Structural and Multidisciplinary Optimization
https://doi.org/10.1007/s00158-018-1904-8

EDUCATIONAL ARTICLE

# An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions

Peng Wei[1] · Zuyu Li[2] · Xueping Li[1] · Michael Yu Wang[3]

Received: 30 September 2016 / Revised: 31 December 2017 / Accepted: 5 January 2018


**Abstract**
This paper presents a compact and efficient 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions (RBFs), which is applied to minimize the compliance of a two-dimensional linear elastic structure. This parameterized level set method using radial basis functions can maintain a relatively smooth level set function with an approximate re-initialization scheme during the optimization process. It also has less dependency on initial designs due to its capability in nucleation of new holes inside the material domain. The MATLAB code and simple modifications are explained in detail with numerical examples. The 88-line code included in the appendix is intended for educational purposes.

**Keywords** Topology optimization · Level set method · Radial basis functions · MATLAB code

## 1 Introduction

This paper presents an 88-line MATLAB code for implementation of the parameterized level set method based topology optimization using Radial Basis Functions (RBFs) for compliance minimization problems. The 88-line MATLAB code is intended for educational purposes to help those new to the field of structural topology optimization as well as those who have only used other topology optimization approaches before.

To the authors' knowledge, there are a number of free codes for topology optimization of continuum structures, such as the classical 99-line MATLAB code of the Solid Isotropic Material/Microstructure with Penalization (SIMP) method (Sigmund 2001), which has been included in the lecture notes of structural optimization courses in many universities. Based on the level set method, the first compact open source code can be considered as the 199-line MATLAB code for the conventional level set method based topology optimization problems (Wang et al. 2004). Besides the scientific programming codes (e.g. by MATLAB), the combination of scripting languages with specific commercial software could be an alternative for achieving the purpose. For instance, a 100-line Python code has been integrated into the Abaqus Scripting Interface for solving three-dimensional topology optimization problems on the basis of the Bi-Directional Evolutionary Structural Optimization (Zuo and Xie 2015). Table 1 presents a number of such implementations for the sake of reading convenience. More details about the current development of various topology optimization approaches, especially the level set-based methods, can refer to the following review articles (Burger and Osher 2005; Gain and Paulino 2013; Dijk et al. 2013; Sigmund and Maute 2013; Deaton and Grandhi 2014).

The present 88-line MATLAB code can be used to implement the parameterized level set topology optimization method by using RBFs, which was first developed by (Wang and Wang 2006a; Wang et al. 2007) and was also called the WW approach in (Gain and Paulino 2013). Compared to the conventional level set method by directly solving either a Hamilton-Jacobi (e.g. Wang et al. 2004; Allaire 2009) or a reaction diffusion (Otomori et al. 2014) equations, the

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s00158-018-1904-8) contains supplementary material, which is available to authorized users.

✉ Peng Wei
ctpwei@scut.edu.cn

[1] State Key Laboratory of Subtropical Building Science, School of Civil Engineering and Transportation, South China University of Technology, Guangzhou, China

[2] Architecture and Civil Engineering Institute, Guangdong University of Petrochemical Technology, Maoming, China

[3] Department of Mechanical and Aerospace Engineering, Hong Kong University of Science and Technology, Hong Kong, China

Published online: 13 February 2018

**Table 1** A list of the educational codes for topology optimization in chronological order

| Year | Authors | Language | Lines | Method | Remarks |
|---|---|---|---|---|---|
| 2001 | Sigmund | MATLAB | 99 | SIMP | Minimum mean compliance problem |
| 2003 | Bendsøe and Sigmund | MATLAB | 105 | SIMP | Compliant mechanism design |
| 2003 | Bendsøe and Sigmund | MATLAB | 91 | SIMP | Heat conduction problem |
| 2004 | Wang et al. | MATLAB | 199 | Level set method | Minimum mean compliance problem |
| 2004 | Kharmanda et al. | MATLAB | 71 | SIMP | Reliability constraints |
| 2006 | Olesen et al. | FEMLAB | 111 | Density based method | Steady-state Navier-Stokes flow problem |
| 2009 | Allaire | Scilab | – | Level set method | Minimum mean compliance problem |
| 2010 | Huang and Xie | MATLAB | 101 | BESO | Minimum mean compliance problem |
| 2010 | Suresh | MATLAB | 199 | SIMP | Pareto-optimal tracing |
| 2010 | Challis | MATLAB | 129 | Level set method | Minimum mean compliance problem |
| 2011 | Andreassen et al. | MATLAB | 88 | SIMP | A successor to the 99- line code |
| 2011 | Schmidt and Schulz | C/C++ | 2589 | SIMP | GPU acceleration |
| 2012 | Talischi et al. | MATLAB | 190 | SIMP | Isoparametric polygonal elements |
| 2012 | Zhou et al. | MATLAB | 99 | BESO | Biphasic cellular materials with desirable t ransport properties |
| 2014 | Liu and Tovar | MATLAB | 169 | SIMP | 3D minimum mean compliance problem |
| 2014 | Otomori | MATLAB | 88 | Level set method | Reaction diffusion equation |
| 2014 | Tavakoli and Mohseni | MATLAB | 115 | Alternating active-phase algorithm | Multi-material topology optimization problem |
| 2015 | Emre and To | MATLAB | 90 | PTO | Proportional topology optimization |
| 2015 | Xia and Breitkopf | MATLAB | 119 | Homogenization method | Design of materials with extreme properties |
| 2015 | Zegard and Paulino | MATLAB | – | Ground structure | Truss optimization for arbitrary 3D domains |
| 2015 | Zuo and Xie | Python | 100 | BESO | Abaqus Scripting Interface |
| 2016 | Zhang et al. | MATLAB | 188 | MMC | Moving Morphable Components |

parameterized level set method (PLSM) evolves boundaries by updating a set of parameterized coefficients at grid points. The PLSM has a number of advantages. Firstly, it is capable of avoiding some numerical manipulations caused by solving the Hamilton-Jacobi equation with a finite difference scheme. Secondly, an approximate re-initialization scheme can be applied to the method, in order to improve the gradient of the level set function around the interface as well as to preserve the ability of nucleation of internal holes. Thirdly, smooth boundaries can be obtained without implementing any filter or additional smoothing schemes. Furthermore, the method can be used for optimizations of irregular design domains of unstructured meshes without difficulty (Cecil et al. 2004).

In this study, the code is used for solving a number of 2D compliance minimization problems, to verify the effectiveness and convergence speed of the method. However, it is straightforward to extend the present code for any 3D problems if the storage and computational cost of the RBF parameterization implementation can be effectively reduced (see discussions in Wang and Wang 2006a).

The remainder of the paper is organized as follows. The level set method based topology optimization using RBFs for compliance minimization problems are described in Section 2. Concrete code implementation details and a numerical example are provided in Section 3. Extensions of the code and numerical examples are given in Section 4. Conclusions are drawn in Section 5. And the 88-line MATLAB code is provided in the Appendix.

## 2 Theoretical backgrounds

### 2.1 The conventional level set method

The level set method was firstly proposed by Osher and Sethian (1988) to deal with the front tracking problem and later introduced into structural optimization by Sethian and Wiegmann (2000) and Osher and Santosa (2001). After that, the level set-based topology optimization method combined with the shape derivative is proposed (Wang et al. 2003; Allaire et al. 2004). Compared to the SIMP and ESO/BESO methods, the main attractive feature of the level set method is that it can always provide a clear boundary and geometry information during the optimization process, and therefore it has inherent advantages when solving boundary and geometry related problems. Also, it has great potential to be incorporated into CAD systems. Furthermore, benefitting from the implicit representation, level set method can easily and naturally handle complex shape and topology change such as boundary splitting and merging.

In the level set method, the dynamic structural interfaces are implicitly embedded as the zero level set of a higher-dimensional level set function $\phi(\boldsymbol{x}, t)$, which is usually defined as follows.

$$\begin{cases} \phi(\boldsymbol{x}, t) > 0 & \forall \boldsymbol{x} \in \Omega \backslash \partial\Omega \\ \phi(\boldsymbol{x}, t) = 0 & \forall \boldsymbol{x} \in \partial\Omega \\ \phi(\boldsymbol{x}, t) < 0 & \forall \boldsymbol{x} \in \mathrm{D} \backslash \Omega \end{cases} \tag{1}$$

where $\boldsymbol{x} \in \mathrm{D} \subset \{(x, y) | \, x, y \in \mathbb{R}\}$ is any point in the full design domain D and $\partial\Omega$ is the boundary of the solid domain $\Omega$ as shown in Fig. 1 for a 2-D case.

The following evolution equation is used to update the level set function in the conventional level set method:

$$V_{\mathrm{n}} = \boldsymbol{V} \cdot \left( -\frac{\nabla\phi}{|\nabla\phi|} \right) \tag{2}$$

这里的V应该不是个常数值，应该是dx/dt

$$\frac{\partial\phi}{\partial t} - V_{\mathrm{n}} |\nabla\phi| = 0 \tag{3}$$

where $\nabla(\cdot)$ denotes the gradient of a scalar function, $t$ is the pseudo time representing the evolution of the level set function, and $V_{\mathrm{n}} = V_{\mathrm{n}}(\boldsymbol{x}, t)$ is the normal velocity towards outside chosen based on the shape derivative of an optimization problem.

The conventional level set method requires appropriate choice of finite difference methods on a fixed Cartesian grid to solve (3), which is a Hamilton-Jacobi Partial Differential Equation (PDE). A general conventional level set updating scheme involves upwind differencing scheme, re-initialization, velocity extension etc. (Sethian 1999; Osher and Fedkiw 2002).

The time step size also should be sufficiently small to satisfy the Courant-Friedrichs-Lewy (CFL) condition for numerical stability (Sethian 1999; Osher and Fedkiw 2002). It indicates that the largest time step cannot be larger than the ratio of the minimum grid interval to the magnitude of the velocity. Theoretically, in structural optimization problems, the CFL condition may limit the numerical step size. But in practical implementation, because the updating of the level set function in an explicit numerical scheme is much cheaper than the finite element analysis, the level set function can be updated several times with one finite element analysis to overcome this limitation (Allaire et al. 2004; Dijk et al. 2013).

In the conventional strategy, the level set function should be reinitialized frequently to maintain a signed distance function, which gives the shortest distance to the nearest point on the interface, by using a PDE-based method (Peng et al. 1999) or a fast marching method (Sethian 1999; Osher and Fedkiw 2002). The re-initialization is an important process in the conventional level set method to keep the norm of the gradient of the level set function constant and make the evolution stable.

Another issue of the conventional level set method is lacking of the capacity to create new holes inside the material domain, which makes it relatively easy to get stuck at a local minimum. A general way to overcome this weakness is to put sufficient number of holes in the initial design to keep the complexity of the structural topology since the level set method has no difficulty in handling topology change by merging holes (Wang et al. 2003; Allaire et al. 2004). Another way is to incorporate nucleation strategy, e.g. the topological derivative (Sokołowski and Żochowski 1999), to create new holes by removing the least useful material during the optimization iterations (Burger et al. 2004; Mei and Wang 2004; Allaire et al. 2005). Recently, some variational level set methods have been developed with nucleation capacity to alleviate the dependency of the final solution on the initial design (Wang and Wang 2006a; Wei and Wang 2009; Yamada et al. 2010).

More detailed discussions about the conventional level set method can be found in (Sethian 1999; Osher and Fedkiw 2002) and in (Van Dijk et al. 2013; Deaton & Grandhi 2014) for structural topology optimization.

