, material re-distribution takes place, where the same amount of material is added and rejected (lines 123–127). Before the redistribution loop starts, it is checked if the volume fraction constraint is accurately satisfied (line 121). When the volume fraction of the structure differs by more than 0.001 from the target volume fraction, material update is skipped and the iteration starts all over again after reducing the Progression Rate value by 70%, repeating the process until the right volume fraction is reached.

Independently of the actual material volume, the same subroutines are called in both stages to remove and add material (Update_R, defined in line 131 and Update_V, defined in line 138). The first subroutine sorts elements with 'real' material according to their sensitivity number from highest to lowest, and defines a threshold value (line 133) depending on the number of elements that should be removed (alfa_R_th). "Real" elements are transformed into 'virtual' if their sensitivity is lower than the previously defined threshold value. In order to avoid numerical issues and unsymmetrical results this condition is specified in relative terms (line 134). The function called Update_V works in the same manner, but transforming "virtual" elements into "real" when their sensitivity number is higher than the corresponding threshold value, alfa_V_th (line 140). The optimization loop terminates when the stopping criteria is met, which means that material re-distribution stage does not further improve the value of the objective function.

4 Numerical examples

The following examples demonstrate the application of the Sequential Rejection and Admission (SERA) method for minimum compliance problems and its extension to multiple load cases, passive elements and compliant mechanism design.

4.1 Minimum compliance

The proposed Matlab code was used to optimize the so-called MBB beam, which has been extensively studied in topology optimization (Fig. 4). The load is applied vertically in the upper left corner and there is a symmetric boundary condition along the left edge and the structure is supported horizontally in the lower right corner. The beam's dimensions are 60 × 20 elements and correspond to half of the structure. The volume fraction limit is 50% and two different filter radii were used: 1.5 and 3.0. Figure 5a and b show the structures obtained with the following inputs, respectively:

sera(60,20,0.5,1.4) **sera(60,20,0.5,3.5)**

It can be seen from the solutions shown in Fig. 5 that optimized topologies are comparable with the optimum designs obtained by other methods such as SIMP.

The next example is a short cantilever beam with the design domain, boundary conditions, and external load shown in Fig. 6a. The mesh employed consists of 30 × 20 elements and the filter radius equals 1.5 times the size of the finite

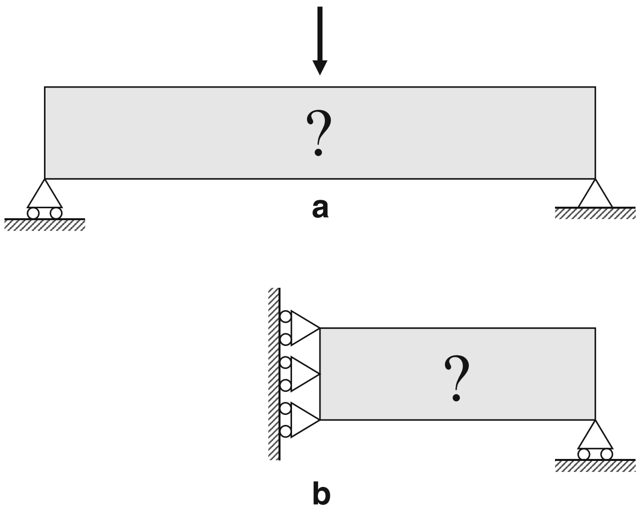

Fig. 4 Design model for the topology optimization of the MBB-beam

element, with the volume constraint set to 40%. The load is applied at the bottom right corner of the design domain while the left side is clamped. In order to solve this optimization problem boundary conditions specified in lines 67 and 68 should be changed in the following way:

```
67   F(2*(nely + 1)*(nelx + 1),1) = −1.0;
68   fixeddofs = [1:2*(nely + 1)];
```

The input line for this case is

sera(32,20,0.4,1.5)

The example in Fig. 6b corresponds to another cantilever beam but with a different aspect ratio. It must be noted that most of the examples shown in this paper present coarse discretizations to comply with the 99-line paper but for all practical applications much finer discretizations should be used. In this case the design domain is discretized with a much more refined mesh of 200 × 50 elements and the load is applied at the middle point of the right edge. The prescribed volume fraction is taken to be 0.55 and the radius filter is set to 1.4. Lines 67 and 68 have to be changed to:

