Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 369 (2020) 113169
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Overhang control based on front propagation in 3D topology optimization for additive manufacturing

Emiel van de Ven[a,b,*], Robert Maas[b], Can Ayas[a], Matthijs Langelaar[a], Fred van Keulen[a]

[a] *Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands*
[b] *Netherlands Aerospace Centre, Anthony Fokkerweg 2, 1059 CM, Amsterdam, The Netherlands*

Received 8 February 2020; accepted 21 May 2020
Available online 22 June 2020

Abstract

It is attractive to combine topology optimization (TO) with additive manufacturing (AM), due to the design freedom provided by AM, and the increased performance that can be achieved with TO. One important aspect is to include the design rules associated with the process restrictions of AM to prevent the requirement of relatively large support volumes during printing. This paper presents a TO filter that enforces a minimum overhang angle, resulting in an optimized topology that is printable without the need for support structures. The filter is based on front propagation, which, as it is described by a PDE, allows for a straightforward application on unstructured meshes, to enforce an arbitrary overhang angle. Efficient algorithms developed for front propagation are used in combination with adjoint sensitivities, in order to have a minor influence on the total computational cost. The focus of this work is on the implementation of the filter for high resolution 3D cases, which requires development of the front propagation for tetrahedral elements, and its parallelization.

Keywords: Topology optimization; Additive manufacturing; Overhang; Front propagation

1. Introduction

Additive manufacturing (AM) offers tremendously more design freedom compared to conventional manufacturing techniques, and geometric complexity has a much lesser relative impact on production cost. Therefore, it is frequently linked to topology optimization (TO), a computational design method that generates optimized designs for given objective and constraints, but often results in complex, organically shaped designs, which were difficult to manufacture until the emergence of AM [1,2].

However, in practice, AM is not completely free of manufacturing constraints. Many studies have been performed to identify design rules for AM [3–7]. Design for manufacturing practice states that ignoring manufacturing constraints during the design process will lead to extra costs during manufacturing [8]. When a topology optimized design is modified after optimization to incorporate the manufacturing constraints, optimality is mostly

* Corresponding author at: Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands.
E-mail address: emiel@emielvandeven.nl (E. van de Ven).

https://doi.org/10.1016/j.cma.2020.113169

compromised. Therefore, the AM constraints should be included into the topology optimization to retain optimality while adhering to the manufacturing constraints.

The AM constraint that is the focus of this study is the overhang limitation. This constraint arises from the fact that each consecutive layer that is printed requires a certain amount of overlap with the previous layer, in order to have sufficient mechanical support and/or heat dissipation [9]. Therefore, the distance with which each layer can "overhang" the previous layer is limited. The degree of overhang is usually measured as the angle between a downward-facing surface and the base plate. The minimum allowable overhang angle α_{oh} depends on the type of process and material, hence a general overhang constraint should be able to handle a range of minimum overhang angles [10]. In the remainder of this paper, surfaces are termed overhanging if they have an overhang angle smaller than the minimum overhang angle.

Incorporation of AM constraints into topology optimization has recently become an active field of study. Various papers have presented 2D approaches [11–15], but relevance to industrial practice requires methods to be highly effective in general 3D settings. Therefore we shall focus our discussion on the studies that show a 3D implementation of the overhang constraint. For a more comprehensive review the reader is referred to [16]. Generally, the overhang constraints can be classified into three categories: local boundary angle control, geometrical AM process modelling, and physics-based AM modelling.

Constraints in the first category attempt to detect the topological boundary and constrain the overhang angle locally. This has been applied in [17,18] and [19].

The geometrical AM process modelling constraints also enforce a given overhang angle, but do so by scanning the topology in the print order to detect overhanging areas — crudely mimicking the printing process [20–23]. Finally, the physics-based constraints incorporate a more elaborate model of the printing process, by modelling the manufacturing, e.g., as a series of self-weight loads [19,24].

The local boundary angle control methods usually converge to sub-optimal local minima, generating saw-tooth like structures that are not manufacturable, whereas the physics-based constraints, although potentially providing more details, are numerically expensive since they generally involve one or more finite element analyses to model the printing process. Therefore, this study presents an overhang constraint of the geometrical AM process modelling type. Compared to the existing methods, it can handle unstructured meshes and variable overhang angles naturally, as opposed to the work of [20] which would require an additional mapping as detailed in [25]. Furthermore, there is no additional non-linearity introduced by a heaviside projection filter [23], and no additional filtering is required to prevent floating supports [21]. Finally, it can be implemented in any density based topology optimization as opposed to [22] which uses a custom topology description.

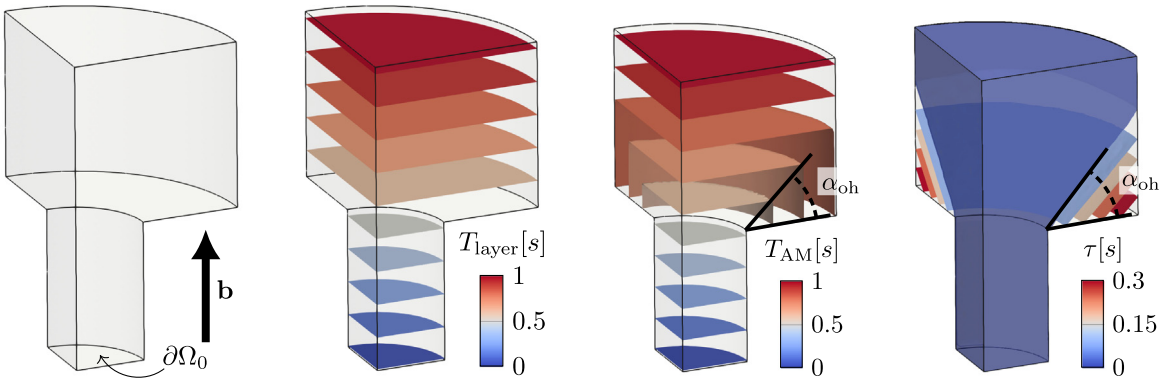
This paper presents the extension to 3D of the front propagation based overhang filter presented in [15]. While our earlier 2D implementation validated the concept of detecting and eliminating overhang using front propagation, the 3D implementation allows true 3D high-resolution design for AM. The non-trivial steps required in expanding from 2D to 3D are (i) efficiently propagating the front on the element level, and (ii) parallelizing the front propagation. For these challenges novel solutions are proposed. Furthermore, improvements have been made to control the length scale of the overhang filtered design. The paper is organized as follows. In Section 2 the method of overhang detection and the incorporation of the filter in TO is explained. Section 3 provides the details of the numerical implementation in 3D. Numerical results are presented in Section 4, and Section 5 concludes this paper.

2. Method

2.1. Detecting overhang using front propagation

The idea of using front propagation for overhang detection originates from the printing process itself, which can be seen as a front evolving with each printed layer. Similar to the real printing process, the front propagation is initiated at the base plate, defined by a boundary $\partial\Omega_0$. The front propagation can be described by an arrival-time field $T(\mathbf{x})$, which represents the pseudo-time at which the front reaches to location $\mathbf{x}$. The core idea of our method is to construct an arrival-time field in a 3D design that distinguishes printable regions from overhanging ones.

Consider the structure given in Fig. 1a on which the overhanging region should be identified, when printed in the build direction indicated by $\mathbf{b}$. The first arrival-time field that is required for overhang detection is termed the layer arrival-time field, T_{layer} (Fig. 1b). It describes the sequence of printing, as each isosurface of this field represents a printing layer. The arrival time of each layer is a measure of distance to the base plate, as can be seen in Fig. 1b.



(a) Part of a domain within which overhanging areas are to be detected.

(b) Isosurfaces of the T_{layer} arrival time field.

(c) Isosurfaces of the T_{AM} arrival time field.

(d) Isosurfaces of the delay field τ, with indication of the minimum overhang angle α_{oh}.

Fig. 1. Overhang can be detected in an arbitrary geometry by calculating the delay τ of the T_{AM} field w.r.t. the T_{layer} field. The regions with zero delay are not overhanging (the blue region in Fig. 1d). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Subsequently, a second arrival-time field, T_{AM}, is constructed that includes information on the overhang limitation of an actual AM process (Fig. 1c). It is constructed by using a front propagation that results in $T_{\text{AM}} = T_{\text{layer}}$ in printable regions, but delays the propagation when the direction of propagation is lower than a given overhang angle α_{oh} (i.e. violating the overhang limitation). By doing so, the isosurfaces of T_{AM} are not flat as T_{layer}, but bend around corners, as displayed in Fig. 1c.

The final step to detect the overhanging regions is to compare the two arrival-time fields T_{layer} and T_{AM}. The delay field τ of the T_{AM} field is defined as

$$\tau(\mathbf{x}) = T_{AM}(\mathbf{x}) - T_{\text{layer}}(\mathbf{x}). \tag{1}$$

When $\tau(\mathbf{x}) > 0$, the AM front is delayed compared to the layer front, which implies that the AM front has propagated in a direction lower than α_{oh}. Consequently, overhanging regions are identified as those regions where $\tau(\mathbf{x}) > 0$. This can be seen in Fig. 1d, where the region $\tau(\mathbf{x}) = 0$ is manufacturable, and the rest is overhanging and, thus, cannot be manufactured. Furthermore, the value of $\tau(\mathbf{x})$ is a measure for the distance to the closest manufacturable region, giving an indication of how much material it would require to support a certain location. This continuous measure of overhang is beneficial for gradient-based optimization used in topology optimization.

2.2. Constructing the arrival-time fields

The construction of T_{layer} is straightforward. Since the arrival time is a measure of distance to the base plate, this field is defined as:

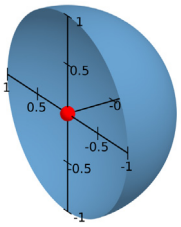
$$T_{\text{layer}}(\mathbf{x}) = \frac{\mathbf{x} \cdot \mathbf{b}}{f_0}, \tag{2}$$

where $\mathbf{b}$ is a unit vector pointing in the build direction, f_0 the propagation speed, which can be interpreted as the printing rate, and assuming that the origin of the coordinate system is on the base-plate.