## 2.2 Parameterized level set method (PLSM) using RBFs

Wang and Wang (2006a) and Wang et al. (2007) developed an effective approach for shape and topology optimization by introducing the RBFs into the level set method. Therein, a level set function is decoupled by a linear combination of a set of RBFs and coefficients. Since RBFs are only spatial coordinate-related, evolution of the level set function is transformed to the updating of the coefficients. This method has

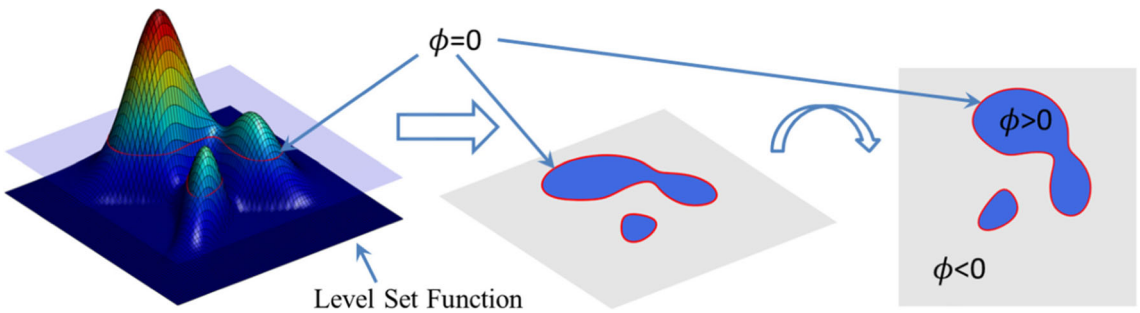



**Fig. 1** The description of a level set function for a 2-D problem

fewer numerical manipulations to handle evolution of level set functions. The parameterized modeling with RBFs can maintain a relatively smooth level set function without re-initialization operation during the optimization process. By applying the natural velocity extension, dependency on initial designs can be largely alleviated due to its capability in nucleation of new holes inside the material domain. However, Gain and Paulino (2013) claimed that the method may generate level set functions of quite high absolute values and then cause the convergence problems. To address the issue, here we propose an approximate re-initialization operation to prevent the level set function from becoming too flat or steep and also to apply a delta function to prevent the values of level set function approaching infinite large.

In the following part, we give a brief introduction about this approach. Radial basis functions are a set of real-valued axial symmetric functions (the value dependents only on the distance to the origin) to be used as basis functions to approximate a given function, e.g. the level set function in this study. There are many types of popular RBFs such as thin-plate splines, Gaussians, Compactly Supported Radial Basis Functions (CSRBFs), MultiQuadric (MQ) splines and Inverse MultiQuadric (IMQ) splines. The MQ spline is used here to construct the level set function. The readers who are interested in other RBFs can edit the code to replace the formulations accordingly.

The MQ spline can be written as:

$$g_i(\boldsymbol{x}) = \sqrt{(\boldsymbol{x}-\boldsymbol{x}_i)^2 + c^2}, \quad \boldsymbol{x}_i \in D \tag{4}$$

where $\boldsymbol{x}_i = (x_i, y_i)$ is the coordinates of the $i$th knot and $c$ is the corresponding shape parameter which is commonly assumed to be a constant with a small value for all knots in most applications. This formula is calculated on lines 14–16 of the MATLAB code. The MQ spline and its partial derivative in $x$ direction at $\boldsymbol{x}_i = (0, 0)$ are displayed in Fig. 2.

The parameterized level set function $\phi(\boldsymbol{x}, t)$ is interpolated by a number $n$ of MQ splines, centered at $n$ fixed knots, which can be written as:

$$\phi(\boldsymbol{x}, t) = \sum_{i=1}^{n} \alpha_i(t) g_i(\boldsymbol{x}) + p(\boldsymbol{x}, t) \tag{5}$$

where $\alpha_i(t)$ is the expansion coefficient of the MQ spline positioned at the $i$th knot changed with the pseudo time $t$, and $p(\boldsymbol{x}, t)$, which is not necessary for some RBFs, a first-degree polynomial changed with time to account for the linear and constant portion of $\phi(\boldsymbol{x}, t)$ and to ensure positive definiteness of the solution (Morse et al. 2001).

For 2D modeling problems, $p(\boldsymbol{x}, t)$ can be given by:

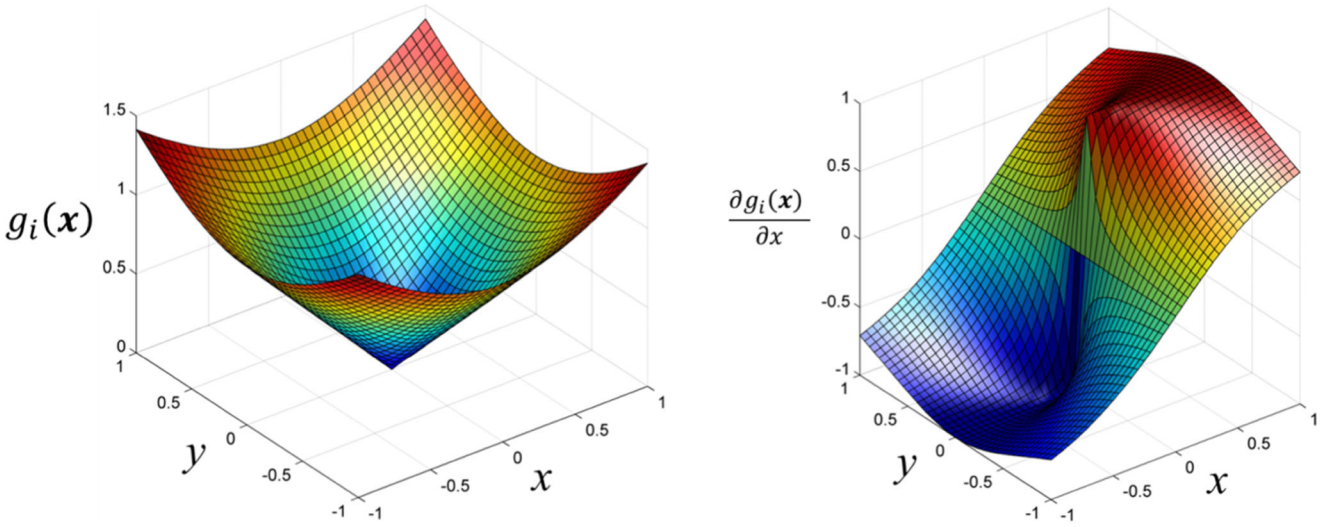
$$p(\boldsymbol{x}, t) = p_0(t) + p_1(t)x + p_2(t)y \tag{6}$$

where $p_0(t)$, $p_1(t)$ and $p_2(t)$ are the coefficients of the polynomial.

To ensure a unique solution of the RBF interpolation of the level set function, the expansion coefficients must be subject to the following constraints (Morse et al. 2001):

$$\sum_{i=1}^{n} \alpha_i(t) = 0, \quad \sum_{i=1}^{n} \alpha_i(t) x_i = 0, \quad \sum_{i=1}^{n} \alpha_i(t) y_i = 0 \tag{7}$$

The level set function in (5) and (6) combined with the constraints in (7) can be rewritten in their matrix forms as:




(a) An MQ spline

(b) Partial derivative of the MQ spline in $x$ direction

**Fig. 2** Shape of an MQ spline and its partial derivative in $x$ direction

$$\mathbf{G}\boldsymbol{\alpha}(t) = \boldsymbol{\phi}(t) \tag{8}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \tag{9}$$

$$\mathbf{A} = \begin{bmatrix} g_1(\boldsymbol{x}_1) & \cdots & g_n(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ g_1(\boldsymbol{x}_n) & \cdots & g_n(\boldsymbol{x}_n) \end{bmatrix} \tag{10}$$

$$\mathbf{P} = \begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \tag{11}$$

$$\boldsymbol{\alpha}(t) = \{\alpha_1(t) \ \cdots \ \alpha_n(t) \ p_0(t) \ p_1(t) \ p_2(t)\}^T \tag{12}$$

$$\boldsymbol{\phi}(t) = \{\phi(\boldsymbol{x}_1, t) \quad \cdots \quad \phi(\boldsymbol{x}_n, t) \quad 0 \quad 0 \quad 0\}^T \tag{13}$$

Since the matrix $\mathbf{G}$ is theoretically invertible (Kansa et al. 2004), the generalized expansion coefficients can be calculated by:

$$\boldsymbol{\alpha}(t) = \mathbf{G}^{-1}\boldsymbol{\phi}(t) \tag{14}$$

And then the level set function at point $\boldsymbol{x}$ defined in (5) can be rewritten compactly as:

$$\phi(\boldsymbol{x}, t) = \boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\alpha}(t) \tag{15}$$

where

$$\boldsymbol{g}(\boldsymbol{x}) = \{g_1(\boldsymbol{x}) \quad \cdots \quad g_n(\boldsymbol{x}) \quad 1 \quad x \quad y\}^T \tag{16}$$

Substituting (15) into the Hamilton-Jacobi equation defined in (3) yields a RBF expansion coefficients-based governing equation:

$$\boldsymbol{g}(\boldsymbol{x})\frac{d\boldsymbol{\alpha}(t)}{dt} - V_n|(\nabla\boldsymbol{g}(\boldsymbol{x}))\boldsymbol{\alpha}(t)| = 0 \tag{17}$$

where

$$|(\nabla\boldsymbol{g}(\boldsymbol{x}))\boldsymbol{\alpha}| = \sqrt{\left(\frac{\partial\boldsymbol{g}(\boldsymbol{x})}{\partial x}\boldsymbol{\alpha}\right)^2 + \left(\frac{\partial\boldsymbol{g}(\boldsymbol{x})}{\partial y}\boldsymbol{\alpha}\right)^2} \tag{18}$$

$$\frac{\partial\boldsymbol{g}(\boldsymbol{x})}{\partial x} = \left\{\frac{\partial g_1(\boldsymbol{x})}{\partial x} \quad \cdots \quad \frac{\partial g_n(\boldsymbol{x})}{\partial x} \quad 0 \quad 1 \quad 0\right\}^T \tag{19}$$

$$\frac{\partial\boldsymbol{g}(\boldsymbol{x})}{\partial y} = \left\{\frac{\partial g_1(\boldsymbol{x})}{\partial y} \quad \cdots \quad \frac{\partial g_n(\boldsymbol{x})}{\partial y} \quad 0 \quad 0 \quad 1\right\}^T \tag{20}$$

And for this time-dependent interpolation problem, the following side constraints must be introduced to guarantee that the expansion coefficients can be solved due to the conditional positive definiteness of the MQ spline (Wang et al. 2007):

$$\sum_{i=1}^{n}\alpha_i(t) = 0, \ \sum_{i=1}^{n}\alpha_i(t)x_i = 0, \ \sum_{i=1}^{n}\alpha_i(t)y_i = 0 \tag{21}$$

In topology optimization problems, because the initial level set function values $\phi(t_0)$ at the knots should be given and the matrix $\mathbf{G}$ is theoretically invertible, the initial generalized expansion coefficients $\boldsymbol{\alpha}(t_0)$ can be obtained by solving the system of $n+3$ linear equations:

$$\boldsymbol{\alpha}(t_0) = \mathbf{G}^{-1}\boldsymbol{\phi}(t_0) \tag{22}$$

The formula above is calculated on line 20 of the MATLAB code.