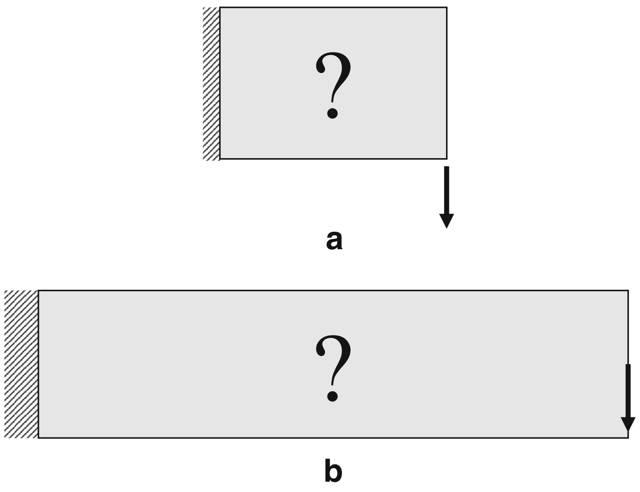

Fig. 6 Design models for short and long cantilever beams

```
67   F(2*(nely + 1)*(nelx + 1)-nely,1) = −1.0;
68   fixeddofs = [1:2*(nely + 1)];
```

This solution is obtained running the code with the following command line input:

sera(200,50,0.55,1.4)

Figure 7 shows the optimum topologies for both cantilever beams. Moreover, it can be seen in Fig. 7b that the algorithm succeeds in achieving a symmetric solution with respect to the horizontal axis of the beam.

4.2 Multiple load cases

Extending the algorithm to account for multiple load cases can be done by making minor changes to a few lines. If two load cases are considered, force and displacement vectors must become two-column vectors (lines 47 and 48):

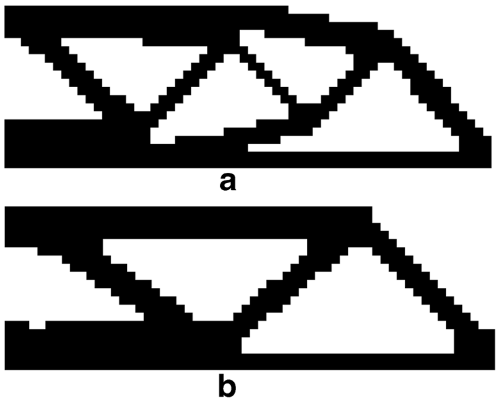

Fig. 5 Resulting topologies for the MBB-beam with different filtering

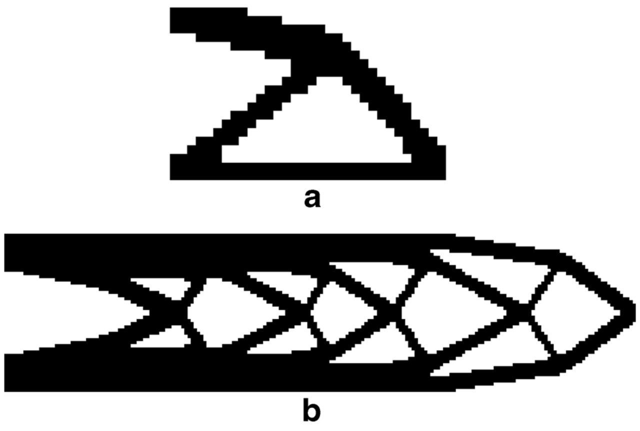

Fig. 7 Topology optimized cantilever beams

47 F = sparse(2*(nely + 1)*(nelx + 1),2);
48 U = zeros(2*(nely + 1)*(nelx + 1),2);

The objective function should be defined as the sum of compliances for each load case, i.e.