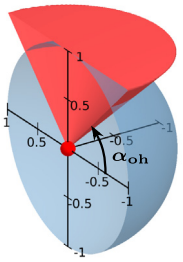
For the second arrival-time field, T_{AM}, the front propagation can be written as a boundary value problem and is governed by the Hamilton–Jacobi–Bellman equation:

$$\begin{aligned} &T(\mathbf{x}) = T_0(\mathbf{x}), && \mathbf{x} \in \partial\Omega_0, \\ &\min_{\mathbf{a} \in S_1} \{(\nabla T(\mathbf{x}) \cdot \mathbf{a}) f_s(\mathbf{x}, \mathbf{a}, \alpha_{oh})\} = 1, && \mathbf{x} \in \Omega, \end{aligned} \tag{3}$$

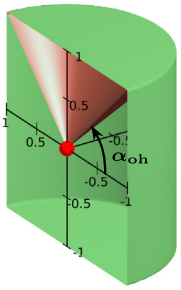
where Ω is the domain under consideration, $T(\mathbf{x})$ is the arrival time at location $\mathbf{x}$, fixed at T_0 at the boundary $\partial\Omega_0$, $\mathbf{a}$ is a unit vector indicating the local propagation direction: $\mathbf{a} \in S_3$, $S_3 = \left\{\mathbf{a} \in \mathbf{R}^3 \mid \|\mathbf{a}\| = 1\right\}$, and $f_s(\mathbf{x}, \mathbf{a}, \alpha_{oh})$ is



(a) Spherical speed function.



(b) Conic speed function (red), with spherical speed function overlay (blue).



(c) Cylindrical speed function (green), with conic speed function overlay (red).

Fig. 2. The depicted surfaces indicate the speed with which the front propagates in each direction from the origin (red dot). The surfaces are cut in half for visualization purposes, but are in fact rotationally symmetric about the vertical axis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a speed function, giving the propagation speed for a given location, propagation direction and overhang angle. The speed function is decomposed in a part dependent on the location, and a part dependent on direction of propagation:

$$f_s(\mathbf{x}, \mathbf{a}, \alpha_{oh}) = \phi(\mathbf{x}) f(\mathbf{a}, \alpha_{\text{oh}}). \tag{4}$$

Let us for now ignore the location dependence, which is detailed in Section 2.3, and focus on the speed function $f(\mathbf{a}, \alpha_{oh})$. In order to be able to detect overhang, the speed function f should be chosen with care. The simplest speed function, used for isotropic front propagation (e.g. wave propagation in isotropic media), is $f = c$, where c represents a constant propagation speed irrespective of the propagation direction. This is depicted in Fig. 2a, where the distance from the origin (the red dot) to the surface gives the propagation speed in each direction. However, it is difficult to detect overhang with this speed function, as no information on the minimum overhang angle is governed by this speed function.

For a front that mimics the printing process, a conic propagation profile, as depicted in Fig. 2b, is suitable. It can be seen that the front propagation speed reduces to zero below the minimum overhang angle, and the propagation speed increases when the front propagates in directions other then the build direction, to compensate for the larger distance it travels to reach the next layer. Basically, the cone represents the region that can be built when starting from a single point. Propagating a front with this speed function would indicate the printable regions in a topology. Unfortunately, this speed function is numerically difficult to propagate, as it has a large anisotropy and a zero sideways velocity, which would result in infinitely large arrival times.

For numerical convenience, the conic speed function is changed to a cylindrical speed function as displayed in Fig. 2c. Compared to the conic speed function it has the same profile for propagation directions above the minimum overhang angle. For other directions, the speed is not set to zero, but to a finite value governed by the surface of the cylinder. This means the front can still propagate in non-manufacturable regions, albeit at a slower speed, and hence it will be delayed. Next to being numerically more tractable compared to the conic speed profile, the cylindrical speed function also has the benefit of yielding information on the severity of the overhang in terms of the delay, which will be utilized during the optimization. The cylindrical speed function for a given overhang angle α_{oh} and propagation direction $\mathbf{a}$ is described by the following equation (see [15]):

$$f(\mathbf{a}, \alpha_{oh}) = \frac{f_0}{\max\left(\tan(\alpha_{oh}) \left\|\mathbf{Pa}\right\|, \left|\mathbf{b} \cdot \mathbf{a}\right|\right)}, \tag{5}$$

where $\mathbf{P}$ is the projection on the plane orthogonal to $\mathbf{b}$, defined as $\mathbf{P} = (\mathbf{I} - \mathbf{b} \otimes \mathbf{b})$, with $\mathbf{x} \otimes \mathbf{y}$ denoting the outer product between the vectors $\mathbf{x}$ and $\mathbf{y}$. Propagating a front with this cylindrical speed function gives the required arrival-time field T_{AM}, as shown in Fig. 1c.

2.3. Incorporation in topology optimization

Thus far, the front propagation was carried out on a given part domain. However, with topology optimization the geometry is implicitly defined. In this study, density-based topology optimization is used [26], where the topology

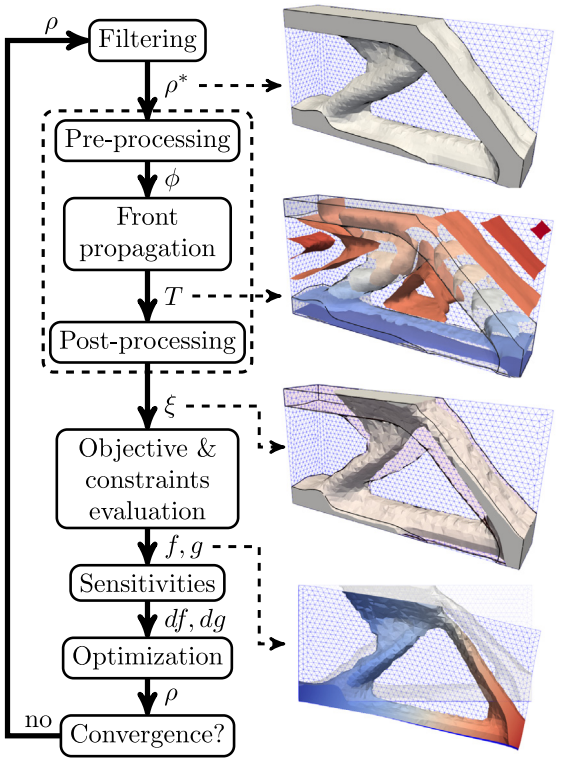


Fig. 3. Schematic of the TO algorithm with overhang filter, indicated by the dashed rectangle. The fields ρ^*, T, and ξ have been depicted for a typical TO result. Note that the objective and constraints are evaluated on the printable design ξ, as shown in the bottom figure.

is defined by a pseudo-density field $\rho(\mathbf{x})$, which indicates for each location if it is either void ($\rho = 0$), contains material ($\rho = 1$), or has an intermediate value ($0 < \rho < 1$). The topology optimization algorithm with overhang filter is schematically depicted in Fig. 3. In the first step, the pseudo-density field ρ is filtered to control length scale and to prevent checkerboarding [27]. The filtered densities ρ^* are thus a weighted average of the densities in their surrounding region up to the filter radius r [28].

The filtered densities ρ^* are input to the overhang filter, which results in the printable density field ξ, a density field similar to ρ^* but with the overhanging regions removed. The objective and constraint evaluation is then carried out on ξ, instead of the filtered densities ρ^* one would normally use.

The overhang filter comprises three steps. First, the filtered densities are processed to serve as a scaling field ϕ for the speed of the front propagation. The purpose of the speed scaling field ϕ is to only allow the front to propagate through material regions, and to delay it in void regions. With this step, the geometry given by the density field ρ^* is coupled into the front propagation; otherwise, the front propagation would not be influenced by the density field at all. For simplicity a linear relation is chosen, and the speed scaling field ϕ is defined as

$$\phi(\mathbf{x}) = v_{\text{void}} + (1 - v_{\text{void}})\rho^*(\mathbf{x}), \tag{6}$$

where v_{void} ($0 < v_{\text{void}} < 1$) is the scaling of the propagation speed in void regions, which is typically chosen as 0.5. As stated in Eq. (4), the speed function for the front propagation (Eq. (5)) is scaled linearly with ϕ:

$$f_s(\mathbf{x}, \mathbf{a}, \alpha_{oh}) = \phi(\mathbf{x}) f(\mathbf{a}, \alpha_{\text{oh}}). \tag{7}$$

Then, in the second step, the front propagation is performed with the scaled speed function f_s, to obtain the arrival-time field T_{AM}.

In the last step of the overhang filter, the arrival-time field T_{AM} is post-processed to obtain the printable density field ξ. First, the delay is evaluated (Eq. (1)):

$$\tau(\mathbf{x}) = T_{\text{AM}}(\mathbf{x}) - T_{\text{layer}}(\mathbf{x}), \tag{8}$$

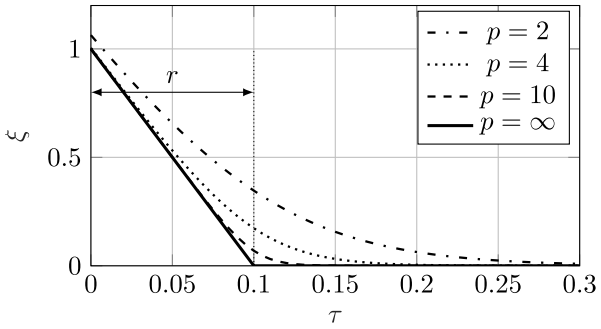


Fig. 4. Relation between the delay τ and the printable density field ξ, for different p with a filter radius $r = 0.1$.

which is transformed into the printable density field ξ by a function h:

$$\xi(\mathbf{x}) = h(\tau(\mathbf{x})). \tag{9}$$

Since manufacturable regions are defined as those regions where the delay $\tau = 0$ (Eq. (1)), h should be such that $h(0) = 1$. Regions with a delay $\tau > 0$ are not manufacturable or void in the original design, and therefore h should decrease towards zero for increasing values of τ. The choice of h dictates the transition of material to void in the printable density field. In order to retain the original length scale of the density filter with a filter radius of r, the following relation has been chosen:

$$h(\tau(\mathbf{x})) = \text{smax}_p\left(0, 1 - \frac{\tau(\mathbf{x}) v_{\text{void}}}{r}\right), \tag{10}$$

$$\text{smax}_p(a, b) = \frac{1}{p}\ln\left(\exp(pa) + \exp(pb)\right), \tag{11}$$

where smax_p is a smooth maximum operator, and p determines the smoothness. The relation between ξ and τ is displayed in Fig. 4. In the manufacturable regions where $\tau = 0$, the printable density becomes $\xi = 1$, and for higher values of p, ξ decreases linearly towards 0 with increasing delay, comparable to a density filter [28]. As can be seen, p should not be chosen too small, as that will result in printable densities significantly larger than one for $\tau = 0$. In this study, p is chosen as $p = 10$.