Hence, the original time-dependent Hamilton-Jacobi partial differential equation is discretized into a system of coupled ODEs governing the motion of the dynamic interfaces, which can be regarded as a collocation formulation of the general method of lines:

$$\mathbf{G}\frac{d\boldsymbol{\alpha}}{dt} - \boldsymbol{B}(\boldsymbol{\alpha}, t) = 0 \tag{23}$$

where

$$\boldsymbol{B}(\boldsymbol{\alpha}, t) = \begin{Bmatrix} V_n(\boldsymbol{x}_1, t)|(\nabla\boldsymbol{g}(\boldsymbol{x}_1))\boldsymbol{\alpha}(t)| \\ \vdots \\ V_n(\boldsymbol{x}_n, t)|(\nabla\boldsymbol{g}(\boldsymbol{x}_n))\boldsymbol{\alpha}(t)| \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{24}$$

And the system of coupled non-linear ODEs can be solved by the first-order forward Euler's method and the approximate solution can be given by:

$$\boldsymbol{\alpha}(t_{i+1}) = \boldsymbol{\alpha}(t_i) + \triangle t\mathbf{G}^{-1}\boldsymbol{B}(\boldsymbol{\alpha}(t_i), t_i) \tag{25}$$

where $t_i$ is the $i$-th time step and $\triangle t$ is the time step size. The foregoing derivation can also be found in (Wang and Wang 2006a, Wang et al. 2007).

In practical implementations, (25) may cause undesired numerical instability, when implementing the RBF-based level set method in active contour model. Xie and Mirmehdi (2011) applied a normalization scheme to tackle the issue, in which the velocities were restated by:

$$\boldsymbol{B}(t_i) = \{V_n(\boldsymbol{x}_1, t_i) \quad \cdots \quad V_n(\boldsymbol{x}_n, t_i) \quad 0 \quad 0 \quad 0\}^T \tag{26}$$

This scheme can be considered as eliminating the gradient part from (23), and it is only consistent with (23) if $|\nabla\phi|$=1, which means a signed distance function should be held to the level set function $\phi$ at least around the boundary. The re-initialization approaches by solving a PDE (Peng et al. 1999) and fast marching method (Sethian 1999) are two common ways to keep the signed distance feature of the level set function to guarantee the accuracy. In this paper, a simple and

approximate re-initialization scheme is applied to prevent the values of $|\nabla\phi|$ around the boundaries from being too large or too small. The level set function $\phi$ in the whole design domain is updated to $\phi^u$ by the following approximate re-initialization scheme:

$$\phi^u = \frac{\phi}{mean\left(\left|\nabla\phi_1^0\right|,\ \left|\nabla\phi_2^0\right|,\ \ldots,\left|\nabla\phi_r^0\right|\right)} \tag{27}$$

where $\phi_r^0$ and $\left|\nabla\phi_r^0\right|$ are the level set value and the gradient on the $r$-th point respectively around the zero level set, and $|\cdot|$ represents the $l^2$-norm and $mean(f_1, f_2, \cdots, f_r)$ denotes the mean value of $f_i (i = 1, 2, \cdots, r)$. Thus the denominator of (27) represents the average value of $|\nabla\phi|$ around the boundaries of the structure, i.e. the zero level sets. Because the relationship between $\phi$ and $\alpha$ is linear, the (27) can be rewritten as an update scheme from $\alpha$ to $\alpha^u$:

$$\alpha^u = \frac{\alpha}{mean\left(\left|\nabla\phi_1^0\right|,\ \left|\nabla\phi_2^0\right|,\ \ldots,\left|\nabla\phi_r^0\right|\right)} \tag{28}$$

The formula above is calculated on line 85 of the MATLAB code.

Figure 3 shows the schematic diagram of this approach. Compared with conventional re-initialization schemes, this approach is easy to be implemented and is able to retain the boundaries effectively. Furthermore, because this scheme only changes the relative value of the level set function, it will not prevent the nucleation of new holes (see discussions in Section 3.6).

An approximate $\delta(\phi)$ function is applied into the evolution scheme of (25) to avoid the unbounded growth of the level set function, which is defined as:

$$\delta(\phi) = \begin{cases} 0, & \phi > \Delta \\ \dfrac{3}{4\Delta}\left(1-\dfrac{\phi^2}{\Delta^2}\right), & -\Delta \le \phi \le \Delta \\ 0, & \phi < -\Delta \end{cases} \tag{29}$$

Therefore, the updating equation is modified as:

$$\alpha(t_{i+1}) = \alpha^u(t_i) + \Delta t \mathbf{G}^{-1}\hat{\boldsymbol{B}}(\alpha^u(t_i), t_i) \tag{30}$$

where $\hat{\boldsymbol{B}}(\alpha^u(t_i), t_i)$ represents the updated velocity vector at time step $t_i$ and it can be written as:

$$\hat{\boldsymbol{B}}(\alpha^u(t_i), t_i) = \begin{Bmatrix} V_n(\boldsymbol{x}_1, t_i)\delta(\phi(\boldsymbol{x}_1, \alpha^u(t_i))) \\ \vdots \\ V_n(\boldsymbol{x}_n, t_i)\delta(\phi(\boldsymbol{x}_n, \alpha^u(t_i))) \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{31}$$

The (29) and (30) are calculated on lines 77–84 of the MATLAB code.

Here the parameter $\Delta$ is used to control the magnitude of the upper and lower bound of the level set function. The effect of $\delta(\phi)$ is illustrated in Fig. 4. When the level set value is above $\Delta$ or below $-\Delta$, the normal velocity will vanish as a result of multiplying the $\delta(\phi)$ function to prevent any value of $\phi$ from approaching infinity. The value of $\Delta$ can be chosen as several times of the length of a mesh grid and larger than 5 is suggested.

Please note according to (30), the updating scheme of the level set function can be considered by replacing $|\nabla\phi|$ by $\delta(\phi)$ in the original (3). This scheme is firstly proposed by Zhao et al. (1996). Later it was studied by Chan and Vese (2001) for image processing and by Wang and Wei (2005) for topology optimization. It is also similar to the one proposed by Yamada et al. (2010) but without incorporating the fictitious interface energy.

Due to the parameterization using MQ RBFs with global supports, a relatively smooth level set function can be maintained with an approximate re-initialization operation during the evolution. The nucleation of new holes inside the material domain becomes possible and thus the final design is fairly insensitive to initial designs. Furthermore, since the time step size and the number of RBF knots are determined by the requirement of optimization convergence rather than the need to solving the Hamilton-Jacobi equation, the constraint from the CFL condition on the temporal and spatial discretization can be relaxed (Wang et al. 2007).

## 2.3 Compliance minimization problem

For the sake of simplicity but without losing generality, we discuss the compliance minimization problem for statically loaded linear elastic structure subject to a constraint on the volume of material as shown in Fig. 5. The following is the mathematical model based on the

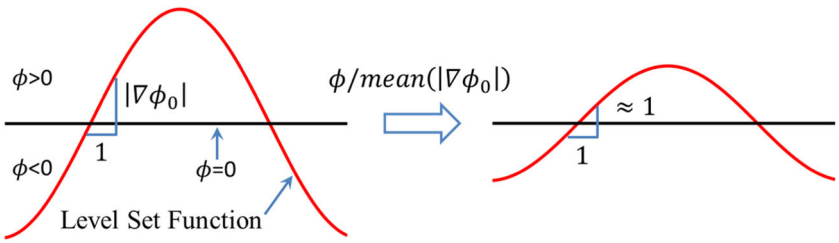



**Fig. 3** An approximate re-initialization process

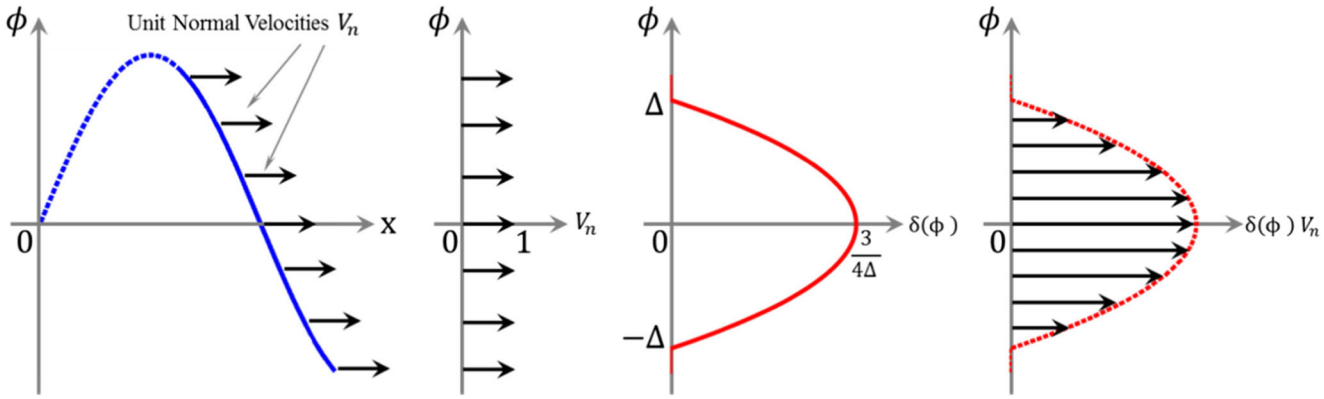



**Fig. 4** The effect of the δ(ϕ) function

level set method of the compliance minimization problem (Wang et al. 2003; Allaire et al. 2004):

$$\text{Minimize}_{\phi}\ J(\boldsymbol{u},\phi)=\int_{\mathrm{D}}(\varepsilon(\boldsymbol{u}):\boldsymbol{C}:\varepsilon(\boldsymbol{u}))H(\phi)d\Omega \tag{32}$$

$$s.t.\begin{cases} a(\boldsymbol{u},\boldsymbol{v},\phi)=l(\boldsymbol{v},\phi) & \forall \boldsymbol{v}\in\boldsymbol{U} \\ G(\phi)=\int_{\mathrm{D}}H(\phi)d\Omega-V_{\max}\le 0 & \\ \boldsymbol{u}=\boldsymbol{u}_0 & in\ \Gamma_u \\ C:\varepsilon(\boldsymbol{u})\cdot\boldsymbol{n}=\boldsymbol{\tau} & in\ \Gamma_\tau \end{cases} \tag{33}$$

where the notations in the above equations are defined as:

$$a(\boldsymbol{u},\boldsymbol{v},\phi)=\int_{\mathrm{D}}(\varepsilon(\boldsymbol{u}):\boldsymbol{C}:\varepsilon(\boldsymbol{v}))H(\phi)d\Omega \tag{34}$$

$$l(\boldsymbol{v},\phi)=\int_{\Gamma_\tau}\boldsymbol{\tau}\cdot\boldsymbol{v}d\Gamma+\int_{\mathrm{D}}\boldsymbol{b}\cdot\boldsymbol{v}H(\phi)d\Omega \tag{35}$$

where $J(\phi)$ is the objective function, $\boldsymbol{u}$ the displacement field, $\varepsilon$ the linearized strain tensor, $\boldsymbol{C}$ the Hook elasticity tensor, $\boldsymbol{u}_0$, $\boldsymbol{\tau}$ and $\boldsymbol{b}$ the constant values that represent the given displacement, traction and body force respectively. The linear elastic equilibrium equation is written in its weak variational form in terms of the energy bilinear form $a(\boldsymbol{u},\boldsymbol{v},\phi)$ and the load linear form $l(\boldsymbol{v},\phi)$, with $\boldsymbol{v}$ denoting a virtual displacement field in the space $\boldsymbol{U}$ of kinematically admissible displacement fields. $G(\phi)$ is the constraint introduced to limit material usage and $V_{\max}$ is the maximum allowable volume in the design domain D. $H(\phi)$ is the Heaviside function defined as follows to represent the existence of the material at a point of the level set function value $\phi$:

$$H(\phi)=\begin{cases} 1 & if\ \phi\ge 0 \\ 0 & if\ \phi<0 \end{cases} \tag{36}$$

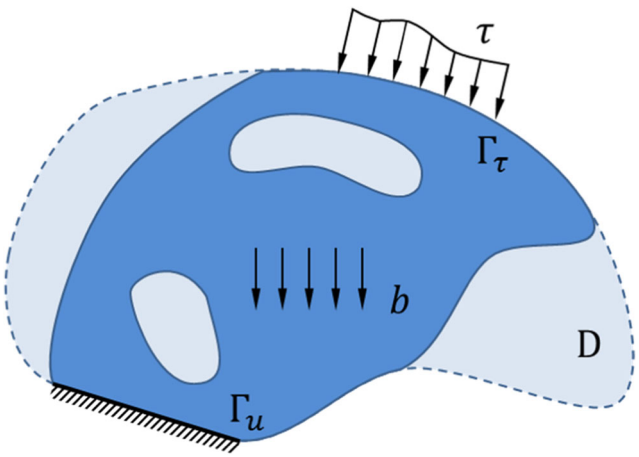



**Fig. 5** Structural topology optimization problem definition

In the level set formulation here, the total design domain consists of the domain of void material and the domain of full material. And there is no intermediate density material, which is different from the variable density method (Bendsøe and Sigmund 2003). It should be noticed that in numerical implementation, instead of exact zero, a small value for the Heaviside function is used when the level set function value is below zero.