$$c(\boldsymbol{\rho}) = \omega_j \sum_{j=1}^{m} \mathbf{U}_j^T \mathbf{K} \mathbf{U}_j \tag{13}$$

where m is the number of load cases and w_j represents the weighting factor for each load case. The sum of compliances for two load cases using the same weight for both cases can be computed by modifying line 13 in the Matlab program:

13 c(i) = U(:,1)'*K* U(:,1) + U(:,2)'*K* U(:,2);

If only two load cases are considered, lines 17–19 must be substituted in the following way:

17b alfa(ely,elx) = 0.0;
17c for j = 1:2
18 Ue = U([2*n1–1;2*n1; 2*n2–1;2*n2;2*n2 + 1;2*n2 + 2;2*n1 + 1;2*n1 + 2],j);
19 a lfa(ely,elx) = alfa(ely,elx) + Ue'*Ko*Ue;
19b end.

To solve the two-load problem of the cantilever beam indicated in Fig. 8a the loading condition in line 67 is changed correspondingly and a unit upward load in the top-right corner is added:

67 F(2*(nelx + 1)*(nely + 1),1) = −1.0;
67b F(2*(nelx)*(nely + 1) + 2,2) = 1.0;
68 fixeddofs = [1:2*(nely + 1)];

This example is promoted by the line

sera(30,30,0.4,1.5)

Figure 8b shows the optimum design for the cantilever when topology is optimized considering only one load case and Fig. 8c corresponds to the two load case. It can be noticed that obtained topologies agree well with the solutions published in the paper by Sigmund (2001) using the SIMP method.

4.3 Passive elements

In some cases the designer may desire some elements to be void permanently. This can be done by defining the nely × nelx array, called passive, with zeros at elements free to change and ones at elements fixed to be always virtual. This array should be transferred to the SERA_Update function in line 25 and added to the list of parameters in line 105. The following line must be added also inside this subroutine after line 110.

110b alfa_V(passive > 0) = alfa_min;

This statement assigns the minimum value to the sensitivity numbers of 'virtual' elements included in the **passive** array. This way, passive elements are moved to the tail of the **alfa_V** vector so that they are never selected to change from 'virtual' to 'real'. Finally, the following line added after the SERA update subroutine looks for passive elements and sets their density equal to the minimum density:

25b x(find(passive)) = 1e-9;

To solve the problem shown in Fig. 9a, the following passive elements need to be defined after line 5, where a circle with radius **nely/3** and center in (**nely/2**, **nelx/3**) is considered as void region:

for ely = 1:nely

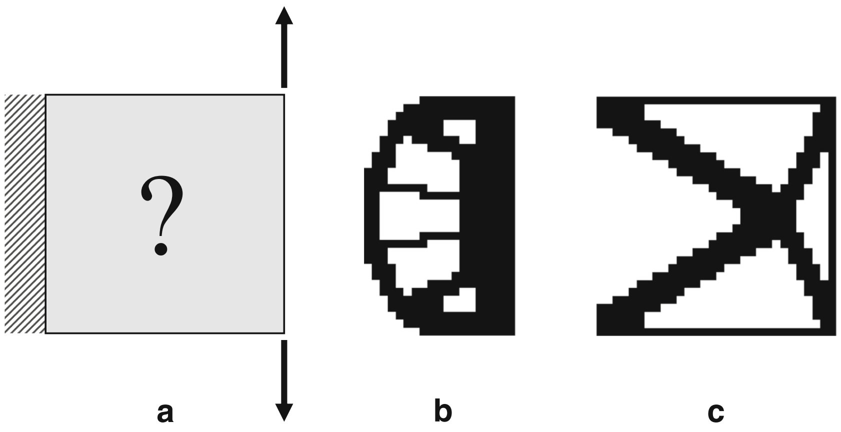

Fig. 8 Topology optimization of cantilever beam using one and two load cases

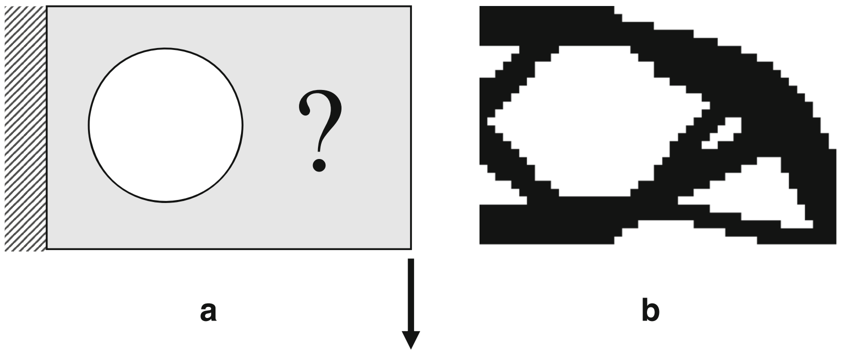

Fig. 9 Design domain and optimum topology for a cantilever beam with a fixed hole

```
for elx = 1:nelx
    if sqrt((ely-((nely + 0)/2))^2 + (elx-((nelx)/3))^2) < (nely/3)
      passive(ely,elx) = 1;
      x(ely,elx) = 1e-9;
      else
      passive(ely,elx) = 0;
    end
  end
end
```

The boundary conditions of the problem are the same as in the short cantilever example of chapter 4.1. Figure 9b shows the optimum structure that includes the fixed hole, obtained with the following input.

sera(45,30,0.5,1.5)

4.4 Compliant mechanisms

The Matlab code for compliance minimization given in the previous sections can be easily changed to solve compliant mechanisms topology optimization problems. Actually, fewer changes than in the original 99 line code need to be made. A compliant mechanism optimum design involves two loading cases: input loading case and dummy (or adjoint) loading case. The allocation of force and displacement vectors for 'real' and adjoint load cases is similar to the two load case problem of section 4.2. Instead of calculating the compliance of the structure, we will compute the output displacement, substituting line 13 with the following code (Saxena and Ananthasuresh 2000):

```
13     c(i) = U(:,1)'*K*U(:,2);
```

Sensitivities are obtained in terms of the solutions to the 'real' case and the adjoint loading case, which correspond to the first and second column of the displacement matrix **U**. The sign in the values of the sensitivities must be inverted as well, since we are trying to maximize the output displacement in this case, instead of minimizing the compliance. Therefore, lines 18 and 19 are substituted in the following way:

```
18     Ue1 = U([2*n1–1;2*n1; 2*n2–1;2*n2; 2*n2 +
1;2*n2 + 2;2*n1 + 1;2*n1 + 2],1);
18b    Ue2 = U([2*n1–1;2*n1; 2*n2–1;2*n2; 2*n2 +
1;2*n2 + 2;2*n1 + 1;2*n1 + 2],2);
19     alfa(ely,elx) = -Ue1'*Ko*Ue2;
```

Finally, we will define the boundary conditions and the input and dummy loads for the inverter problem considered in Fig. 10a. External springs are also added with a default value of 0.1 to the input and output points of the design domain. Input and output port degrees of freedom are labeled as **din** and **dout**, respectively. Below the necessary changes to the Matlab code are listed, in order to define the necessary boundary conditions of the inverter problem for topology optimization of compliant mechanisms design (lines 67 and 68 should be substituted with the following code):

```
din = 1;
dout = 2*nelx*(nely + 1) + 1;
K(din,din) = K(din,din) + 0.1;
K(dout,dout) = K(dout,dout) + 0.1;
F(din,1) = 1;
F(dout,2) = −1;
fixeddofs = union([2:2*(nely + 1):2*(nely + 1)*(nelx +
1)],[2*(nely + 1):-1:2*(nely + 1)-3]);
```

The progression, smoothing and material redistribution ratios in the SERA algorithm may need to be adjusted in some cases to stabilize and improve convergence of the problem. The convergence time and final topologies of compliant mechanisms are more sensitive to the values of these parameters than the maximum stiffness structures (Alonso et al. 2013). Anyhow, if problems