The relation specified in Eq. (10) implies that when the front is propagating at the void speed v_{void}, the transition from $\xi = 1$ to $\xi = 0$ will take place within the original filter radius, as depicted in Fig. 4. However, this is not true for every transition from $\xi = 1$ to $\xi = 0$. First of all, due to the original density filter, the propagation speed will not decrease to v_{void} instantaneously, but gradually ramp off, and second, the propagation speed is not equal in the build direction and in directions orthogonal to the build direction if $\alpha_{\text{oh}} \neq 45°$. Therefore, the length scale can be influenced by the application of the overhang filter. If an exact length scale is required, the overhang filter can be applied before the density filter. However, this will reintroduce some overhang due to rounding of the density filter, and therefore the density filter is applied first in this work.

With h specified in Eq. (10), the prescribed values T_0 for the front propagation at the boundary can be determined (Eq. (3)). Since material points at the base plate $\partial\Omega_0$ are printable, it should hold that

$$\xi(\mathbf{x}) = \rho^*(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega_0. \tag{12}$$

At the base plate, $T_{\text{layer}} = 0$, and thus $\xi = h(\tau) = h(T_0)$. To satisfy Eq. (12), the initial arrival time at the boundary is chosen as

$$T_0(\mathbf{x}) = h^{-1}(\rho^*(\mathbf{x})), \quad \mathbf{x} \in \partial\Omega_0, \tag{13}$$

$$= \frac{-r}{v_{\text{void}}}\left(\ln\left(\exp(\rho^*(\mathbf{x})p)\right)/p - 1\right), \tag{14}$$

where h^{-1} is the inverse function of h such that $h^{-1}(h(\rho^*(\mathbf{x}))) = \rho^*(\mathbf{x})$. Note that due to the logarithm, $T_0 = \infty$ for $\rho^* = 0$. If infinite values cannot be used, T_0 should be set to an arbitrary high value wherever $\rho^* = 0$.

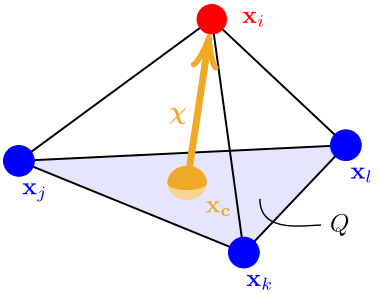


Fig. 5. In the local update, the arrival time of a node $\mathbf{x}_i$ is calculated from three nodes with known arrival times, by finding a location $\mathbf{x}_\mathbf{c}$ on the triangle Q spanned by these nodes, that minimizes the arrival time at $\mathbf{x}_i$.

3. Numerical implementation

The two main challenges of the 3D implementation of the overhang filter are (i) parallelizing the front propagation problem to achieve the desired scalability with problem size, and (ii) efficiently propagating the front on element level, i.e. the local arrival time update, which is a key component in the numerical front propagation scheme. This section is intended to aide with the numerical implementation of the method and mainly concerns implementation details. Readers who want to obtain a general overview of the front propagation-based overhang filter can choose to continue with the numerical results in Section 4.

3.1. Updating local arrival time in 3D

Most front propagation algorithms contain a function where within an element the arrival time of one node is calculated based on the known arrival time of the other nodes. This local update is used to propagate the front throughout the domain, as with each newly calculated node, arrival times for other unknown nodes can now be computed. This update is performed multiple times for each node from different directions, up to the number of elements that contain the node. Eventually, the update resulting in the lowest arrival time is accepted. Because of the multiple evaluations per node, it is the function that is called the most times in the propagation algorithm and thus it is essential that the update is numerically inexpensive. The local update is the only part of the algorithm that is different from the 2D implementation presented in [15], although a similar approach is taken to evaluate the local update.

In the following, the local update is detailed for a tetrahedron, as any other polygonal element can be constructed from it. It is possible to define the local update for other element types as well. Consider a tetrahedron $\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k\mathbf{x}_l$ as displayed in Fig. 5, where the arrival time T_i at node $\mathbf{x}_i$ is unknown, and the arrival times T_j, T_k, and T_l on the remaining nodes are known. The triangle $Q = \mathbf{x}_j\mathbf{x}_k\mathbf{x}_l$ spanned by the nodes with known arrival times. If these arrival times are equal, Q is the front of the propagation. A point $\mathbf{x}_\mathbf{c}$ on this triangle can be defined in parametric form:

$$\mathbf{x}_\mathbf{c}(\zeta_1, \zeta_2) = \mathbf{x}_j + \zeta_1(\mathbf{x}_k - \mathbf{x}_j) + \zeta_2(\mathbf{x}_l - \mathbf{x}_j), \tag{15}$$

with $0 \leq \zeta_1 \leq 1$, $0 \leq \zeta_2 \leq 1$ and $\zeta_1 + \zeta_2 \leq 1$. The arrival time at $\mathbf{x}_\mathbf{c}$ on the front is a linear interpolation between the known nodes and is defined as

$$T_\mathbf{c}(\zeta_1, \zeta_2) = T_j + \zeta_1(T_k - T_j) + \zeta_2(T_l - T_j). \tag{16}$$

Now, the arrival time T_i, assuming it is updated from $\mathbf{x}_\mathbf{c}$, can be calculated as the distance $\|\boldsymbol{\chi}\|$ between both points divided by the propagation speed f_s, plus the arrival time at $\mathbf{x}_\mathbf{c}$, $T_\mathbf{c}$:

$$T_i = \frac{\|\boldsymbol{\chi}\|}{f_s(\phi_i, \mathbf{a}, \alpha_{oh})} + T_\mathbf{c}, \tag{17}$$

where $\boldsymbol{\chi} = \mathbf{x}_i - \mathbf{x}_\mathbf{c}$, $\mathbf{a}$ is the direction of propagation defined as $\mathbf{a} = \boldsymbol{\chi} / \|\boldsymbol{\chi}\|$ (Fig. 5), and $f_s(\phi_i, \mathbf{a}, \alpha_{oh})$ is the speed function as defined in Eq. (7). Note that the propagation speed is only dependent on the speed scaling value ϕ_i, and thus density ρ_i, of the target node. This is done to simplify the front propagation and sensitivity evaluation, as it introduces less dependencies as compared to interpolating the speed scaling values ϕ of all the nodes in the element.

What remains is to determine from which point $\mathbf{x}_\mathbf{c}$ on the triangle Q the arrival time of $\mathbf{x}_i$ should be updated. Consequently, the local update is essentially a minimization problem where a point $\mathbf{x}_\mathbf{c}$ on triangle Q is sought that minimizes the arrival time T_i at the unknown node. Following [29], the equation for the local update for node $\mathbf{x}_i$ as updated from triangle Q is given by

$$T^i_{jkl} = \min_{\zeta_1,\zeta_2:\mathbf{x}_\mathbf{c}\in Q} \left\{ \frac{\|\boldsymbol{\chi}\|}{f_s(\phi_i, \mathbf{a}, \alpha_{oh})} + T_\mathbf{c} \right\}, \tag{18}$$

where T^i_{jkl} is the arrival time at node i when evaluated from nodes j, k, and l. The final arrival time of node is the minimum of its local updates. The remainder of this section will detail the solving of the minimization problem posed in Eq. (18). First, a list of potential locations that minimize Eq. (18) is determined, by carefully examining the possible scenarios in the minimization problem posed by the maximum and absolute function present in the speed function f_s (Eq. (7)). Then, the arrival times resulting from these locations are evaluated and the minimum arrival time is selected.

3.1.1. A closer look at the local update function

The minimization problem posed by Eq. (18) is first solved on the plane that contains the triangle Q. It is later evaluated if the minimum is actually inside Q (i.e. $0 \leq \zeta_1 \leq 1$, $0 \leq \zeta_2 \leq 1$ and $\zeta_1 + \zeta_2 \leq 1$). When the speed function f_s as defined in Eq. (7) is substituted in Eq. (18), it can be rewritten as

$$T^i_{jkl} = \min_{\zeta_1,\zeta_2:\mathbf{x}_\mathbf{c}\in Q} \left\{ \frac{\max\left(\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\|\,, |\mathbf{b}\cdot\boldsymbol{\chi}|\right)}{f_0\phi_i} + T_\mathbf{c} \right\}. \tag{19}$$

Here $|\mathbf{b}\cdot\boldsymbol{\chi}|$ represents the length of $\boldsymbol{\chi}$ projected on the build direction $\mathbf{b}$, and $\|\mathbf{P}\boldsymbol{\chi}\|$ represents the length of $\boldsymbol{\chi}$ projected on the plane orthogonal to the build direction. This can be seen as a right-angled triangle, where $\boldsymbol{\chi}$ is the hypotenuse, and $|\mathbf{b}\cdot\boldsymbol{\chi}|$ and $\|\mathbf{P}\boldsymbol{\chi}\|$ are the lengths of the legs, as displayed in Fig. 6.

Next, the maximum function in Eq. (19) can be interpreted as follows. Let C be a double cone with vertex $\mathbf{x}_i$, its axis parallel to the build direction $\mathbf{b}$ and its aperture such that the generatrices of the cone make an angle α_{oh} with the base plate (see Fig. 6). The following relations apply:

$$\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\| = |\mathbf{b}\cdot\boldsymbol{\chi}|\,, \qquad \mathbf{x}_\mathbf{c} \in \partial C \tag{20}$$

$$\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\| < |\mathbf{b}\cdot\boldsymbol{\chi}|\,, \qquad \mathbf{x}_\mathbf{c} \in C \tag{21}$$

$$\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\| > |\mathbf{b}\cdot\boldsymbol{\chi}|\,, \qquad \mathbf{x}_\mathbf{c} \notin C. \tag{22}$$

This implies that inside the cone (the red area in Fig. 6), $|\mathbf{b}\cdot\boldsymbol{\chi}|$ is dominant, and outside the cone $\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\|$ is dominant. This is also drawn for 2D in Fig. 7, where it can be seen that for any plane, indicated with the blue line, that does not contain $\mathbf{x}_i$, $|\mathbf{b}\cdot\boldsymbol{\chi}|$ is minimized outside the cone, and $\|\mathbf{P}\boldsymbol{\chi}\|$ is minimized inside the cone. The maximum function is thus minimized on the cone edge ∂C when $\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\| = |\mathbf{b}\cdot\boldsymbol{\chi}|$.

Besides the maximum term, Eq. (19) also contains the interpolation of the known arrival times $T_\mathbf{c}$, which adds a linear field to the maximum term. Since the linear field $T_\mathbf{c}$ has no interior optimum (Eq. (16)), the minimum of Eq. (19) will either remain on the cone edge ∂C when the gradient of this linear field is small, or the minimum will be at the bounds of the domain when the gradient of the linear field dominates. Based on these observations, the minimization problem is examined for two possibilities:

1. The maximum term is dominant. The minimum occurs at the cone edge ∂C.
2. The arrival time interpolation term is dominant. The minimum occurs outside of the cone C, on the edge of the triangle Q.