According to the methods using the shape derivative (Sokołowski and Zolésio 1992; Choi and Kim 2005), the normal velocity $V_{\mathrm{n}}$ along the moving free boundary can be simply determined by the strain energy density and a Lagrange multiplier to perform gradient-based optimization methods, such as the steepest descent method. We refer the readers to the classical literatures (Wang et al. 2003; Allaire et al. 2004) for more details about the shape sensitivity analysis. For a compliance minimum problem, the normal velocity can be derived as follows:

$$V_{\mathrm{n}}=\varepsilon(\boldsymbol{u}):\boldsymbol{C}:\varepsilon(\boldsymbol{u})-\lambda \tag{37}$$

where $\lambda$ is the Lagrange multiplier to deal with the constraint of volume fraction, which can be calculated using the bi-sectioning algorithm (Wang and Wang 2006b; Sigmund 2001) or the following augmented Lagrangian updating scheme (Wei and Wang 2006; Rochafellar 1973):

$$\lambda^{k+1}=\begin{cases} \mu G^k & k\le n_{\mathrm{R}} \\ \lambda^k+\gamma^k G & k>n_{\mathrm{R}} \end{cases} \tag{38}$$

where $\mu$ and $\gamma^k$ are parameters in the $k$-th iteration of the optimization, and $\gamma^k$ is updated using the scheme:

$$\gamma^{k+1}=\min\left(\gamma^k+{}^{\triangle}\gamma,\gamma_{\max}\right)\quad k>n_{\mathrm{R}} \tag{39}$$

where $\triangle\gamma$ is the increment and $\gamma_{\max}$ is the upper limit of the parameter $\gamma$.

And since the volume fraction of initial design in the level set-based optimization method usually does not meet the prescribed volume fraction, the volume constraint is relaxed in the first $n_R$ iterations as below:

$$G^k = \int_D H(\phi)d\Omega - \left[V_0 - (V_0 - V_{\max})\frac{k}{n_R}\right] \quad k \le n_R \tag{40}$$

Here the ersatz material model is used to perform the finite element analysis to obtain the naturally extended velocity field from the strain energy density in the whole design domain (Wang et al. 2007). Those readers who are interested in more discussion of the algorithm are referred to the publications referenced above, and more details revealed by the MATLAB implementation are discussed in the following section.

## 3 MATLAB implementation

In this section, the 88-line MATLAB code (see the Appendix) for the compliance minimization problem is explained in detail. It should be noted that following the well-known 99-line code (Sigmund 2001), several assumptions are made to simplify the MATLAB code. In particular, the design domain is assumed to be rectangular and meshed with square finite elements of unit length.

The code is called from the MATLAB prompt by means of the following line:

```
TOPRBF(nelx,nely,volfrac)
```

where `nelx` and `nely` are the number of elements in the horizontal and vertical directions respectively, and `volfrac` is the prescribed volume fraction. Other variables as well as boundary conditions can be edited in the code if needed. The version of the code given here performs the optimization for a cantilever beam as seen in Fig. 6, the detail of which is given in Section 3.6. A suggested first call of the program is:

```
TOPRBF(60,30,0.5)
```

The program consists of five parts: level set function initialization (lines 3–10), radial basis functions initialization (lines 11–20), finite element analysis preparation (lines 21–32), boundary conditions definition (lines 33–37), and the main optimization loop (lines 38–87). These parts are discussed in detail from subsection 3.1 to 3.5, and subsection 3.6 presents some numerical results obtained with the code.

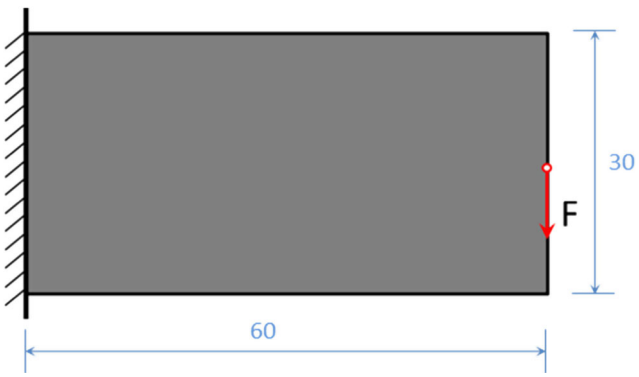



**Fig. 6** The design domain and boundary conditions for the optimization of a cantilever beam

### 3.1 Level set function initialization (lines 3–10)

The initial level set function `Phi` is chosen as a signed distance function with initial holes distributed in the design domain and then the values are constrained between −3 and 3. The matrices `X` and `Y` are the coordinates of the nodes in the Cartesian coordinate system. The vectors `hX` and `hY` are the coordinates of the centers of initial holes, whose radii are all defined by `r` that is tenth of the width `nely` of the design domain here.

Instead of using the `for` loop, a vectorized function `bsxfun`, is adopted here to construct the initial level set function (lines 8–10).

The classic code with the `for` loop which has been often used before:

```
  Phi = sqrt((X-hX(1)).^2+(Y-hY(1)).^2)-r;
for iH = 2:15
  Phi = min(Phi,sqrt((X-hX(iH)).^2+(Y-
hY(iH)).^2)-r);
end
Phi = max(-3,min(3,Phi));
```

The more concise vectorized code with the function `bsxfun` used here:

```
dX= bsxfun(@minus,repmat(X,[ 1,1,numel
(hX)]),reshape(hX,1,1,numel(hX)));
dY = bsxfun(@minus,repmat
(Y,[ 1,1,numel(hY)]),reshape(hY,1,1,nume-
l(hY)));
Phi = max(-3,min(3,min(sqrt(dX.^2+dY.^2)-
r,[],3)));
```

### 3.2 Radial basis function initialization (lines 11–20)

`cRBF` corresponds to the shape parameter $c$ of the MQ spline defined in (4). `A` and `G` correspond to the matrices **A** and **G** respectively defined in (9–11), `pGpX` and `pGpY` correspond to the $\frac{\partial g(\mathbf{x})}{\partial x}$ and $\frac{\partial g(\mathbf{x})}{\partial y}$ on all points respectively defined in (19–20), and `Alpha` corresponds to the generalized expansion coefficients $\boldsymbol{\alpha}$ defined in (12). Here it is assumed that the knots of RBFs are coincident to the nodes of the structural mesh.

### 3.3 Finite element analysis preparation (lines 21–32)

The definition of the material property is on line 22: `E0` is the Young's modulus of the solid material, `Emin` is the Young's modulus of the void material, and `nu` is the Poisson's ratio of both materials. The element stiffness matrix `KE` with unit Young's modulus is identical for all elements, whose construction on lines 23–27 is the same as in the 88-line code (Andreassen et al. 2011). The assembly of the global stiffness matrix `K` also follows almost the same procedure as in the 88-line code (Andreassen et al. 2011) with the index vectors `iK` and `jK` and the element-wise matrix `sK`. Please note here both nodes and elements are numbered column-wise from the lower left corner to the upper right corner of the design domain, which is different from the way in the previous 88-line code developed by Andreassen et al. (2011). The `nely`-by-`nelx` matrix `eleN1` stores the first node number index for all elements, and it is used to construct the matrix `eleNode`, the *i*-th row of which contains the four node number indices corresponding to the *i*-th element. `eleNode` is commonly used in the general finite element method programming, and it helps to calculate the volume of elements (lines 46–48) and approximately re-initialize the level set function (line 85). And now the same variable `edofMat` as in the 88-line code (Andreassen et al. 2011) can also be obtained by the matrix `eleNode`.

### 3.4 Boundary conditions definition (lines 33–37)

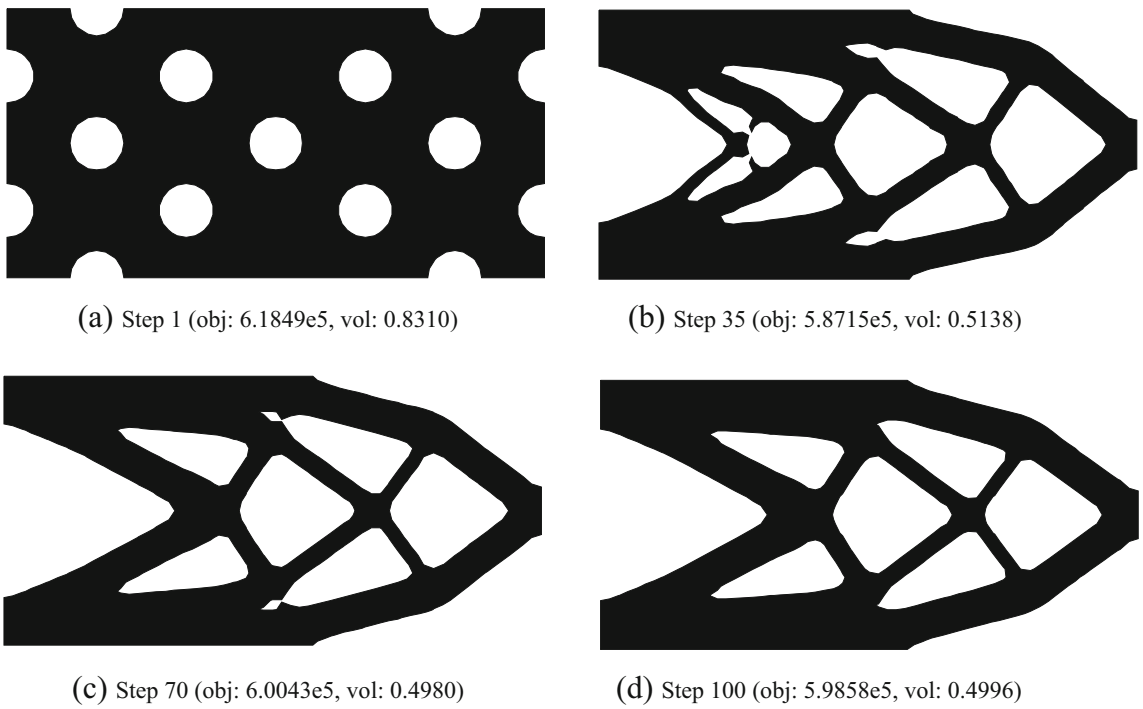
The nodal force vector `F` applied to the design domain is constructed on line 34. And the displacement boundary condition is defined on line 35, while `fixeddofs` stores the indexes of the degrees of freedoms that the displacements are fixed as zero.