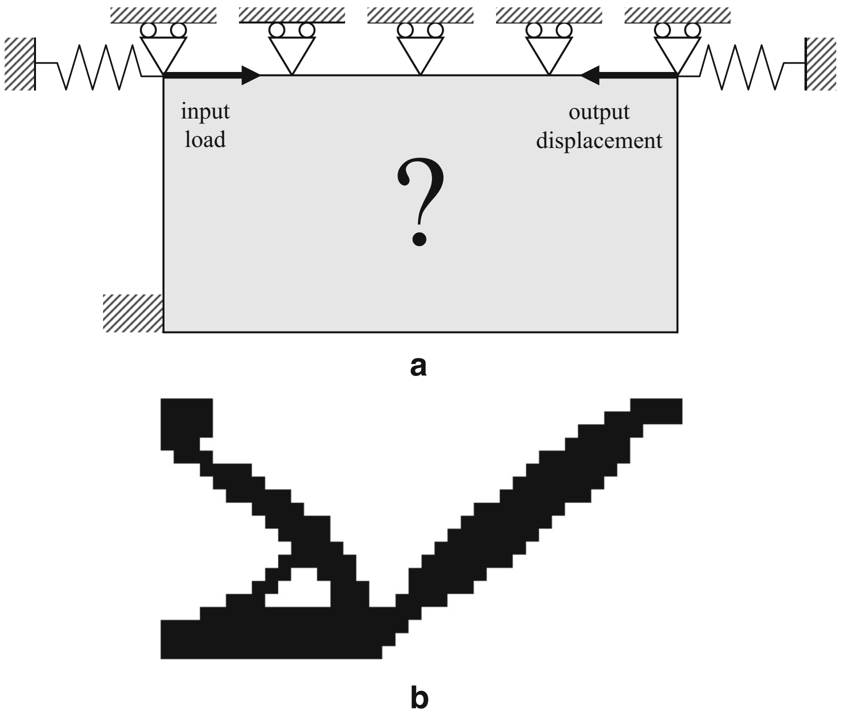

Fig. 10 Half design domain and solution of the inverter compliant mechanism

were encountered it should usually be enough to decrease the value of the progression ratio *PR* by approximately 0.01 and the smoothing ratio *SR* by no more than 0.1.

Figure 10b corresponds to the optimum material distribution in the inverter problem when the Matlab code is called with the following input line:

sera(40,20,0.3,1.1)

5 Conclusions

This paper describes the numerical implementation in Matlab of the Sequential Rejection and Admission (SERA) method for minimum compliance problems and its extension to multiple load cases, passive elements and compliant mechanism design. The main difference of this method with respect to other bi-directional methods which add and remove elements from the design domain is that 'real' and 'virtual' materials are treated separately so that the addition and removal of elements have separate criteria. The code is inspired by the work presented by Sigmund (2001) and its objective is to provide students with an academic implementation of the SERA topology optimization strategy. The main differences with respect to the original 99 line code are the material rejection, addition and re-distribution subroutines that substitute the classical optimality criteria optimizer, where special care has been taken not to affect the readability of the code. The use of the implemented code is demonstrated through several numerical examples. We hope that the code given in the appendix will be useful for students and professors working in the area of structural optimization.

Acknowledgements This work was supported by The European Regional Development Fund (ERDF-FEDER) and the Ministry of Education and Science in Spain through the DPI2015-64863-R project (MINECO/FEDER-UE). The authors also wish to thank the Basque Government for financial assistance through IT919-16.