Both scenarios are discussed subsequently.

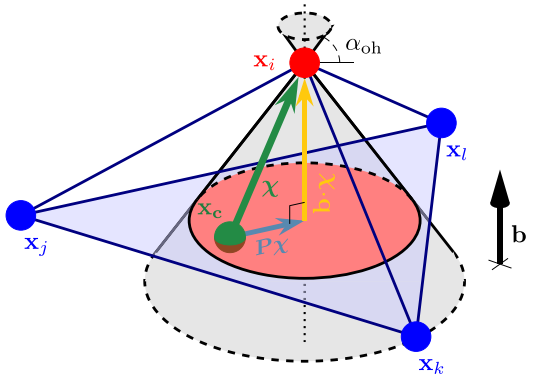


Fig. 6. Given a point $\mathbf{x_c}$ on the triangle $Q = \mathbf{x}_j\mathbf{x}_k\mathbf{x}_l$, the arrival time at $\mathbf{x}_i$ is dependent on either the distance $|\mathbf{b}\cdot\boldsymbol{\chi}|$ inside the given cone, or $\|\mathbf{P}\boldsymbol{\chi}\|$ outside this cone, due to the maximum operator between the two (Eq. (19)).

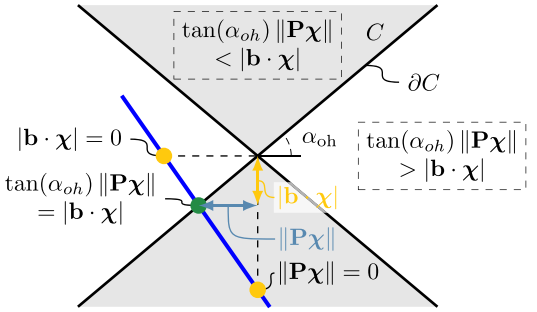


Fig. 7. For any plane, indicated in blue, that does not contain $\mathbf{x}_i$, both $|\mathbf{b}\cdot\boldsymbol{\chi}|$ and $\|\mathbf{P}\boldsymbol{\chi}\|$ are minimized at a location where they are not dominant in the maximum term of Eq. (19). The maximum term is thus minimized at the cone edge ∂C.

Minimum on cone edge. On the cone edge, both terms that appear on the maximum function in Eq. (19) are equal, and either one can be chosen to minimize over the cone edge. Since $|\mathbf{b}\cdot\boldsymbol{\chi}|$ is piecewise linear, it is preferred over the quadratic function $\|\mathbf{P}\boldsymbol{\chi}\|$. First, this function is minimized on the plane defined by the triangle Q. The update can be cast in the following form (see Appendix for details)

$$\begin{aligned} \min_{\mathbf{y}} \quad & \mathbf{g}^T\mathbf{y} \\ \text{s.t.} \quad & \mathbf{y}^T\mathbf{K}\mathbf{y} + \mathbf{l}^T\mathbf{y} + c_6 = 0, \end{aligned} \tag{23}$$

where $\mathbf{g}$ contains the gradients of the update functions inside the cone, $\mathbf{K}$, $\mathbf{l}$ and c_6 are parameters defined by the geometry of the cone–plane intersection, and $\mathbf{y} = \begin{bmatrix}\zeta_1 & \zeta_2\end{bmatrix}^T$. This formulation minimizes a linear function on a cone–plane intersection, and can be solved explicitly using Lagrange multipliers (Appendix).

If the minimum found by solving Eq. (23) is not inside the triangle Q, the minimum might occur on the intersection of an edge of Q with the cone. In order to find the edge–cone intersections, an algorithm described in [30] is used, which solves the following quadratic equation for a given edge $\mathbf{x}_j\mathbf{x}_k$ and unknown $\mathbf{x}_i$:

$$c_2\gamma^2 + c_1\gamma + c_0 = 0, \tag{24}$$

with $c_2 = \bar{\mathbf{a}}^T M\bar{\mathbf{a}}$, $c_1 = 2\bar{\mathbf{a}}^T M\bar{\mathbf{b}}$, $c_0 = \bar{\mathbf{b}}^T M\bar{\mathbf{b}}$, $M = \mathbf{b}\otimes\mathbf{b} - \sin^2(\alpha_{oh})I$, $\bar{\mathbf{a}} = \mathbf{x}_k - \mathbf{x}_j$, and $\bar{\mathbf{b}} = \mathbf{x}_j - \mathbf{x}_i$. This gives two solutions for γ, from which the minimum locations are calculated with $\mathbf{x_c} = \gamma\mathbf{x}_j + (1-\gamma)\mathbf{x}_k$.

Minimum outside cone. If the minimum lies outside of the cone, the following equation is to be minimized on each edge of triangle Q:

$$T^i_{jk} = \min_{\gamma\in[0,1]} \left\{ \frac{\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\|}{f_0\phi_i} + \gamma T_j + (1-\gamma)T_k \right\}. \tag{25}$$

This equation can be rewritten as

$$T^i_{jk} = \min_{\gamma\in[0,1]} \left\{ \hat{d}\sqrt{\gamma^2\hat{a} + \gamma\hat{b} + \hat{c}} + \gamma\hat{e} + \hat{f} \right\} \tag{26}$$

with $\hat{a} = \hat{\mathbf{a}}\cdot\hat{\mathbf{a}}$, $\hat{b} = 2\hat{\mathbf{a}}\cdot\hat{\mathbf{b}}$, $\hat{c} = \hat{\mathbf{b}}\cdot\hat{\mathbf{b}}$, $\hat{d} = \tan(\alpha_{oh})/(f_0\phi_i)$, $\hat{e} = T_j - T_k$, $\hat{f} = T_k$, $\hat{\mathbf{a}} = \mathbf{P}(\mathbf{x}_k - \mathbf{x}_j)$, and $\hat{\mathbf{b}} = \mathbf{P}(\mathbf{x}_i - \mathbf{x}_k)$. The stationary points are found at

$$\gamma = \frac{\hat{b}\hat{e}^2 - \hat{a}\hat{b}\hat{d}^2 \pm \hat{e}\sqrt{(\hat{a}\hat{d}^2 - \hat{e}^2)(-\hat{b}^2 + 4\hat{a}\hat{c})}}{2\hat{a}(\hat{a}\hat{d}^2 - \hat{e}^2)}. \tag{27}$$

This again gives two solutions for γ, from which the minimum locations are calculated with $\mathbf{x_c} = \gamma\mathbf{x}_j + (1-\gamma)\mathbf{x}_k$.

Finding the minimum. Each of the minimization problems posed in the previous two paragraphs returns a number of potential minimizers $\mathbf{x_c}$ for Eq. (18). Since the minimum might not be inside the triangle, and not on the edges, the three corners of Q are also appended to the list of potential minimizers. Finally, the true minimum can be found by evaluating T_i with Eq. (17) for each potential minimizer, and accepting the one with the lowest value.

Note that it is possible to exclude potential minimizers by looking at the second derivative, however because the evaluation of the minimum is extremely cheap, this did not result in a speed up.

3.2. Parallel front propagation

The local update described in the previous section can propagate the front from three nodes with a known arrival time to a fourth node. Another aspect of a front propagation algorithm is the order in which the nodes are updated. This is independent of the spatial dimension of the design domain, and in principle one could use the Ordered Upwind Method (OUM) [29], similar to the 2D overhang filter as presented in [15]. From this work it followed that in 2D, the computational time of the overhang filter scaled roughly linear with number of DOFs, and was about one to two orders of magnitude smaller, compared to the computational time of the finite element analysis (FEA) for a stiffness optimization. In 3D, the computation times continue to scale favourably for the front propagation. However, the FEA associated with the objective evaluation is commonly parallelized, reducing its computation time proportional to the number of processors available. It is therefore paramount to parallelize the front propagation as well, such that the overhang filter does not become the main computational burden of the topology optimization.

3.2.1. Sequential ordered upwind method

For the sake of completeness, we briefly discuss the sequential OUM detailed in [29]. The OUM is an extension of the Fast Marching Method (FMM) [31,32] that can also handle anisotropic speed functions. In the OUM, nodes are labelled either *Far*, *Considered* or *Accepted*. The *Accepted* nodes have their final arrival times, i.e. are not subject to further change. The *Considered* nodes have been updated at least once from the *Accepted* nodes, but their arrival time values might still change. Finally, the *Far* nodes have not yet been updated and their arrival time are initialized at ∞. Furthermore, *Accepted* nodes can have the additional label *AcceptedFront* when they are adjacent to at least one *Considered* or *Far* node. The *AcceptedFront* is the current frontier of the propagation. When a node is updated, its arrival time is calculated from the nodes on the *AcceptedFront* within a radius $\bar{h}(F_2/F_1)$, where $\bar{h}$ is the maximum element edge length, and F_1 and F_2 the minimum and maximum of the direction dependent part of speed function given in Eq. (5). Thus, F_2/F_1 represents the anisotropy ratio of the speed function, and for $\alpha_{\text{oh}} = 45°$ this is $F_2/F_1 = \sqrt{2}$.

The domain is initialized with all nodes labelled *Far*, except for the boundary nodes on $\partial\Omega_0$, which are labelled *Accepted*. Next, the *Far* nodes adjacent to the boundary are updated from the *AcceptedFront* nodes, using the local update described in the previous section, and become *Considered*. The algorithm then proceeds as described in Algorithm 1. When the algorithm terminates, all the nodes are in *Accepted* with the correct arrival times.

Algorithm 1 Front propagation

1: Move the *Considered* node with the lowest arrival time to *Accepted*.
2: Compute arrival times for the *Far* and *Considered* nodes within the radius $\bar{h}(F_2/F_1)$ from the latest *Accepted* node, with adjacent nodes in the *AcceptedFront*, using the local update (Section 3.1). If the computed arrival time is smaller than the current value, update the arrival time. Label the nodes as *Considered* if they were previously labelled as *Far*.
3: If *Considered* is not empty, move to Step 1.

3.2.2. Parallel ordered upwind method

Several parallel implementations of the FMM have been presented (e.g. [33–37]). However, to the best of our knowledge, no parallel implementation of the OUM has been published. We therefore present the parallel implementation of the OUM by adapting the algorithm presented in [33] to accommodate the anisotropic front propagation.