### 3.5 Optimization loop (lines 38–87)

The optimization loop consists of six parts: iteration preparation (lines 39–41), finite element analysis (lines 43–54), results visualization (lines 55–62), convergence check (lines 63–67), computation of Lagrange multiplier (lines 68–74) and the level set function update (lines 75–86).

#### 3.5.1 Iteration preparation (lines 39–41)

`nLoop` is the maximum iteration number of optimization. When the volume of the initial design is greater than the prescribed volume, the volume constraint would be relaxed at first and then gradually tightened during `nRelax` iterations. The time step size `dt` for the level set function evolution corresponds to $\triangle t$ in (25). And `delta` corresponds to $\Delta$ in (29). Line 40 also defines the parameters of Lagrange multiplier calculation. Finally, the vectors of objective function value and total volume fraction are preallocated respectively outside the iteration.



(a) Step 1 (obj: 6.1849e5, vol: 0.8310) (b) Step 35 (obj: 5.8715e5, vol: 0.5138)

(c) Step 70 (obj: 6.0043e5, vol: 0.4980) (d) Step 100 (obj: 5.9858e5, vol: 0.4996)

**Fig. 7** Evolution history of the optimized design of the cantilever beam problem

#### 3.5.2 Finite element analysis (lines 43–54)

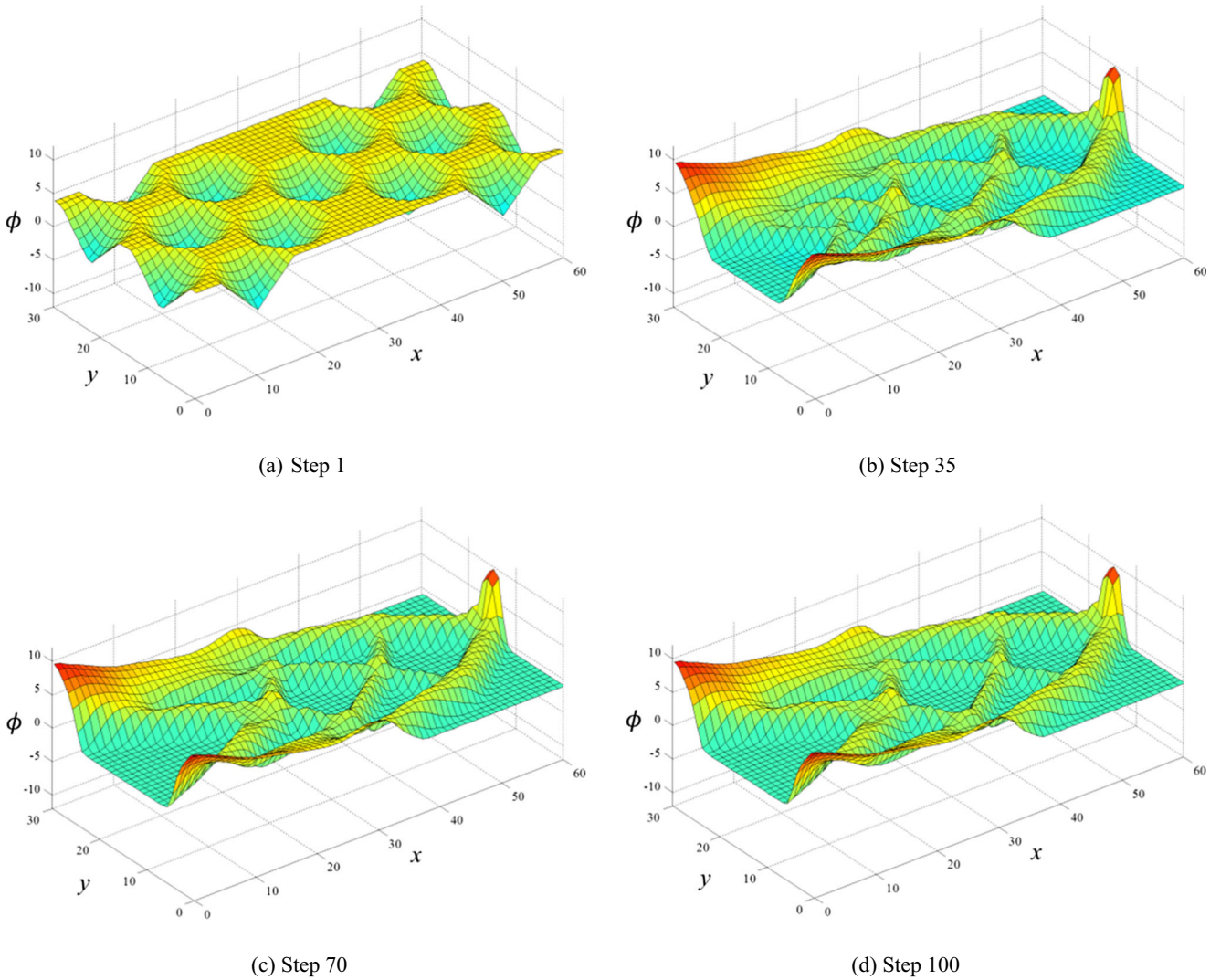
Each iteration of the optimization loop starts with the finite element analysis using the ersatz material model based on the approximate volume of each element. The bilinear interpolation is adopted to calculate the values of the level set function on a refined mesh grid (21-by-21 in the appended code) inside each element. The ratio of the number of the points with non-negative interpolation value is assumed to be the volume fraction of the solid part which is stored in the vector `eleVol`. And then the equivalent Young's modulus in the element stiffness matrix can be calculated by: `Emin+eleVol'*(E0-Emin)`. Finally, the element strain energy field or the so-called velocities field `eleComp` and the objective function value `comp` are calculated.

#### 3.5.3 Results visualization (lines 55–62)

During the optimization loop, the information of the objective function values and the structural volumes is always printed on the screen. The solid material distribution is plotted using the `contourf` function, while the black region represents the solid part. And the level set function itself is also plotted using the `surf` function. Finally, the history graphs of the objective functions and the total volume fractions are given.

#### 3.5.4 Convergence check (lines 63–67)

The optimization loop is terminated before reaching the maximum number of iterations when both of the following conditions are satisfied after `nRelax` iterations: the volume difference is within 0.1% of the prescribed volume, and the




(a) Step 1

(b) Step 35

(c) Step 70

(d) Step 100

**Fig. 8** Evolution history of the level set surface of the cantilever beam problem

previous nine values of the objective function differ from the current value by less than 0.1% of the current value.

#### 3.5.5 Lagrange multiplier computation (lines 68–74)

The penalty function method is used in the first `nRelax` iterations and then the augmented Lagrangian method is used to solve the constrained optimization problem. During the optimization loop, the parameters for imposing the volume constraint with the penalty function method or the augmented Lagrange method should be updated prior to the level set function update according to their previous values and the current volume. It should be noted that the parameter `mu` ($\mu$ in (38)) should be large enough to make the volume decrease quickly at first and `gamma` ($\gamma^k$ in (38)) should also be chosen carefully to avoid oscillation. It should be noted that, it is necessary for the users to adjust the time-step size and the above parameters for Lagrange multiplier computation depending on the specific problems.

#### 3.5.6 Level set function update (lines 75–86)

Firstly, $|\nabla\phi_0|$ in (28) is calculated as `gradPhi`. Then $\delta(\phi)$ in (29) is calculated as `DeltaPhi`. And `nodeComp` is the strain energy density in each node mapped from the strain energy of elements `eleComp` and it corresponds to the velocities $\hat{\boldsymbol{B}}$ in (31). Here, we divide `nodeComp` by their median to normalize the velocities to make parameters for Lagrange multiplier computation insensitive to the finite element models. However, along with the adjustment for these parameters, other normalization strategies can also be implemented. For example, we can divide `nodeComp` by their mean instead (replacing `median` on line 83 with `mean`), which works better for the cantilever beam problem called with the input line with smaller volume fraction:

```
TOPRBF(60,30,0.3)
```

Finally, the new generalized expansion coefficients `Alpha` is obtained based on (28) and (30), and the level set function `Phi` can then be updated. Here, `unique(eleNode((eleVol<1 & eleVol>0),:))` is used to find out the node numbers of the elements cut by the structural interface.

### 3.6 Numerical examples

In this section, a numerical example for classical benchmark problems is presented to illustrate the performance and success of the parameterized level set method using the MQ RBF for structural shape and topology optimization. For the following example, a fixed mesh using four-node bilinear square elements is specified over the entire rectangular design domain for finite element analysis of the plane stress problem.

As shown in Fig. 6, the left side of a cantilever beam is fixed and a concentrated force $F = -100$ is vertically applied at the middle point of the right side. This whole design domain of size $60 \times 30$ is discretized by $60 \times 30$ elements with the following properties: the Young's elasticity modulus $E = 1$ for solid material, $E = 10^{-9}$ for void material, and Poisson's ratio $\nu = 0.3$. And the prescribed volume fraction is set to 50%. Figure 7 shows the iterative optimization of the structure

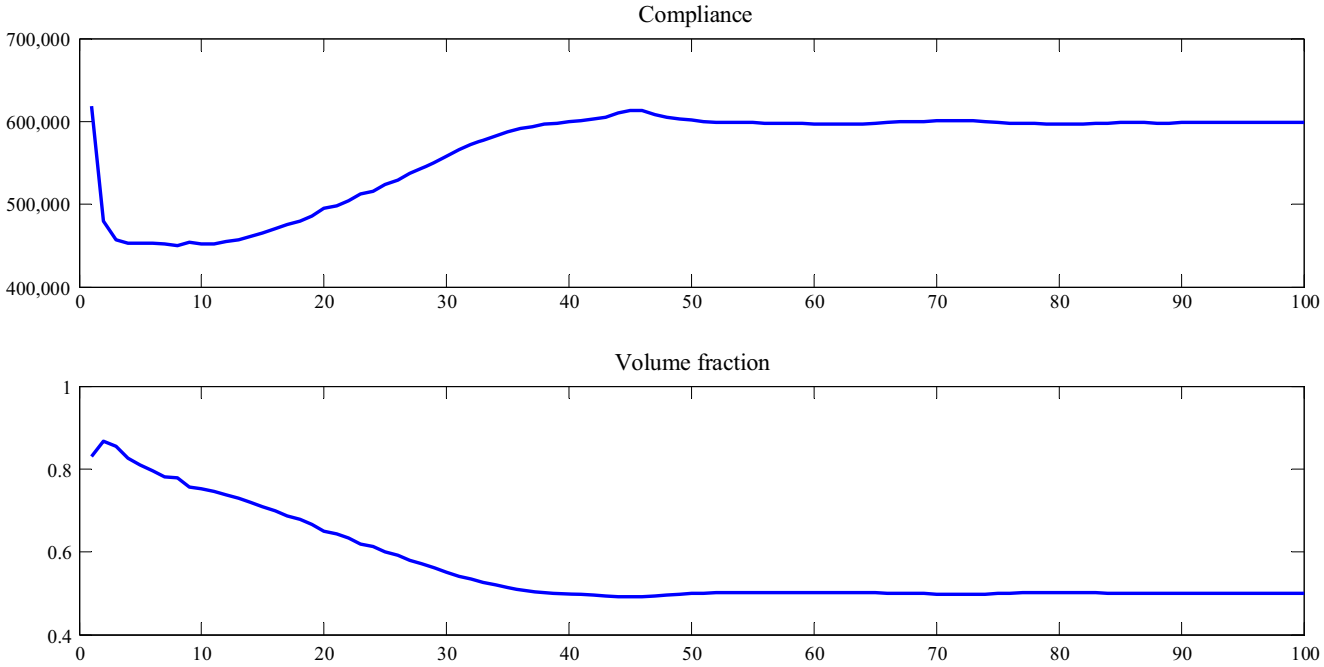
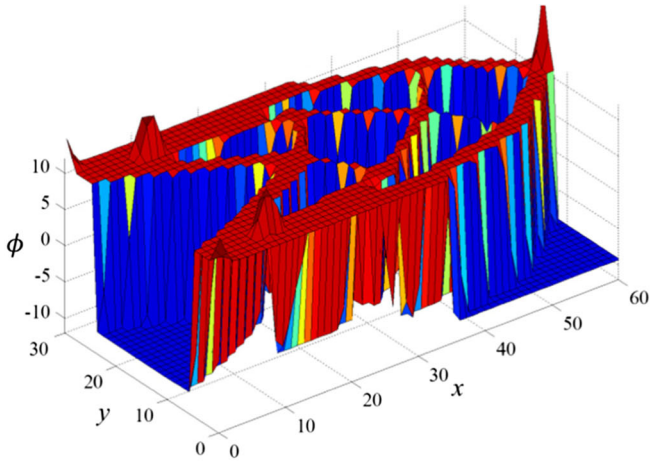
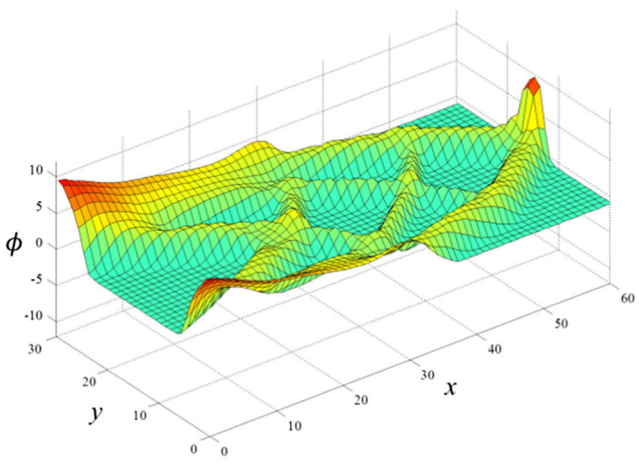