Appendix - Matlab code

```
%%%%%%%%%% SERA TOPOLOGY OPTIMIZATION CODE %%%%%%%%%%
function sera(nelx,nely,volfrac,rmin)
% INITIALIZE
i=1;Change=1;
x(1:nely,1:nelx)=1.0;
VF(i)=ceil(sum(sum(x)))/(nelx*nely);
PR = 0.03;SR=1.3;B=0.003;
% START ITERATION
while Change>0.0001
  i=i+1;
  VF(i)=max(VF(i-1)*(1-PR),volfrac);
  [U,Ko,K]=FE(nelx,nely,x);
  c(i)=U'*K*U;
  for ely=1:nely
   for elx=1:nelx
    n1=(nely+1)*(elx-1)+ely;
    n2=(nely+1)*elx+ely;
    Ue=U([2*n1-1;2*n1;2*n2-1;2*n2;2*n2+1;2*n2+2;2*n1+1;2*n1+2],1);
    alfa(ely,elx)=Ue'*Ko*Ue;
   end
  end
% FILTERING TECHNIQUE
  [alfa]=Filter(nelx,nely,rmin,x,alfa);
% DESIGN UPDATE
  [x,NumElem_Add,NumElem_Rem,PR]=SERA_Update(nelx,nely,alfa,x,VF,
  volfrac,i,SR,B,PR);
  VF(i)=ceil(sum(sum(x)))/(nelx*nely);
% STOPPING CRITERION
  if i>20
   Change=abs((sum(c(i-19:i-10))-sum(c(i-9:i)))/sum(c(i-9:i)));
  end
% PRINT RESULTS
  disp([' Iteration: ' sprintf('%4i', (i-1)) ' Volume fraction: '
  sprintf('%6.4f', VF(i)) ' Compliance: ' sprintf('%6.6f',c(i)) '
  Change: ' sprintf('%7.5f', Change) ' Removed: ' sprintf('%5i',
  NumElem_Rem) ' Added: ' sprintf('%5i', NumElem_Add)])
% PLOT DENSITIES
  colormap(gray); imagesc(-x); axis equal; axis tight;
  axis off; pause(1e-6)
end
%%%%%%%%%%  FE-ANALYSIS  %%%%%%%%%%
function [U,Ko,K]=FE(nelx,nely,x)
[Ko] = lk;
I=zeros(nelx*nely*64,1);
J=zeros(nelx*nely*64,1);
X=zeros(nelx*nely*64,1);
F=sparse(2*(nely+1)*(nelx+1),1);
U=zeros(2*(nely+1)*(nelx+1),1);
ntriplets=0;
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx +ely;
```

```
      edof = [2*n1-1 2*n1 2*n2-1 2*n2 2*n2+1 2*n2+2 2*n1+1 2*n1+2];
      for krow = 1:8
        for kcol = 1:8
          ntriplets = ntriplets+1;
          I(ntriplets) = edof(krow);
          J(ntriplets) = edof(kcol);
          X(ntriplets) = x(ely,elx)*Ko(krow,kcol);
        end
      end
    end
  end
  K=sparse(I,J,X,2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1));
  % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
  F(2,1) = -1;
  fixeddofs = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
  alldofs = [1:2*(nely+1)*(nelx+1)];
  freedofs = setdiff(alldofs,fixeddofs);
  % SOLVING
  U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
  U(fixeddofs,:) = 0;
  %%%%%%%%%%  ELEMENT STIFFNESS MATRIX  %%%%%%%%%%
  function [Ko] = lk
  E = 1; nu = 0.3;
  k=[1/2-nu/6;1/8+nu/8;-1/4-nu/12;-1/8+3*nu/8;
  -1/4+nu/12;-1/8-nu/8;nu/6;1/8-3*nu/8];
  Ko = E/(1-nu^2)*