The parallelization of the FMM presented in [33] is based on a domain decomposition. At the basis of the parallel implementation of the OUM lies the sequential front propagation described in Algorithm 1, but within a subset of the entire domain, i.e. the narrow band. That implies that Algorithm 1 is terminated either if *Considered* is empty, or if the node in *Considered* with the lowest arrival time has a value higher than a given band limit. As such, the front is only propagated a limited distance, instead of through the entire domain. Furthermore, for anisotropic speed functions, nodes are not necessarily updated from adjacent nodes. Therefore, there must be an overlapping region between adjacent domains, termed the ghost region, at least as wide as the update radius $\bar{h}(F_2/F_1)$. Finally, the front propagation must be redone for all the *Accepted* nodes that have an arrival time higher than the updated ghost nodes, because these *Accepted* nodes can possibly get a lower arrival time from the ghost nodes updated from adjacent domains. An outline of the parallel OUM algorithm that is executed in every parallel domain, is listed in Algorithm 2. The algorithm mainly consists of a loop which performs a front propagation, updates the domain boundary with parallel domains, rolls back the front propagation, and performs a corrective front propagation with the updated ghost nodes.

In the domain decomposition strategy as proposed by [33], each processor has a fixed part of the domain. This has the benefit that only boundary information needs to be communicated to other processors. However, it also results in a large amount of idle time, as the front will only propagate briefly through each decomposed domain. We therefore implemented a shared memory approach per machine, where every processor can work on every domain. By having many more domains than processors, there is little idle time as each processor can pick a domain in which the front needs to be propagated. The downside is that between each machine, not only the boundary nodes, but all the information of every domain that has been updated needs to be shared, in order to allow every processor to work on every domain. This large amount of data communication could be avoided by assigning a fixed domain to every machine, and then further dividing these domains locally. This is however left for future work.

Algorithm 2 Parallel front propagation

1: Perform a front propagation up to the band limit.
2: For each ghost node determine the domain containing the minimum value, and scatter that value to the other domains.
3: Exit if there are no updated nodes and *Considered* is empty.
4: Move *Accepted* nodes that have an arrival time higher than or equal to the updated ghost node with the lowest arrival time to *Considered*, and add these nodes to the heap.
5: Perform a front propagation up to the band limit.
6: Increase the band limit, move to Step 1.

3.2.3. Sensitivities

The sensitivities of the front propagation problem can be calculated efficiently in an adjoint sense, as shown in [15]. The adjoint is calculated backward, in exactly the opposite order as the front propagation. However, when

the front propagation is performed in parallel, the order of operations is no longer clearly defined, as several nodes are accepted simultaneously in different domains. Fortunately, for every node it is known from which nodes its arrival time has been calculated. Assume that for every node i, the array $dependence(i)$ contains the indices of the nodes from which the arrival time of node i is calculated. In order to calculate the adjoint in the correct order we therefore propose the strategy outlined in Algorithm 3.

The sensitivity calculation can also be parallelized, where essentially this algorithm is executed in every domain, until the queue is empty, then the dependency and adjoint values of the ghost nodes are exchanged, and Algorithm 3 is restarted. This is repeated until all the sensitivities have been calculated.

Algorithm 3 Parallel front propagation sensitivities

```
Set up a vector dep_count, such that dep_count(i) contains the number of nodes whose arrival time is calculated from node i
Add the nodes i with dep_count(i) = 0 to queue L
while L is not empty do
    i = POP(L)
    Calculate adjoint and sensitivities for node i
    for all j ∈ dependence(i) do
        Propagate the adjoint to node j
        dep_count(j)−−
        if dep_count(j) == 0 then
            Add j to queue L
        end if
    end for
end while
```

4. Numerical examples

The proposed front-propagation-based overhang filter is demonstrated on three cases, presented in the following, using the optimization scheme presented in Section 2.3. In all three cases compliance is minimized, and the optimization problem reads

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \mathbf{f}^T\mathbf{u} \\ \text{s.t.} \quad & \mathbf{K}(\boldsymbol{\xi})\mathbf{u} = \mathbf{f}, \\ & \int_\Omega \xi \,\mathrm{d}\Omega / V_{\lim} - 1 \leq 0, \\ & \boldsymbol{\rho}_{\min} \leq \boldsymbol{\rho} \leq \mathbf{1}, \end{aligned} \tag{28}$$

where $\mathbf{f}$ and $\mathbf{u}$ represent the load and displacement vectors, respectively. $\mathbf{K}(\boldsymbol{\xi})$ is the stiffness matrix dependent on the printable densities ξ. $V_{\lim}$ is the maximum allowed volume fraction and $\rho_{\min} = 0.01$. For more details on the compliance optimization problem and its sensitivities, the reader is referred to [26].

In all cases, a density filter is used with a radius of roughly $2\bar{h}$, where $\bar{h}$ is the average edge length of the elements. The domains are discretized with linear tetrahedral elements using Gmsh [38]. Furthermore, the material properties are chosen as $E = 1.0\,\mathrm{N/m^2}$ and $\nu = 0.3$, and the SIMP-model is used to interpolate the Young's modulus with $p = 3$ [39]. The topology optimization code utilizes the Portable, Extensible Toolkit for Scientific Computing (PETSc), which is used to parallelize the density filter and the FEA, and to handle the unstructured mesh [40–42]. The optimization is performed with the Method of Moving Asymptotes (MMA) [43], with default asymptote increase and decrease parameters of 1.2 and 0.7, respectively. For this, the PETSc based MMA class of the code presented in [44] has been used [45]. The topologies are visualized by displaying the printable density field for $\xi \geq 0.5$. This gives an isosurface of $\xi = 0.5$ inside the domain, and $\xi \geq 0.5$ at the bounds of the domain, as can be seen in Fig. 8. The topologies are displayed without any smoothing, and are obtained using ParaView [46].

Furthermore, similar to the 2D implementation, it is preferred to initially deactivate and then gradually activate the overhang filter, in order to apply the overhang filter to a better initial design. This is achieved with a linear

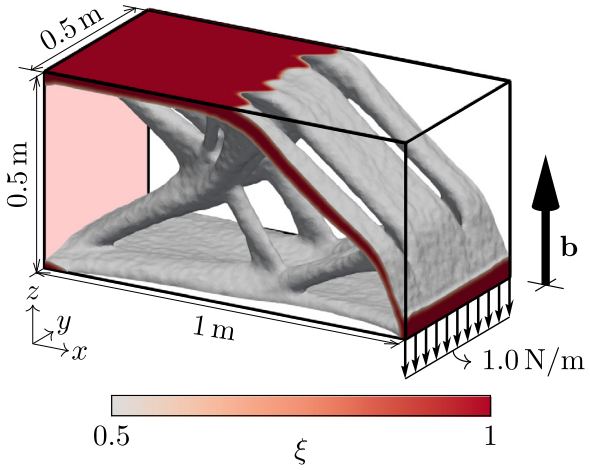


Fig. 8. The cantilever beam design domain and boundary conditions. The light red surface at $x = 0$ is fully clamped, while a distributed load is acting on the front rib, and the build direction is equal to the positive z-axis. The displayed topology is optimized without overhang filter. The density field is displayed for $\xi \geq 0.5$. This gives an isosurface of $\xi = 0.5$ inside the domain, and $\xi \geq 0.5$ at the bounds of the domain.

interpolation between the filtered densities and the printable densities:

$$\hat{\xi} = (1 - c)\rho^* + c\xi, \tag{29}$$

where c is the continuation variable. The objective and constraints are then evaluated on $\hat{\xi}$ instead of ξ. The continuation scheme applied is as follows: $c = 0$ for the first 10 iterations, then c is increased by 0.1 every iteration until $c = 1$ at iteration 20.

Finally, all optimizations are terminated after 300 iterations. It was found that for the larger 3D cases, it can easily take up to 1000 iterations before conventional convergence criteria are met, such as the maximum change in design variables smaller than $1 \cdot 10^{-2}$, or the relative objective change below $1 \cdot 10^{-6}$. However, after 300 iterations the change in design is minute, and the largest improvement in objective observed by allowing the optimization to run for 1000 iterations instead of 300 was around 0.5%. Therefore, in order to save computational time, and also to mimic a practical engineering application where computational time will be the limiting factor, the number of iterations has been limited to 300, instead of waiting for convergence.

4.1. Case 1: the cantilever beam

The overhang filter is first demonstrated on the well-known cantilever beam problem, for several different overhang angles. The design domain is a block with aspect ratio 2:1:1, which is fully clamped on one side and a distributed load is acting on a rib on the opposite side, as displayed in Fig. 8. The resulting design is to be printed in the positive z-direction, as indicated by the build direction **b**, and the base plate coincides with the bottom of the domain ($z = 0$). The filter radius is chosen to be 15 mm, the volume constraint is set at 20%, and the domain is discretized with an unstructured tetrahedral mesh, consisting of $4.3 \cdot 10^6$ elements and $756 \cdot 10^3$ nodes, resulting in an average edge length of 8 mm.

The resulting designs are shown in Fig. 9. It can be seen that the design without overhang filter contains a large overhanging surface at the top. With an increasing allowable minimum overhang angle, the design changes towards a topology with two separated columns, connected with a plate to the bottom. A similar design can be observed in [20]. In Fig. 9c the topology is coloured by the angle between the surface and the base plate, with $\alpha = 0°$ implying a surface parallel to the base plate. The view is from underneath the domain, with the bottom part cut off to allow a view on the downward facing surfaces. It can be seen that for the topologies optimized with overhang filter, there are a lot of green areas, which are surfaces that lie exactly at the minimum overhang angle. Overall the overhang constraint is satisfied, except in some small regions when two surfaces meet, or some individual elements due to the mesh roughness. These small localized overhanging regions are not an issue in AM practice, as they can be printed without support.