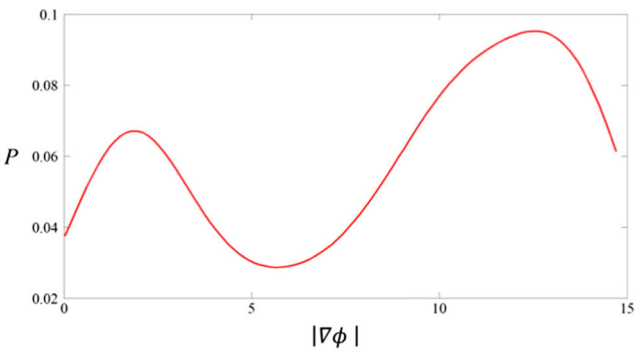

**Fig. 9** Convergence curves of the compliance and volume fraction of the cantilever beam problem



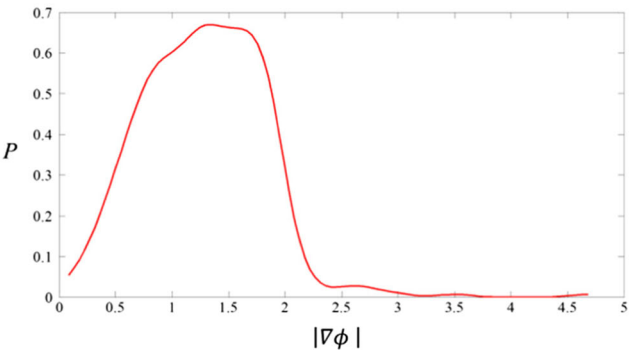

(a) Level set surface without the approximate re-initialization




(b) Level set surface with the approximate re-initialization




(c) Histogram of the distribution of $|\nabla\phi|$ without the approximate re-initialization




(d) Histogram of the distribution of $|\nabla\phi|$ with the approximate re-initialization

**Fig. 10** Numerical effect of the approximate re-initialization operation

given in black-and-white form based on the zero isosurface of the level set function, which is obtained by the code given in the Appendix called with the input line:

```
TOPRBF(60,30,0.5)
```

Figure 8 shows the corresponding evolution of the level set function, and Fig. 9 shows the convergence curve of the objective function value and the volume constraint.

Figure 10 shows the optimized design and its corresponding level set surface of the cantilever beam problem without using the approximate re-initialization operation proposed in (28). And Fig. 10 also gives the comparison of the histograms which provide the proportions

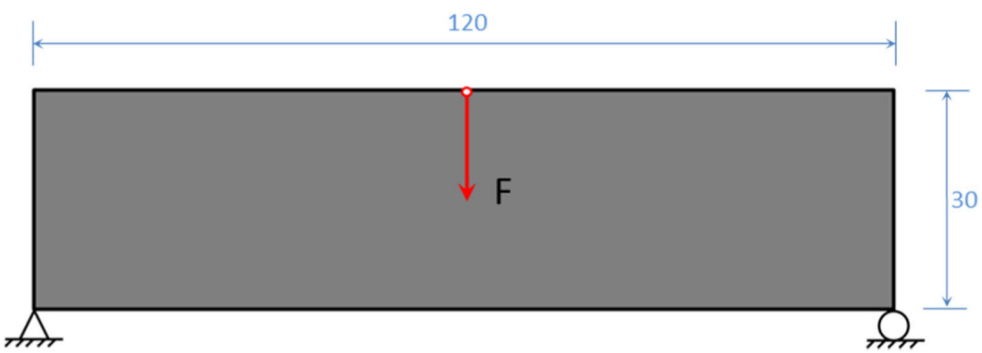



**Fig. 11** The design domain and boundary conditions for the optimization of a freely supported beam

(a) Step 1 (obj: 2.9640e5, vol: 0.9155)



(b) Step 35 (obj: 3.5842e5, vol: 0.5067)

(c) Step 70 (obj: 3.6105e5, vol: 0.4988)

(d) Step 102 (obj: 3.6187e5, vol: 0.5001)

**Fig. 12** Evolution history of the optimized design of the freely supported beam problem

of $|\nabla\phi|$ around the boundary for the two cases. It can be seen that the proposed approximate re-initialization scheme can effectively prevent the value of $|\nabla\phi|$ around the boundary from being too large or too small.

## 4 Simple extensions

### 4.1 Different boundary condition

It is easy to change the code to solve other compliance minimization problems with different boundary conditions, which is the same as the implementation in the 88-line code. An additional example is presented here.

As shown in Fig. 11, the bottom left corner of the beam is totally fixed while only the vertical displacement of the bottom right corner of the beam is fixed, and a concentrated force $F = -100$ is vertically applied at the middle point of the top side. This whole design domain of size $120 \times 30$ is discretized by $120 \times 30$ elements with the same properties as in Section 3.6. And the prescribed volume fraction is also set to 50%.

To solve this optimization problem, original lines 34–35 of the code should be changed as follows:

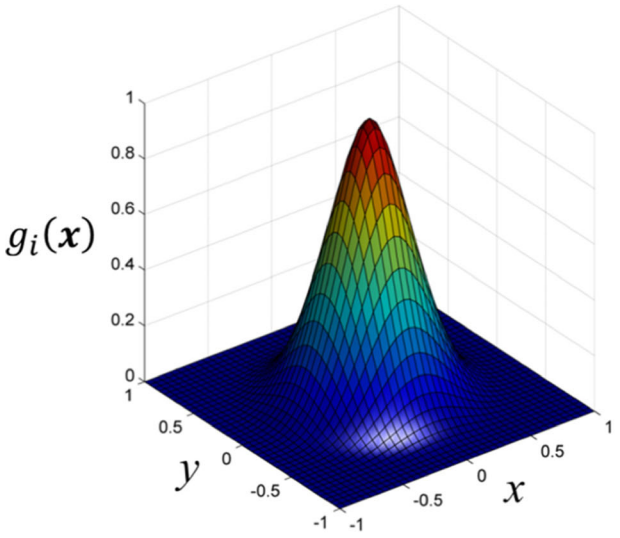
```
F = sparse(2*((nely+1)*(ceil(nelx/2)+
1)),1,-100,2*nNode,1);
fixeddofs=[1,2,2*(nely+1)*nelx+2];
```

Figure 12 shows the iterative optimization of the structure obtained by the changed code called with the input line:

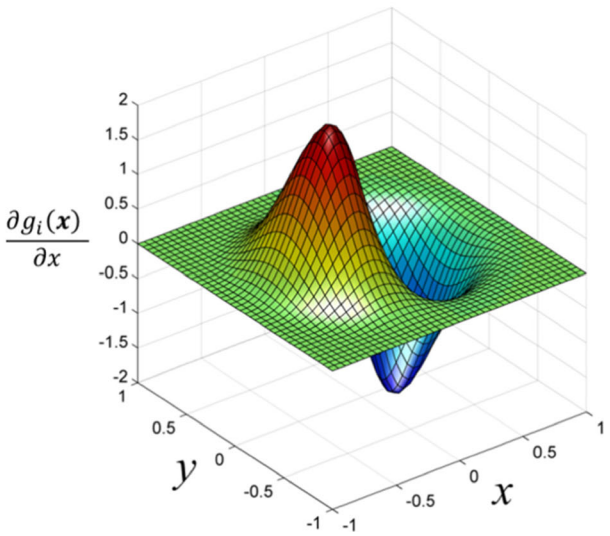
```
TOPRBF(120,30,0.5)
```

### 4.2 Different type of RBFs

Here, we replace the MQ splines with the $C^2$ Wendland CSRBFs (Wendland 1995) given as:

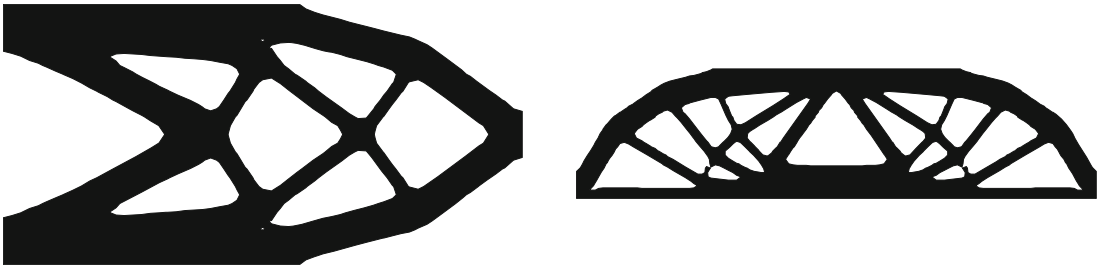



(a) $C^2$ CSRBF




(b) Partial derivative of $C^2$ CSRBF in $x$ direction

**Fig. 13** Shape of $C^2$ CSRBF and its partial derivative in $x$ direction



(a) Cantilever beam problem (Step 88)

(b) Freely supported beam problem (Step 107)

**Fig. 14** Optimized design using the $C^2$ CSRBF

$$g(r) = (\max(0, 1-r))^4 \cdot (4r + 1) \tag{41}$$

where the radius of support $r$ in two-dimensional Euclidean space is given as:

$$r(\boldsymbol{x}) = \frac{1}{d_{sp}} \sqrt{(\boldsymbol{x}-\boldsymbol{x}_i)^2 + c^2}, \quad \boldsymbol{x}_i \in \mathrm{D} \tag{42}$$

where $d_{sp}$ is a positive constant as the support radius of the RBF, and $c$ is the small enough positive constant as given in (4) to prevent divide-by-zero error in the following calculation of derivatives which can be written as:

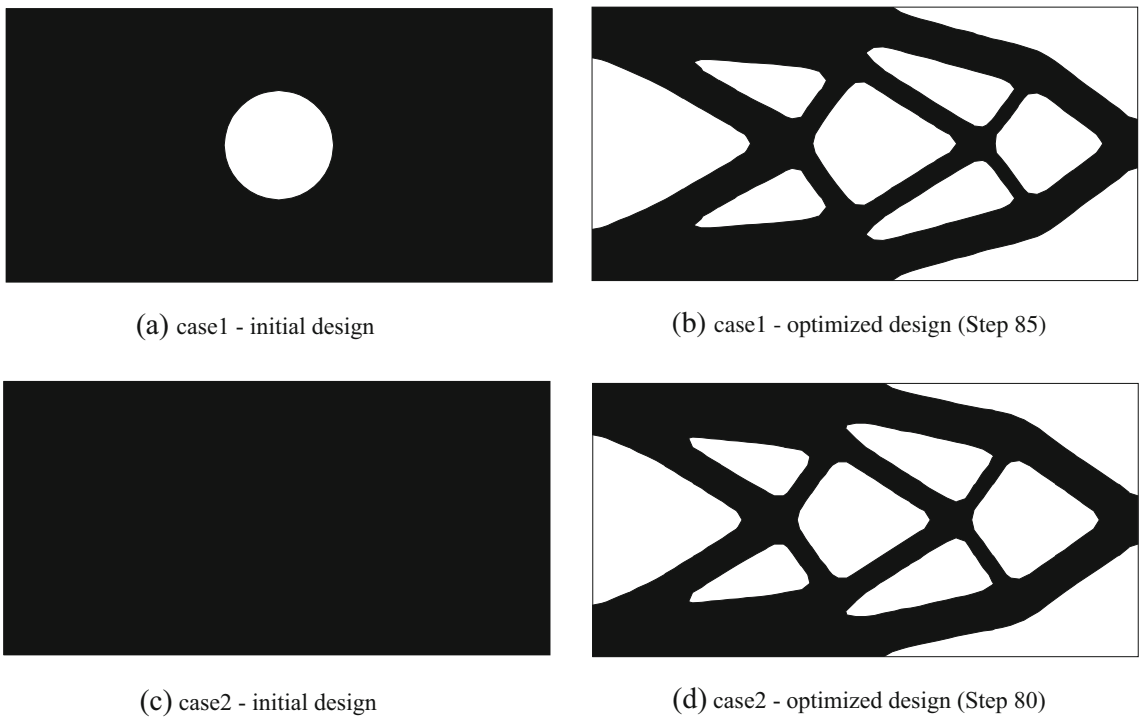
$$\frac{\partial g}{\partial x} = (\max(0, 1-r))^3 \cdot (-20r) \cdot \frac{\partial r}{\partial x} \tag{43}$$

$$\frac{\partial g}{\partial y} = (\max(0, 1-r))^3 \cdot (-20r) \cdot \frac{\partial r}{\partial y} \tag{44}$$

The $C^2$ CSRBF and its partial derivative in $x$ direction are displayed in Fig. 13.

The users just need to replace the original lines 16–19 with the following code:

```
dsp = 2;    rCS = sqrt(Ax.^2+Ay.^2+cRBF^2)/dsp;
G = [(max(0,1-rCS).^4).*(4*rCS+1),ones(nNode,1),X(:),Y(:);[ones(1,nNode);X(:)';Y(:)'],zeros(3,3)];
pGpX = [(max(0,1-rCS).^3).*(-20*rCS).*(Ax./rCS/dsp^2),repmat([0,1,0],nNode,1);repmat([0;1;0],1,nNode),zeros(3,3)];
```



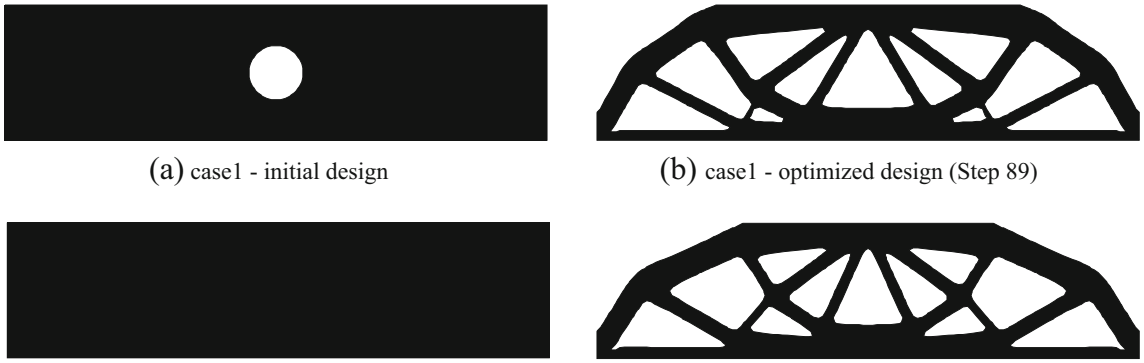
(a) case1 - initial design

(b) case1 - optimized design (Step 85)

(c) case2 - initial design

(d) case2 - optimized design (Step 80)

**Fig. 15** Different initial designs and the optimized designs of the cantilever beam problem



(a) case1 - initial design (b) case1 - optimized design (Step 89)

(c) case2 - initial design (d) case2 - optimized design (Step 100)

**Fig. 16** Different initial designs and the optimized designs of the freely supported beam problem

```
pGpY = [ (max(0,1-rCS).^3).*(-20*rCS).*
(Ay./rCS/dsp^2),repmat([ 0,0,1],nNode,1);
repmat([ 0;0;1],1,nNode),zeros(3,3)];
```

It can be seen that the optimized designs for the cantilever beam and the freely supported beam obtained by using the $C^2$ CSRBF shown in Fig. 14 are almost the same as those shown in Figs. 7 and 12 respectively using the MQ spline. This result indicates that the algorithm is not sensitive to the type of RBFs.

### 4.3 Different initial designs

It is easy to change the code to define initial designs different from the default one with distributed holes.

The first case of different initial design is chosen as a signed distance function with only one initial hole in the center of design domain, and the values are also constrained between −3 and 3. The users just need to replace the original lines 4–6 with the following code:

```
r = nely*0.2;%RADIUS OF INITIAL HOLES
hX = nelx/2;
hY = nely/2;
```

The second case of different initial design is chosen as a fully solid design with no initial hole in the design domain, and the values of level set function are set to be a small positive constant inside the design domain. The users need to replace the original lines 4–10 with the following code:

```
Phi = zeros(nely+1,nelx+1);
Phi(2:end-1,2:end-1) = 1e-3*ones(nely-
1,nelx-1);
```

And the original line 85 should also be replaced with the following code since no hole exists in the initial fully solid design:

```
if any(eleVol<1 & eleVol>0)
  Alpha = Alpha/mean(gradPhi(unique
(eleNode((eleVol<1 & eleVol>0),:))));end
```

It can be seen that the optimized designs obtained by using the two different initial designs shown in Fig. 15 are almost the same as the result shown in Fig. 7. And the optimized designs of the freely supported beam problem shown in Fig. 16 are only a little different from the original result shown in Fig. 12. This result may indicates that the algorithm is fairly insensitive to the initial designs and can obtain complex optimized designs by using the initial designs with less number of holes or even without holes.

## 5 Conclusions

This paper has presented an 88-line MATLAB implementation of the parameterized level set based structural topology optimization method using RBFs. This level set approach can maintain a relatively smooth level set function with an approximate re-initialization operation during the optimization process and largely eliminate the dependency on initial designs due to its capability in nucleation of new holes inside the material domain. Furthermore, there is no need to implement any filter or smoothing scheme during the optimization process. Numerical examples are provided to demonstrate the capability of the method in topology optimization for two-dimensional compliance minimization problems.

The 88-line code is provided for educational purposes, and the users can edit the code according to their requirement. This code provides a compact and vectorized way for the initialization of the level set function with distributed holes. Some modifications of the code such like a different boundary condition, a different type of RBF and a different initial design are also given. The 88-line code is given in the appendix for users' reference and the source code is available as supplementary material to be downloaded with the article.

**Acknowledgements** This research was supported by the National Natural Science Foundation of China under Grant Nos. 11372004, 11002056, 11002058 and the State Key Laboratory of Subtropical Building Science under Grant No. 2016 KB13.