  [k(1),k(2),k(3),k(4),k(5),k(6),k(7),k(8);
   k(2),k(1),k(8),k(7),k(6),k(5),k(4),k(3);
   k(3),k(8),k(1),k(6),k(7),k(4),k(5),k(2);
   k(4),k(7),k(6),k(1),k(8),k(3),k(2),k(5);
   k(5),k(6),k(7),k(8),k(1),k(2),k(3),k(4);
   k(6),k(5),k(4),k(3),k(2),k(1),k(8),k(7);
   k(7),k(4),k(5),k(2),k(3),k(8),k(1),k(6);
   k(8),k(3),k(2),k(5),k(4),k(7),k(6),k(1)];
  %%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%
  function [alfaNew]=Filter(nelx,nely,rmin,x,alfa)
  alfaNew=zeros(nely,nelx);
  for i = 1:nelx
    for j= 1:nely
     sum=0.0;
     for k = max(i-floor(rmin),1) : min(i+floor(rmin),nelx)
      for l = max(j-floor(rmin),1) : min(j+floor(rmin),nely)
       sum = sum+max(0,rmin-sqrt((i-k)^2+(j-l)^2));
       alfaNew(j,i) = alfaNew(j,i) +
       max(0,rmin-sqrt((i-k)^2+(j-l)^2))*x(l,k)*alfa(l,k);
      end
     end
     alfaNew(j,i) = alfaNew(j,i)/x(j,i)/sum;
    end
  end
  %%%%%%%%%% SERA UPDATE %%%%%%%%%%
  function [x,NumElem_Add,NumElem_Rem,PR]=SERA_Update(nelx,nely,
  alfa,x,VF,volfrac,i,SR,B,PR)
  alfa_min=min(min(alfa));alfa_max=max(max(alfa));
  alfa_V=alfa;alfa_R=alfa;
  alfa_R(x<1.0)=alfa_max; alfa_V(x>1e-9)=alfa_min;
  NumElem_Add=0;NumElem_Rem=0;
  if VF(i)>volfrac
    DeltaV(i)=abs(VF(i)-VF(i-1));
    DeltaV_Rem=DeltaV(i)*(SR);
    NumElem_Rem=max(1,floor(nelx*nely*DeltaV_Rem));
    [x,NumElem_Rem]=Update_R(nelx,nely,x,alfa_R,NumElem_Rem);
    if i>2
      NumElem_Add=max(1,floor(NumElem_Rem*(SR-1)));
      [x,NumElem_Add]=Update_V(nelx,nely,x,alfa_V,NumElem_Add);
    end
  else
    if (VF(i-1)-volfrac)>=0.001 PR=PR*0.7;
    else
      DeltaV_Rem=B*volfrac;
      NumElem_Rem=max(1,floor(nelx*nely*DeltaV_Rem));
      [x,NumElem_Rem]=Update_R(nelx,nely,x,alfa_R,NumElem_Rem);
      NumElem_Add=NumElem_Rem;
      [x,NumElem_Add]=Update_V(nelx,nely,x,alfa_V,NumElem_Add);
    end
  end
  %%%%%%%%%% UPDATE_R %%%%%%%%%%
  function[x,NumElem_Rem]=Update_R(nelx,nely,x,alfa_R,NumElem_Rem)
  alfa_R_vec=sort(reshape(alfa_R,(nelx*nely),1),'descend');
  alfa_R_th=alfa_R_vec((nelx*nely)-NumElem_Rem,1);
  Elem_Rem=((alfa_R-alfa_R_th)/abs(alfa_R_th))<1e-6;
  NumElem_Rem=sum(sum(Elem_Rem));
  x(Elem_Rem)=1e-9;
  %%%%%%%%%% UPDATE_V %%%%%%%%%%
  function [x,NumElem_Add]=Update_V(nelx,nely,x,alfa_V,NumElem_Add)
  alfa_V_vec=sort(reshape(alfa_V,(nelx*nely),1), 'descend');
  alfa_V_th=alfa_V_vec(NumElem_Add,1);
  Elem_Add=((alfa_V-alfa_V_th)/abs(alfa_V_th))>-1e-6;
  NumElem_Add=sum(sum(Elem_Add));
  x(Elem_Add)=1.0;
```