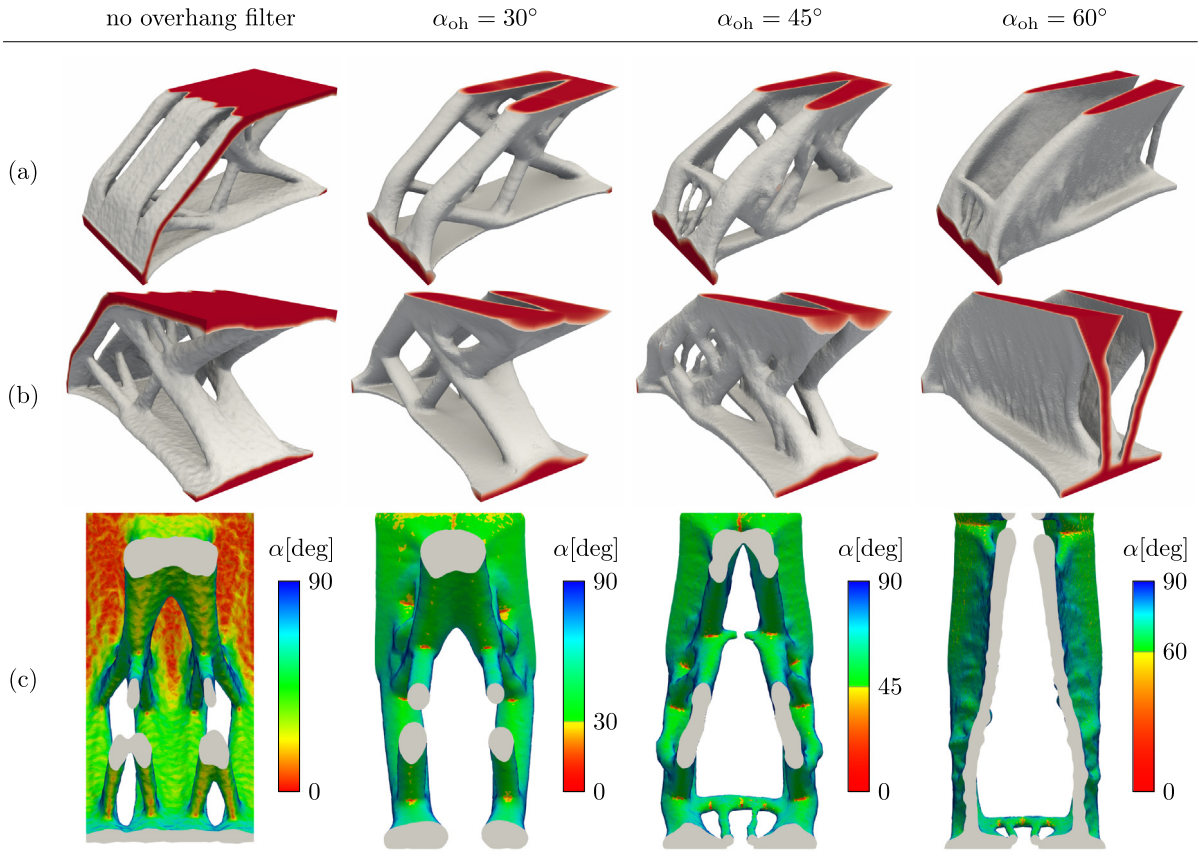


Fig. 9. Resulting topologies for the cantilever beam with a 20% volume constraint, without (left), and with overhang filter with overhang angles of 30°, 45°, and 60°. Rows (a)–(b) show the designs from different angles, while row (c) shows the designs from the bottom, with the bottom plate cut off. For rows (a)–(b), the same colour scheme as in Fig. 8 has been used. In row (c), the colours represent the angle α between the base plate and the isosurface, chosen such that yellow–red colours indicate overhanging surfaces for the given overhang constraint. It can be seen that all designs are practically free of overhang.

4.1.1. Convergence and objective

An interesting aspect of applying an overhang filter is the effect on the convergence behaviour and the final objective. The convergence graphs for the cantilever beam problem are displayed in Fig. 10. For the first 10 iterations, the overhang filter is inactive and thus all optimizations have the same objective values. During iterations 10–20, the overhang filter is activated gradually as descried in Section 2.3, and there is a significant increase in objective for $\alpha_{oh} = 45°$ and $\alpha_{oh} = 60°$. Then, the designs adapt to the overhang filter, and the objective steadily decreases until the optimization is stopped at 300 iterations.

As expected, the optimization without overhang filter has the lowest objective. However, the performance of the overhang filtered optimization is counter-intuitive: the optimization with the lowest minimum overhang angle has the worst performance, while the largest overhang angle performs best. The reason for this behaviour is the fact that the optimizer can use the overhang filter to make smaller features on overhanging surfaces than the filter radius. This effect increases with increasing minimum overhang angle, as larger minimum overhang angles have a lower sideways propagation speed (see Fig. 2c, for larger overhang angles the height of the cylinder remains the same, but the radius decreases). This means that the delay increases faster over distance, which produces sharper features. Therefore, the $\alpha_{oh} = 60°$ setting can make the sharpest edges, and achieves a lower objective. This can be prevented by placing the density filter after the overhang filter, however, this will reintroduce some overhang due to rounding. Therefore, we choose to place the overhang filter after the density filter. Furthermore, this behaviour only emerges in cases that are relatively insensitive to the design, such as this cantilever beam.

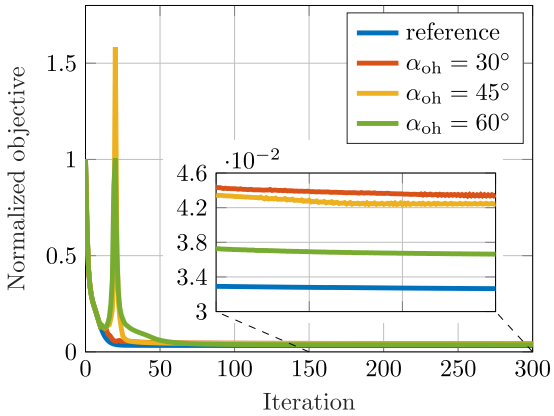


Fig. 10. Convergence graph for the cantilever beam problem, normalized with the first objective evaluation. The reference is optimized without overhang filter.

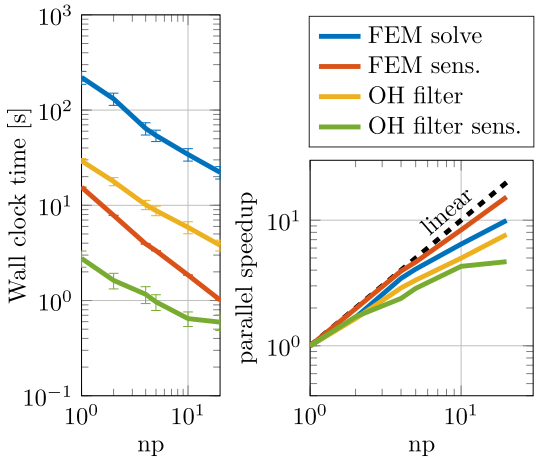


Fig. 11. Wall clock time and speedup, defined as T_{np}/T_1, over number of processors (np) of the overhang filter compared to the FEA, and the sensitivity computations. Error bars indicate the standard deviation.

4.1.2. Parallel speedup

The cantilever beam problem is used to evaluate the parallel performance of the overhang filter. The average wall clock time over the first 10 iterations of the FEA, the overhang filter, and the corresponding sensitivities is plotted in Fig. 11, for various numbers of processors. It can be seen that the overhang filter is roughly a factor 10 faster than the FEA. The speedup is defined as T_{np}/T_1, where T_1 is the wall clock time for the serial process, and T_{np} is the wall clock time for np processors. The speedup of the overhang filter is close to linear for a low number of processors, and drops for larger number of processors, comparable to the FEA. The speedup is tested up to 20 processors, which was the largest number of cores available for a single computational node. Because the available cluster did not feature a fast interconnect, the speedup of the overhang filter did not increase when an extra node was added to the system. Therefore, a hybrid implementation between a complete domain decomposed approach as presented in [33], and a shared memory approach adopted in this work (see Section 3.2.2) would be faster, as it would require only boundary node communication.

4.2. Case 2: tension cylinder

The second case on which the overhang filter is tested is a cylinder under tensional loading. This is a critical test for the overhang filter, as for the lower volume fractions, most of the material that is needed for support is

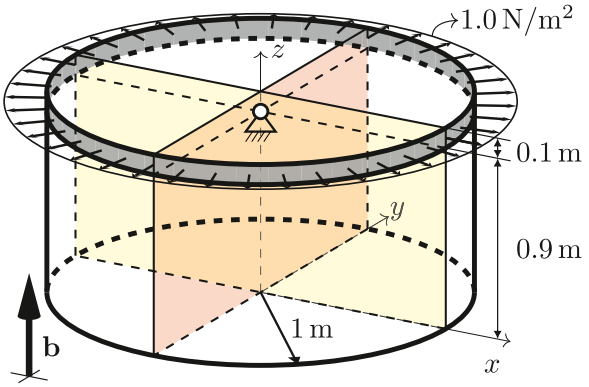


Fig. 12. The tension cylinder design domain with boundary conditions. The tension load acts on the grey ring, while the red surface is clamped in the x-direction, and the yellow surface is clamped in the y-direction. One point on the z-axis is fully clamped, as indicated. The build direction is equal to the positive z-axis, and the build plate is the $z = 0$ plane.

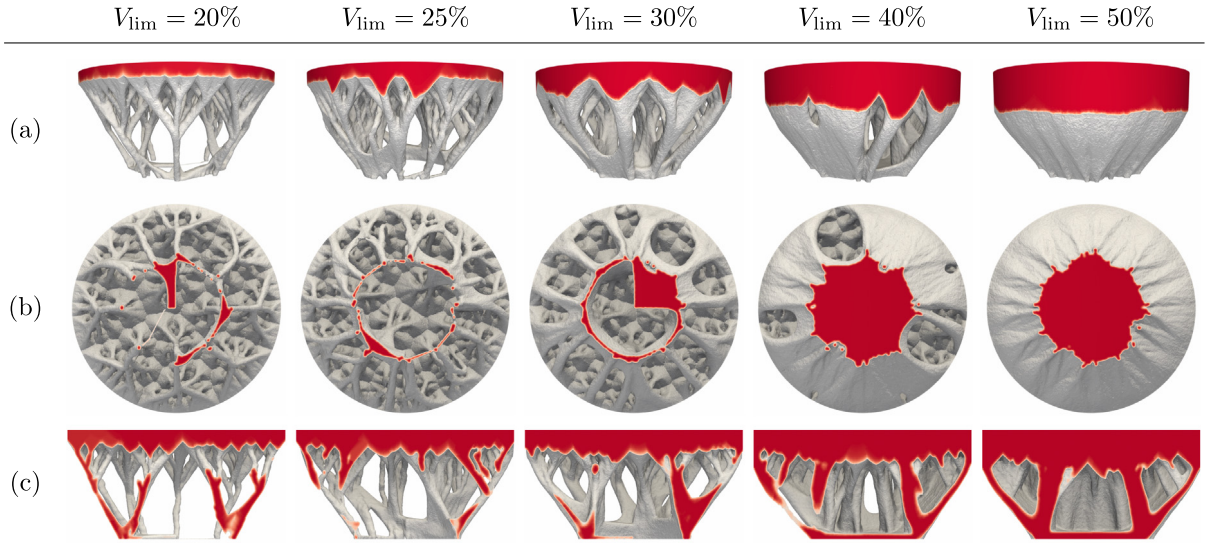


Fig. 13. Resulting designs for the tension cylinder case for different volume fractions viewed (a) from the side, (b) from the bottom, and (c) from the side while cut in half. The same colour scheme as in Fig. 8 has been used. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

not contributing to the structural stiffness. The design domain is a cylinder, with a distributed load acting on the top part of the outer boundary, as displayed in Fig. 12. The filter radius is 20 mm, and the domain is meshed with $17 \cdot 10^6$ elements and $2.9 \cdot 10^6$ nodes, with in an average edge length of 12 mm.