## Appendix

```
%%%% An 88 LINE PARAMETERIZED LEVEL SET-BASED TOPOLOGY OPTIMIZATION CODE %%%%
function TOPRBF(nelx,nely,volfrac)
%% LEVEL SET FUNCTION INITIALIZATION
    r = nely*0.1;%RADIUS OF INITIAL HOLES
    hX = nelx*[repmat([1/6,5/6],1,3),repmat([0,1/3,2/3,1],1,2),1/2];
    hY = nely*[kron([0,1/2,1],ones(1,2)),kron([1/4,3/4],ones(1,4)),1/2];
    [X,Y] = meshgrid(0:1:nelx,0:1:nely);
    dX = bsxfun(@minus,repmat(X,[1,1,numel(hX)]),reshape(hX,1,1,numel(hX)));
    dY = bsxfun(@minus,repmat(Y,[1,1,numel(hY)]),reshape(hY,1,1,numel(hY)));
    Phi = max(-3,min(3,min(sqrt(dX.^2+dY.^2)-r,[],3)));
%% RADIAL BASIS FUNCTION INITIALIZATION
    cRBF = 1e-4;%RBF PARAMETER
    nNode = (nely+1)*(nelx+1);
    Ax = bsxfun(@minus,X(:),X(:)');
    Ay = bsxfun(@minus,Y(:),Y(:)');
    A = sqrt(Ax.^2+Ay.^2+cRBF^2);
    G = [A,ones(nNode,1),X(:),Y(:);[ones(1,nNode);X(:)';Y(:)'],zeros(3,3)];
    pGpX = [Ax./A,repmat([0,1,0],nNode,1);repmat([0;1;0],1,nNode),zeros(3,3)];
    pGpY = [Ay./A,repmat([0,0,1],nNode,1);repmat([0;0;1],1,nNode),zeros(3,3)];
    Alpha = G\[Phi(:);0;0;0];
%% FINITE ELEMENT ANALYSIS PREPARATION
    E0 = 1; Emin = 1e-9; nu = 0.3; %MATERIAL PROPERTIES
    A11 = [12  3 -6 -3;  3 12  3  0; -6  3 12 -3; -3  0 -3 12];
    A12 = [-6 -3  0  3; -3 -6 -3 -6;  0 -3 -6  3;  3 -6  3 -6];
    B11 = [-4  3 -2  9;  3 -4 -9  4; -2 -9 -4 -3;  9  4 -3 -4];
    B12 = [ 2 -3  4 -9; -3  2  9 -2;  4  9  2  3; -9 -2  3  2];
    KE = 1/(1-nu^2)/24*([A11 A12;A12' A11]+nu*[B11 B12;B12' B11]);
    eleN1 = repmat((1:nely)',1,nelx)+kron(0:nelx-1,(nely+1)*ones(nely,1));
    eleNode = repmat(eleN1(:),1,4)+repmat([0,nely+[1,2],1],nelx*nely,1);
    edofMat = kron(eleNode,[2,2])+repmat([-1,0],nelx*nely,4);
    iK = reshape(kron(edofMat,ones(8,1))',64*nelx*nely,1);
    jK = reshape(kron(edofMat,ones(1,8))',64*nelx*nely,1);
%% BOUNDARY CONDITION DEFINITION
    F = sparse(2*((nely+1)*nelx+ceil(nely/2)+1),1,-100,2*nNode,1);%NODAL LOADS
    fixeddofs = 1:1:2*(nely+1);%DISPLACEMENT CONSTRAINTS
    freedofs = setdiff(1:2*nNode,fixeddofs);
    U = zeros(2*nNode,1);
%% ITERATION OPTIMIZATION
    nLoop = 200; nRelax = 30;
    dt = 0.5; delta = 10; mu = 20; gamma = 0.05;
    comp = zeros(nLoop,1); vol = zeros(nLoop,1);
    for iT = 1:nLoop
      %% FINITE ELEMENT ANALYSIS
       [s,t] = meshgrid(-1:0.1:1,-1:0.1:1);
```

```
        tmpPhi = (1-s(:)).*(1-t(:))/4*Phi(eleNode(:,1))'+(1+s(:)).*(1-t(:))/4*...
                 Phi(eleNode(:,2))'+(1+s(:)).*(1+t(:))/4*Phi(eleNode(:,3))'+...
                (1-s(:)).*(1+t(:))/4*Phi(eleNode(:,4))';
        eleVol = sum(tmpPhi>=0,1)'/numel(s);
        vol(iT) = sum(eleVol)/(nelx*nely);
        sK = reshape(KE(:)*(Emin+eleVol'*(E0-Emin)),64*nelx*nely,1);
        K = sparse(iK,jK,sK); K = (K+K')/2;
        U(freedofs,1) = K(freedofs,freedofs)\F(freedofs,1);
        eleComp = sum((U(edofMat)*KE).*U(edofMat),2).*(Emin+eleVol*(E0-Emin));
        comp(iT) = sum(eleComp);
       %% DISPLAY RESULTS
        fprintf('No.%i,  Obj:%f,  Vol:%f\n',[iT,comp(iT),vol(iT)]);
        figure(1); contourf(Phi,[0,0]);
        colormap([0,0,0]); set(gcf,'color','w'); axis equal; axis off;
        figure(2); surf(Phi); caxis([-12,12]);
        axis equal; axis([0,nelx,0,nely,-12,12]); view(3);
        figure(3); subplot(2,1,1); plot(comp(1:iT),'-'); title('Compliance');
                   subplot(2,1,2); plot(vol(1:iT),'-'); title('Volume fraction');
       %% CONVERGENCE CHECK
        if iT>nRelax && abs(vol(iT)-volfrac)/volfrac<1e-3 && ...
           all(abs(comp(iT)-comp(iT-9:iT-1))/comp(iT)<1e-3)
           break;
        end
       %% LAGRANGE MULTIPLIER
        if iT<=nRelax
           lag = mu*(vol(iT)-vol(1)+(vol(1)-volfrac)*iT/nRelax);
        else
           lag = lag+gamma*(vol(iT)-volfrac);
           gamma = min(gamma+0.05,5);
        end
       %% LEVEL SET FUNCTION EVOLUTION
        gradPhi = sqrt((pGpX*Alpha).^2+(pGpY*Alpha).^2);
        indexDelta = (abs(Phi(:))<=delta);
        DeltaPhi = zeros(size(Phi));
        DeltaPhi(indexDelta) = 0.75/delta*(1-Phi(indexDelta).^2/delta^2);
        eleComp = reshape(eleComp,nely,nelx);
        eleCompLR = [eleComp(:,1),eleComp]+[eleComp,eleComp(:,end)];
        nodeComp = ([eleCompLR;eleCompLR(end,:)]+[eleCompLR(1,:);eleCompLR])/4;
        B = (nodeComp(:)/median(nodeComp(:))-lag).*DeltaPhi(:)*delta/0.75;
        Alpha = Alpha+dt*(G\[B;0;0;0]);
        Alpha = Alpha/mean(gradPhi(unique(eleNode((eleVol<1 & eleVol>0),:))));
        Phi = reshape(G(1:end-3,:)*Alpha,nely+1,nelx+1);
    end
end
```

## References


Allaire G (2009) A 2-d Scilab Code for shape and topology optimization by the level set method. http://www.cmap.polytechnique.fr/~allaire/levelset_en.html

Allaire G, Jouve F, Toader AM (2004) Structural optimization using sensitivity analysis and a level-set method. J Comput Phys 194(1): 363–393

Allaire G, De Gournay F, Jouve F, Toader AM (2005) Structural optimization using topological and shape sensitivity via a level set method. Control Cybern 34(1):59–80

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):1–16

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

Burger M, Osher SJ (2005) A survey on level set methods for inverse problems and optimal design. Eur J Appl Math 16(2):263–301

Burger M, Hackl B, Ring W (2004) Incorporating topological derivatives into level set methods. J Comput Phys 194(1):344–362

Cecil T, Qian J, Osher S (2004) Numerical methods for high dimensional Hamilton-Jacobi equations using radial basis functions. J Comput Phys 196(1):327–347

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

Chan TF, Vese LA (2001) Active contours without edges. IEEE Trans Image Process 10(2):266–277

Choi KK, Kim NH (2005) Structural sensitivity analysis and optimization 1. Springer, Berlin

Deaton JD, Grandhi RV (2014) A survey of structural and multidisciplinary continuum topology optimization: post 2000. Struct Multidiscip Optim 49(1):1–38

Dijk NPV, Maute K, Langelaar M, van Keulen F (2013) Level-set methods for structural topology optimization: a review. Struct Multidiscip Optim 48(3):437–472

Emre B, To AC (2015) Proportional topology optimization: a new non-sensitivity method for solving stress constrained and minimum compliance problems and its implementation in MATLAB. PLoS One 10(12):e0145041

Gain AL, Paulino GH (2013) A critical comparative assessment of differential equation-driven methods for structural topology optimization. Struct Multidiscip Optim 48(4):685–710

Huang X, Xie YM (2010) Evolutionary Topology Optimization of Continuum Structures: Methods and Applications. Wiley, New York

Kansa EJ, Power H, Fasshauer GE, Ling L (2004) A volumetric integral radial basis function method for time-dependent partial differential equations. I. Formulation. Eng Anal Bound Elem 28(10):1191–1206

Kharmanda G, Olhoff N, Mohamed A, Lemaire M (2004) Reliability-based topology optimization. Struct Multidiscip Optim 26(5):295–307

Liu K, Tovar A (2014) An efficient 3d topology optimization code written in MATLAB. Struct Multidiscip Optim 50(6):1175–1196

Mei Y, Wang X (2004) A level set method for structural topology optimization and its applications. Adv Eng Softw 35(7):415–441

Morse BS, Yoo TS, Chen DT, Rheingans P, Subramanian KR (2001) Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. Int Conf Shape Model Appl 15(2):89–98

Olesen LH, Okkels F, Bruus H (2006) A high-level programming-language implementation of topology optimization applied to steady-state navier-stokes flow. Int J Numer Methods Eng 65(7):975–1001

Osher S, Fedkiw R (2002) Level set methods and dynamic implicit surfaces. Springer, New York

Osher S, Santosa F (2001) Level set methods for optimization problems involving geometry and constraints: i. Frequencies of a two-density inhomogeneous drum. J Comput Phys 171(1):272–288

Osher S, Sethian JA (1988) Fronts propagating with curvature-dependent speed: algorithms based on HamiltonJacobi formulations. J Comput Phys 79(1):12–49

Otomori M, Yamada T, Izui K, Nishiwaki S (2014) MATLAB code for a level set-based topology optimization method using a reaction diffusion equation. Struct Multidiscip Optim 51(5):1159–1172

Peng D, Merriman B, Osher S, Zhao H.& Kang M. (1999) A PDE-based fast local level set method. J Comput Phys 155(2):410–438

Rochafellar RT (1973) The multiplier method of Hestenes and Powell applied to convex programming. J Optim Theory Appl 12:555–562

Schmidt S, Schulz V (2011) A 2589 line topology optimization code written for the graphics card. Comput Vis Sci 14(6):249–256

Sethian JA (1999) Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge University Press, Cambridge

Sethian JA, Wiegmann A (2000) Structural boundary design via level set and immersed interface methods. J Comput Phys 163(2):489–528

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

Sigmund O, Maute K (2013) Topology optimization approaches. Struct Multidiscip Optim 48(6):1031–1055

Sokołowski J, Żochowski A (1999) On the topological derivative in shape optimization. SIAM J Control Optim 37(4):1251–1272

Sokołowski J, Zolésio JP (1992) Introduction to shape optimization: shape Sensitity analysis. Introduction to shape optimization : shape sensitivity analysis. Springer-Verlag, Berlin

Suresh K (2010) A 199-line MATLAB code for Pareto-optimal tracing in topology optimization. Struct Multidiscip Optim 42(5):665–679

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

Tavakoli R, Mohseni SM (2014) Alternating active-phase algorithm for multimaterial topology optimization problems: a 115-line MATLAB implementation. Struct Multidiscip Optim 49(4):621–642

Van Dijk NP, Maute K, Langelaar M, Van Keulen F (2013) Level-set methods for structural topology optimization: a review. Struct Multidiscip Optim 48(3):437–472

Wang SY, Wang MY (2006a) Radial basis functions and level set method for structural topology optimization. Int J Numer Meth Engng 65: 2060–2090

Wang SY, Wang MY (2006b) Structural shape and topology optimization using an implicit free boundary parameterization method. Comput Model Eng Sci 13(2):119–147

Wang MY, Wei P (2005) Topology optimization with level set method incorporating topological derivative. 6th World Congress on Structural & Multidisciplinary Optimization, Rio de Janeiro, Brazil

Wang MY, Wang X, Guo D (2003) A level set method for structural topology optimization. Comput Methods Appl Mech Eng 192(1–2):227–246

Wang MY, Chen SK, Xia Q (2004) TOPLSM, 199-line version. http://ihome.ust.hk/~mywang/download/TOPLSM_199.m

Wang SY, Lim KM, Khoo BC, Wang MY (2007) An extended level set method for shape and topology optimization. J Comput Phys 221(1):395–421

Wei P, Wang MY (2006) The augmented Lagrangian method in structural shape and topology optimization with RBF based level set method, The 4th China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kunming, China

Wei P, Wang MY (2009) Piecewise constant level set method for structural topology optimization. Int J Numer Methods Eng 78(4):379–402

Wendland H (1995) Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv Comput Math 4(1):389–396

Xia L, Breitkopf P (2015) Design of materials using topology optimization and energy-based homogenization approach in MATLAB. Struct Multidiscip Optim 52(6):1229–1241

Xie X, Mirmehdi M (2011) Radial basis function based level set interpolation and evolution for deformable modelling. Image Vis Comput 29(2–3):167–177

Yamada T, Izui K, Nishiwaki S, Takezawa A (2010) A topology optimization method based on the level set method incorporating a fictitious interface energy. Comput Methods Appl Mech Eng 199(45–48):2876–2891

Zegard T, Paulino GH (2015) GRAND3 - ground structure based topology optimization for arbitrary 3D domains using MATLAB. Struct Multidiscip Optim 52(6):1161–1184

Zhang W, Yuan J, Zhang J, Guo X (2016) A new topology optimization approach based on moving Morphable components (MMC) and the ersatz material model. Struct Multidiscip Optim 53(6):1243–1260

Zhao HK, Chan T, Merriman B, Osher S (1996) A variational level set approach to multiphase motion. J Comput Phys 127:179–195

Zhou S, Cadman J, Chen Y, Li W, Xie YM, Huang X et al (2012) Design and fabrication of biphasic cellular materials with transport properties – a modified bidirectional evolutionary structural optimization procedure and MATLAB program. Int J Heat Mass Transf 55(25–26):8149–8162

Zuo ZH, Xie YM (2015) A simple and compact python code for complex 3D topology optimization. Adv Eng Softw 85:1–11