References

Allaire G (2012) http://www.cmap.polytechnique.fr/~allaire/levelset_en.html

Alonso C, Ansola R, Querin OM (2014a) Topology synthesis of multi-material compliant mechanisms with a sequential element rejection and admission method. Finite Elem Anal Des 85:11–19

Alonso C, Ansola R, Querin OM (2014b) Topology synthesis of multi-input multi-output compliant mechanisms. Adv Eng Softw 76:125–132

Alonso C, Querin OM, Ansola R (2013) A sequential element rejection and admission (SERA) method for compliant mechanisms design. Struct Optim 47:795–807

Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in matlab using 88 lines of code. Struct Multidiscip Optim 43:1–16

Beckers M (1999) Topology optimization using a dual method with discrete variables. Struct Optim 17:14–24

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

Bendsoe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71:197–224

Borrvall T (2001) Topology optimization of elastic continua using restriction. Arch Comput Meth Eng 8(4):351–385

Brodie RN (2007) Development of controllability and robustness methodologies for bi-directional evolutionary structural optimisation (BESO). PhD thesis, University of Leeds

Challis VJ (2010) A discrete level-set topology optimization code written in matlab. Struct Multidiscip Optim 41(3):453–464

Haber RB, Bendsoe MP, Jog CS (1996) A new approach to variable topology shape design using constraint on the perimeter. Struct Optim 11:1–12

Hajela P, Lee E (1995) Genetic algorithms in truss topological optimization. Int J Solids Struct 22:3341–3357

Huang X, Xie YM (2010) A further review of ESO type methods for topology optimization. Struct Multidiscip Optim 41:671–683

Liu K, Tovar A (2014) An efficient 3D topology optimization code written in Matlab. Struct Multidiscip Optim 50:1175–1196

Michell AGM (1904) The limits of economy of material in frame structures. Philos Mag 8:589–597

Nobel-Jørgensen M, Bærentzen JA (2016) Interactive topology optimization. Lyngby: Technical University of Denmark (DTU). (DTU Compute PHD-2015; No. 375)

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

Querin OM (1997) Evolutionary structural optimization: stress based formulation and implementation. Ph.D. Thesis, University of Sydney

Querin OM, Steven GP, Xie YM (1998) Evolutionary structural optimization (ESO) using a bidirectional algorithm. Eng Comput 15:1031–1048

Rozvany G (2009) A critical review of established methods of structural topology optimization. Struct Multidiscip Optim 37:217–237

Rozvany GIN, Querin OM (2002a) Combining ESO with rigorous optimality criteria. Int J Veh Des 28(4):294–299

Rozvany GIN, Querin OM (2002b) Theoretical foundations of Sequential Element Rejections and Admissions (SERA) methods and their computational implementations in topology optimisation. In: Proceedings of the 9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, Atlanta, GA, 4.–6. September 2002

Rozvany GIN, Querin O (2004) Sequential element rejections and admissions (SERA) method: applications to multiconstraint problems. In: Proceedings of the 10th AIAA/ISSMO Multidisc. Anal. Optim. Conference, Albany, New York. https://doi.org/10.2514/6.2004-4523

Saxena A, Ananthasuresh G (2000) On an optimal property of compliant topologies. Struct Multidiscip Optim 19:36–49

Sethian JA, Wiegman A (2000) Structural boundary design via level set and immersed interface methods. J Comp Physiol 163:489–528

Sigmund O (1994) Design of material structures using topology optimization. Ph.D. Thesis, University of Denmark

Sigmund O (2001) A 99 line topology optimization code written in Matlab. Struct Multidiscip Optim 21:120–127

Sigmund O, Maute K (2013) Topology optimization approaches. A comparative review. Struct Multidiscip Optim 48:1031–1055

Sigmund O, Petersson J (1998) Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. Struct Optim 16:68–75. https://doi.org/10.1007/BF01214002

Sokolowski J, Zochowski A (1999) On the topological derivative in shape optimization. SIAM J Control Optim 37:1251–1272

Suresh K (2010) A 199-line matlab code for pareto-optimal tracing in topology optimization. Struct Multidiscip Optim 42:665–679

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

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 45:329–357

Xie YM, Steven GP (1993) A simple evolutionary procedure for structural optimization. Comput Struct 49:885–896

Xie YM, Steven GP (1997) Evolutionary structural optimization evolutionary structural optimization. Springer, New York

Yang XY, Xie YM, Steven GP, Querin OM (1999) Bi-directional evolutionary method for stiffness optimisation. AIAA J 37:1483–1488

Zegard T, Paulino GH (2014) GRAND - ground structure based topology optimization for arbitrary 2D domains using MATLAB. Struct Multidiscip Optim 50:861–882

Zegard T, Paulino GH (2015) GRAND3 - ground structure based topology optimization for arbitrary 3D domains using MATLAB. Struct Multidiscip Optim 52:1161–1184

Zegard T, Paulino GH (2016) Bridging topology optimization and additive manufacturing. Struct Multidiscip Optim 53:175–192

Zhou M, Rozvany GIN (2001) On the validity of ESO type methods in topology optimization. Struct Multidiscip Optim 21:80–83