Without overhang filter, the resulting design will roughly be a solid disk starting from the top of the cylinder, with a height equal to the allowed volume fraction. The resulting topologies for different volume fractions with $\alpha_{\text{oh}} = 45°$ are displayed in Fig. 13. It can be seen that even for the lowest volume fraction, fully dense supports are generated, although they do not contribute to the stiffness. The supports fan out in a tree-like fashion to span an as large as possible area with the least amount of material. As the volume fraction increases, the topology resembles an inverted dome. For the higher volume fractions, the supports are also used to increase the stiffness, as they become interconnected.

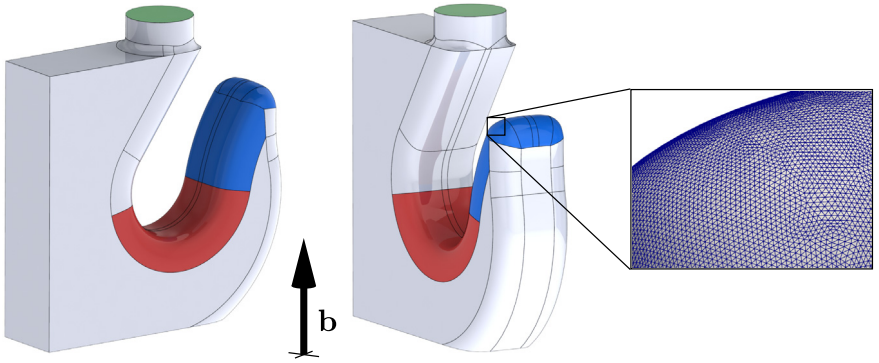


Fig. 14. Design domain and tetrahedral mesh for the crane hook case. The green surface is fully clamped, a vertical distributed load is applied on the red surface, and the red and blue surfaces are constrained at full density. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.3. Case 3: crane hook

The final case on which the overhang constraint is demonstrated is the stiffness optimization of a crane hook. Due to the capability to detect overhang on an unstructured mesh, it is straightforward to optimize a geometry generated in a CAD environment, with overhang constraint. The geometry of the crane hook is displayed in Fig. 14. It is a complex domain containing multiple curved surfaces, which would be difficult to accurately represent with at structured mesh. The design domain measures $455 \times 491 \times 118$ mm (width $\times$ height $\times$ depth). A density filter with a radius of 3 mm is applied, and the elements have an average edge length of 1.5 mm. The volume constraint is set at 20%, and the build direction is in the vertical direction as displayed in Fig. 14. The build plate is at the bottom of the domain.

The domain is fully clamped at the top, with a vertical distributed load applied on the hook, representing a lifting force. Furthermore, this test case features a fixed solid region at the loaded surface and near the tip of the hook (the red and blue surfaces in Fig. 14). If the densities of the blue region would not be fixed, the material will be removed since it does not contribute to the stiffness. However, this region should be retained in the final topology to prevent the load from slipping off the hook. By simply fixing the design variables, the fixed regions will still be removed by the overhang filter when it is overhanging. We therefore added the following constraint:

$$\int_{\Gamma_f} (1 - \xi) \ \mathrm{d}\Gamma_f \leq 0, \tag{30}$$

where Γ_f is the fixed density region. This constraint is only satisfied when the printable densities $\xi = 1$ on the fixed region, and this successfully fixed the printable densities of the fixed region.

The crane hook is the largest test case in the set, with the final mesh containing $50 \cdot 10^6$ elements and $8.8 \cdot 10^6$ nodes. The problem was solved using 160 cores divided over 8 nodes with two 10 core 2.2 Ghz Intel Xeon CPUs (E-2630v4) each. The average wall clock time for the FEA was 193 s, while the average time for the overhang filter was 115 s. The computational time for the sensitivities was negligible compared to the forward solve, 1.9 s for the adjoint sensitivities of the FEA, and 17 s for the overhang filter. Although the computational time required for the overhang filter is considerable at 60% of the FEA, one has to keep in mind that the overhang filter is only running at 1 node, compared to 8 nodes for the FEA.

The optimized design is displayed in Fig. 15. It resembles a common crane hook, with a curved stiffener at the back (the leftmost part in Fig. 15a) to counteract the torque on the hook due to the asymmetric geometry. The stiffener contains some small hollow sections to save material. Also the fixed region at the tip is made as thin as possible, for the most part it is only a shell. The design is completely printable, as can be seen on the bottom in Figs. 15b to 15d, where the members in the middle join under a 45° angle. The decrease in performance compared to an optimization without overhang constraint is 14%.

(a) Side view. (b) Back view. (c) Front view. (d) Bottom view.

Fig. 15. Results of the crane hook case. The same colour scheme as in Fig. 8 has been used.

5. Conclusion

In this work, the front propagation based overhang constraint as presented for 2D in [15] has been successfully extended to 3D. The two main challenges of the extension were the efficient propagation in tetrahedral elements, as opposed to triangles in 2D, and the parallelization of the front propagation in order to keep the computational times in the same order as the FEA, which is usually parallelized for 3D cases.

It is shown that also for 3D front propagation, a minimization problem can be set up for each tetrahedral element that can be solved by probing a limited set of cheaply obtainable locations. This enables the fast propagation through an unstructured mesh, for arbitrary overhang angle, contrary to minimum overhang angle filters that work on structured meshes.

The parallelization of the ordered upwind method was achieved by following the same strategy that has been presented for the closely related fast marching method [33]. However, instead of providing each processor with its own fixed domain, a different approach was taken where every processor can work on any domain. Although this reduces processor idle time and provides speedup for relatively small meshes (on the order of $1 \cdot 10^6$ nodes), the approach presented in [33] is likely to give better performance on increasing mesh sizes. Nevertheless, the parallelized overhang filter is about a factor 10 faster than the parallel FEA, when executed on a single machine.

The overhang filter is demonstrated on three problems to test the performance of the filter. It is shown that the filter handles arbitrary overhang angles, generates solid supports even when there is no benefit for the objective, and works on large unstructured meshes. With the framework for parallel 3D front propagation, further research will focus on different applications of front propagation for topology optimization, for example for accessibility of supports: the complete removal of overhang is often not in the designer's best interest, and we thus would like to activate the overhang constraint only in areas where supports are difficult to remove.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was carried out under project number S12.7.14549a in the framework of the Metals for Additive Manufacturing Partnership Program of the Materials innovation institute M2i (www.m2i.nl) and the Netherlands Organization for Scientific Research (www.nwo.nl).

Appendix. Solving the cone problem

Following [30], on the cone edge, the following equality holds:

$$(\mathbf{b}\cdot\boldsymbol{\chi})^2 - \sin(a_{\text{oh}})^2\|\boldsymbol{\chi}\|^2 = 0, \tag{A.1}$$

with $\boldsymbol{\chi} = \mathbf{x}_i - \mathbf{c} = \zeta_1\tilde{\mathbf{a}} + \zeta_2\tilde{\mathbf{b}} + \tilde{\mathbf{c}}$, and $\tilde{\mathbf{a}} = (\mathbf{x}_j - \mathbf{x}_k)$, $\tilde{\mathbf{b}} = (\mathbf{x}_j - \mathbf{x}_l)$, and $\tilde{\mathbf{c}} = (\mathbf{x}_i - \mathbf{x}_j)$. By defining $\tilde{a} = \mathbf{b}\cdot\tilde{\mathbf{a}}$, $\tilde{b} = \mathbf{b}\cdot\tilde{\mathbf{b}}$, and $\tilde{c} = \mathbf{b}\cdot\tilde{\mathbf{c}}$, Eq. (A.1) can be rewritten as

$$c_1\zeta_1^2 + 2c_2\zeta_1\zeta_2 + c_3\zeta_2^2 + c_4\zeta_1 + c_5\zeta_2 + c_6 = 0 \tag{A.2}$$

with

$$\begin{aligned} c_1 &= \tilde{a}^2 - \sin(a_{\text{oh}})^2\tilde{\mathbf{a}}\cdot\tilde{\mathbf{a}}, & c_4 &= 2\tilde{a}\tilde{c} - 2\sin(a_{\text{oh}})^2\tilde{\mathbf{a}}\cdot\tilde{\mathbf{c}},\\ c_2 &= \tilde{a}\tilde{b} - \sin(a_{\text{oh}})^2\tilde{\mathbf{a}}\cdot\tilde{\mathbf{b}}, & c_5 &= 2\tilde{b}\tilde{c} - 2\sin(a_{\text{oh}})^2\tilde{\mathbf{b}}\cdot\tilde{\mathbf{c}},\\ c_3 &= \tilde{b}^2 - \sin(a_{\text{oh}})^2\tilde{\mathbf{b}}\cdot\tilde{\mathbf{b}}, & c_6 &= \tilde{c}^2 - \sin(a_{\text{oh}})^2\tilde{\mathbf{c}}\cdot\tilde{\mathbf{c}}. \end{aligned}$$

Eq. (A.2) can be rewritten as

$$\mathbf{y}^T\mathbf{K}\mathbf{y} + \mathbf{l}^T\mathbf{y} + c_6 = 0, \tag{A.3}$$

with $\mathbf{K} = \begin{bmatrix} c_1 & c_2 \\ c_2 & c_3 \end{bmatrix}$, $\mathbf{l} = \begin{bmatrix} c_4 & c_5 \end{bmatrix}^T$, $\mathbf{y} = \begin{bmatrix} \zeta_1 & \zeta_2 \end{bmatrix}^T$.

Since on the cone edge, $\tan(\alpha_{oh})\,\|\mathbf{P}\boldsymbol{\chi}\| = |\mathbf{b}\cdot\boldsymbol{\chi}|$, the minimization problem posed in Eq. (19) simplifies on the cone edge to

$$\begin{aligned} \min_{\mathbf{y}} \quad & \left\{ \frac{|\mathbf{b}\cdot\boldsymbol{\chi}|}{f_0\phi_i} + (1-\zeta_1-\zeta_2)T_j + \zeta_1 T_k + \zeta_2 T_l \right\},\\ \text{s.t.} \quad & \mathbf{y}^T\mathbf{K}\mathbf{y} + \mathbf{l}^T\mathbf{y} + c_6 = 0, \end{aligned} \tag{A.4}$$

which can be rewritten as

$$\begin{aligned} \min_{\mathbf{y}} \quad & \mathbf{g}^T\mathbf{y},\\ \text{s.t.} \quad & \mathbf{y}^T\mathbf{K}\mathbf{y} + \mathbf{l}^T\mathbf{y} + c_6 = 0, \end{aligned} \tag{A.5}$$

with $\mathbf{g} = (\pm\tilde{\mathbf{d}}/(f_0\phi_i) + \tilde{\mathbf{T}})$, $\tilde{\mathbf{d}} = \begin{bmatrix} \tilde{a} & \tilde{b} \end{bmatrix}^T$ and $\tilde{\mathbf{T}} = \begin{bmatrix} T_k - T_j & T_l - T_j \end{bmatrix}^T$. Note that in Eq. (A.5), the constant terms have been dropped as they do not influence the minimization, and the absolute function is omitted by probing both the positive and the negative variant of $\tilde{\mathbf{d}}$. The constraint in Eq. (A.5) can be added to the objective with a Lagrange multiplier:

$$\mathcal{L}(\mathbf{y},\lambda) = \mathbf{g}^T\mathbf{y} + \lambda\left(\mathbf{y}^T\mathbf{K}\mathbf{y} + \mathbf{l}^T\mathbf{y} + c_6\right), \tag{A.6}$$

whose stationary points can be found by solving

$$\frac{\partial\mathcal{L}}{\partial\mathbf{y}} = \mathbf{g} + \lambda\,(2\mathbf{K}\mathbf{y} + \mathbf{l}) = 0, \tag{A.7}$$

$$\frac{\partial\mathcal{L}}{\partial\lambda} = \mathbf{y}^T\mathbf{K}\mathbf{y} + \mathbf{l}^T\mathbf{y} + c_6 = 0. \tag{A.8}$$

Now, λ can be found by solving Eq. (A.7) for $\mathbf{y}$ and substituting the result into Eq. (A.8). This gives

$$\lambda = \pm\sqrt{\frac{\mathbf{g}^T\mathbf{K}^{-1}\mathbf{g}}{\mathbf{l}^T\mathbf{K}^{-1}\mathbf{l} - 4c_6}}. \tag{A.9}$$

Substituting this result back into Eq. (A.7) gives

$$\mathbf{y} = -\frac{1}{2}\mathbf{K}^{-1}\left(\frac{\mathbf{g}}{\lambda} + \mathbf{l}\right). \tag{A.10}$$

Note that in total four potential minimum locations are found. Two resulting from the omission of the absolute sign by probing both possibilities, and two from the square root when calculating λ.

References

[1] O. Sigmund, K. Maute, Topology optimization approaches: A comparative review, Struct. Multidiscip. Optim. 48 (6) (2013) 1031–1055.
[2] J.D. Deaton, R.V. Grandhi, A survey of structural and multidisciplinary continuum topology optimization: Post 2000, Struct. Multidiscip. Optim. 49 (1) (2014) 1–38.
[3] J.-P. Kruth, B. Vandenbroucke, J. Van Vaerenbergh, P. Mercelis, Benchmarking of different SLS/SLM processes as Rapid Manufacturing techniques, IEEE Electron Device Lett. - IEEE Electron Device Lett. 10 (2005).
[4] D. Thomas, The Development of Design Rules for Selective Laser Melting (Ph.D. thesis), University of Wales, 2009.
[5] G.A. Adam, D. Zimmer, Design for additive manufacturing-element transitions and aggregated structures, CIRP J. Manuf. Sci. Technol. 7 (1) (2014) 20–28.
[6] J. Kranz, D. Herzog, C. Emmelmann, Design guidelines for laser additive manufacturing of lightweight structures in tial6v4, J. Laser Appl. 27 (S1) (2015) S14001.
[7] M.K. Thompson, G. Moroni, T. Vaneker, G. Fadel, R.I. Campbell, I. Gibson, A. Bernard, J. Schulz, P. Graf, B. Ahuja, F. Martina, Design for additive manufacturing: Trends, opportunities, considerations, and constraints, CIRP Ann. 65 (2) (2016) 737–760.
[8] G. Boothroyd, P. Dewhurst, W.A. Knight, Product Design for Manufacture and Assembly, third ed., in: Manufacturing Engineering and Materials Processing, CRC Press, Boca Raton, Fl, 2011.
[9] B. Vandenbroucke, J.-P. Kruth, Selective laser melting of biocompatible metals for rapid manufacturing of medical parts, Rapid Prototyp. J. 13 (4) (2007) 196–203.
[10] D. Wang, Y. Yang, Z. Yi, X. Su, Research on the fabricating quality optimization of the overhanging surface in SLM process, Int. J. Adv. Manuf. Technol. 65 (9–12) (2013) 1471–1484.
[11] D. Brackett, I. Ashcroft, R. Hague, Topology optimization for additive manufacturing, in: Proceedings of the Solid Freeform Fabrication Symposium, Austin, TX, 2011, pp. 348–362.
[12] A.T. Gaynor, J.K. Guest, Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design, Struct. Multidiscip. Optim. 54 (5) (2016) 1157–1172.
[13] M. Langelaar, An additive manufacturing filter for topology optimization of print-ready designs, Struct. Multidiscip. Optim. (2016) 1–13.
[14] X. Guo, J. Zhou, W. Zhang, Z. Du, C. Liu, Y. Liu, Self-supporting structure design in additive manufacturing through explicit topology optimization, Comput. Methods Appl. Mech. Engrg. 323 (Supplement C) (2017) 27–63.
[15] E. van de Ven, R. Maas, C. Ayas, M. Langelaar, F. van Keulen, Continuous front propagation-based overhang control for topology optimization with additive manufacturing, Struct. Multidiscip. Optim. (2018).
[16] J. Liu, A.T. Gaynor, S. Chen, Z. Kang, K. Suresh, A. Takezawa, L. Li, J. Kato, J. Tang, C.C.L. Wang, L. Cheng, X. Liang, A.C. To, Current and future trends in topology optimization for additive manufacturing, Struct. Multidiscip. Optim. 57 (6) (2018) 2457–2483.
[17] X. Qian, Undercut and overhang angle control in topology optimization: A density gradient based integral approach, Internat. J. Numer. Methods Engrg. (2017) nme.5461.
[18] A.M. Mirzendehdel, K. Suresh, Support structure constrained topology optimization for additive manufacturing, Comput. Aided Des. 81 (2016) 1–13.
[19] G. Allaire, C. Dapogny, R. Estevez, A. Faure, G. Michailidis, Structural optimization under overhang constraints imposed by additive manufacturing technologies, J. Comput. Phys. 351 (Supplement C) (2017) 295–328.
[20] M. Langelaar, Topology optimization of 3D self-supporting structures for additive manufacturing, Addit. Manuf. 12 (2016) 60–70.
[21] M. Hoffarth, N. Gerzen, C. Pedersen, Alm overhang constraint in topology optimization for industrial applications, in: Proceedings of the 12th World Congress on Structural and Multidisciplinary Optimisation, Braunschweig, Germany, 2017.
[22] W. Wang, Y.J. Liu, J. Wu, S. Tian, C.C.L. Wang, L. Liu, X. Liu, Support-free hollowing, IEEE Trans. Vis. Comput. Graphics PP (99) (2017) 1.
[23] T.E. Johnson, A.T. Gaynor, Three-dimensional projection-based topology optimization for prescribed-angle self-supporting additively manufactured structures, Addit. Manuf. (2018).
[24] O. Amir, Y. Mass, Topology optimization for staged construction, Struct. Multidiscip. Optim. (2017).
[25] M. Langelaar, Combined optimization of part topology, support structure layout and build orientation for additive manufacturing, Struct. Multidiscip. Optim. 57 (5) (2018) 1985–2004.
[26] M. Bendsoe, O. Sigmund, Topology Optimization: Theory, Methods and Applications, in: Engineering online library, Springer, 2003.
[27] O. Sigmund, J. Petersson, Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima, Struct. Optim. 16 (1) (1998) 68–75.
[28] T.E. Bruns, D.A. Tortorelli, Topology optimization of non-linear elastic structures and compliant mechanisms, Comput. Methods Appl. Mech. Engrg. 190 (26) (2001) 3443–3459.
[29] J.A. Sethian, A. Vladimirsky, Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms, SIAM J. Numer. Anal. 41 (1) (2003) 325–363.
[30] P.J. Schneider, D.H. Eberly, Geometric Tools for Computer Graphics, in: The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Boston : Morgan Kaufmann Publishers, Amsterdam, 2003.
[31] R. Kimmel, J.A. Sethian, Computing geodesic paths on manifolds, Proc. Natl. Acad. Sci. 95 (15) (1998) 8431–8435.
[32] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. 93 (4) (1996) 1591–1595.
[33] J. Yang, F. Stern, A highly scalable massively parallel fast marching method for the eikonal equation, J. Comput. Phys. 332 (2017) 333–362.
[34] M. Breuß, E. Cristiani, P. Gwosdek, O. Vogel, An adaptive domain-decomposition technique for parallelization of the fast marching method, Appl. Math. Comput. 218 (1) (2011) 32–44.

[35] M.C. Tugurlan, Fast Marching Methods-Parallel Implementation and Analysis (Ph.D. thesis), Louisiana State University, Baton Rouge, LA, 2008.
[36] O. Weber, Y.S. Devir, A.M. Bronstein, M.M. Bronstein, R. Kimmel, Parallel algorithms for approximation of distance maps on parametric surfaces, ACM Trans. Graph. 27 (4) (2008) 104.
[37] M. Herrmann, A domain decomposition parallelization of the fast marching method, in: Annual Research Briefs, Center for Turbulence Research, Stanford University, Stanford, CA, 2003, pp. 213–225.
[38] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities, Internat. J. Numer. Methods Engrg. 79 (11) (2009) 1309–1331.
[39] M.P. Bendsøe, Optimal shape design as a material distribution problem, Struct. Optim. 1 (4) (1989) 193–202.
[40] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.8, Argonne National Laboratory, 2017.
[41] S. Balay, W.D. Gropp, L.C. McInnes, B.F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.
[42] G. Karypis, V. Kumar, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering, J. Parallel Distrib. Comput. 48 (1998) 71–85.
[43] K. Svanberg, The method of moving asymptotes – a new method for structural optimization, Internat. J. Numer. Methods Engrg. 24 (2) (1987) 359–373.
[44] N. Aage, E. Andreassen, B.S. Lazarov, Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework, Struct. Multidiscip. Optim. 51 (3) (2015) 565–572.
[45] N. Aage, B.S. Lazarov, Parallel framework for topology optimization using the method of moving asymptotes, Struct. Multidiscip. Optim. 47 (4) (2013) 493–505.
[46] J. Ahrens, B. Geveci, C. Law, Paraview: An end-user tool for large data visualization, Vis. Handb. 717 (2005).