Efficient filtering in topology optimization via B-splines*

Mingming Wang
Mechanical, Materials and Aerospace
Engineering Department
Illinois Institute of Technology, Chicago, IL 60616
Email: mwang11@hawk.iit.edu

Xiaoping Qian†
Department of Mechanical Engineering
University of Wisconsin, Madison WI 53706
Email: qian@engr.wisc.edu

Abstract

This paper presents a B-spline based approach for topology optimization of three-dimensional (3D) problems where the density representation is based on B-splines. Compared with the usual density filter in topology optimization, the new B-spline based density representation approach is advantageous in both memory usage and CPU time. This is achieved through the use of tensor-product form of B-splines. As such, the storage of the filtered density variables is linear with respect to the effective filter size instead of the cubic order as in the usual density filter. Numerical examples of 3D topology optimization of minimal compliance and heat conduction problems are demonstrated. We further reveal that our B-spline based density representation resolves the bottleneck challenge in multiple density per element optimization scheme where the storage of filtering weights had been prohibitively expensive.

Introduction

Topology optimization is a computational technique for optimally determining the shape and connectivity of material distribution under physical constraints [1]. Recently a B-spline based topology optimization approach has been proposed in [2] where the density representation is based on B-splines and is independent from the finite element elements. The design variables are B-spline coefficients. The feature size in the optimized structures can be controlled via B-spline degrees and the number of knot intervals. The goal of this paper is to elucidate the computational advantages of such B-spline based density representation in topology optimization of three-dimensional (3D) problems.

A widely used method for topology optimization is based on a solid isotropic material with penalization (SIMP) scheme that converts the 0-1 discrete design problem into a continuous optimization problem [3, 4]. To avoid numerical artifacts such as checkerboard and mesh-dependency [5], various regularization techniques have been introduced. The most common ones are based on

*An earlier version of this paper appeared in 2014 ASME International Design Engineering Technical Conferences.
†Corresponding author for this paper.

the sensitivity filter and density filter [6, 7], with alternatives including adding restrictions such as perimeter constraint [8], gradient constraint and slope constraint [9]. In a typical finite element based topology optimization, the density variables are represented based on elements. Both the density filter and the sensitivity filter involve the weight factor H_{ei} that depends on the center-to-center distance $\mathbb{D}(e,i)$ between the element e and element i. For example, the density filter is defined [10] as follows

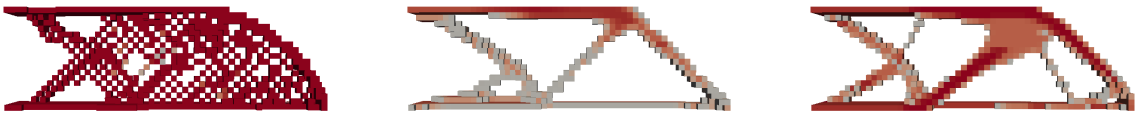
$$\tilde{\rho}_e = \frac{1}{\sum_{i \in \mathbb{N}_e} H_{ei}} \sum_{i \in \mathbb{N}_e} H_{ei} \rho_i \tag{1}$$

where the weight factor $H_{ei} = \max(0, R - \mathbb{D}(e,i))$ and ρ_i is referred to as design variables and the filtered density $\tilde{\rho}_e$ as physical density in element e. $\mathbb{N}_e$ is the set of elements i for which the center-to-center distance $\mathbb{D}(e,i)$ to element e is smaller than the filter radius R. The weight factor H_{ei} only depends on the geometry of the elements, not the densities. In order to avoid computing it in each optimization iteration, such weight factors for each element e, H_{ei} and $i \in \mathbb{N}_e$, are usually pre-generated and stored before each optimization iteration. The number of entries in $\mathbb{N}_e$ is quadratic with respect to R/h where h is the element size. It is cubic with respect to R/h for 3D problems. To solve the storage problem in filters, a Helmholtz partial differential equation based filter in [11] has been proposed. It is estimated in [11] that for $100,000$ elements, a very moderate size for 3D problems, when the filter size R ranges from 2 to 10 times the element size h, the memory required to store the weight factor is 12M, 102M, 818MB, and 1.6GB. The problem is even more exacerbated in alternative topology optimization schemes where more density variables are used in computing each element's physical density. For example, in multi-density topology optimization (MTOP) scheme proposed in [12], each element involves m^d number of density variables where $d = 2, 3$ for 2D and 3D problems. Solving 3D problems with MTOP at $m = 5$ and the filter size R is 10 times element size h, the memory requirement would be 200GB. Such large memory requirements are prohibitively expensive for most desktop computers and would require external storage and extraordinary communication bandwidth between CPU and storage devices.

In this paper, we extend our B-spline based topology optimization approach [2] [1] to 3D problems. Figure 1a shows that checkerboard appears in the three-dimensional optimized design with quadratic ($p = 2$) B-spline of $28 \times 56 \times 112$ knot intervals. The checkerboard can be suppressed either with fewer knot intervals ($7 \times 14 \times 28$ knot intervals in Fig. 1b) or higher B-spline degrees ($p = 8$ in Fig. 1c). More specifically, in this paper we attempt to elucidate the computational advantages of B-spline based density representation, in terms of memory usage and CPU time in B-spline filtering in topology optimization, due to the use of tensor-product form of B-splines. B-splines have been used in topology optimization in the past. For example, B-spline based finite element has also been used in topology optimization [13]. However, in our approach, the B-spline based density representation is independent from analysis techniques and can be used with various FE techniques for topology optimization [2]. Material representation in B-splines and geometry representation in implicit form has also been explored in [14]. The use of B-splines as an implicit representation for shape optimization with topological control has been explored in [15]. However, in our approach, we explicitly discuss the role of B-splines as a filter for obtaining optimized designs that are free from checkerboard patterns and with minimal feature size controlled by B-spline degrees and knot intervals and we discuss the comparison with the usual density filter in the SIMP approach [2].

[1] Matlab implementation of the 2D version is available at `http://cdm.me.wisc.edu/code/btop85.htm`.

(a) p=2,D28 × 56 × 112 (b) p=2, D7 × 14 × 28 (c) p=8, D28 × 56 × 112



(d) Cross section of fig (a) (e) Cross section of fig (b) (f) Cross section of fig (c)

Figure 1: Checkerboard appear in Fig (a) can be suppressed either with fewer knot intervals (Fig. b) or high B-spline degrees (Fig (c)). The cross sections of three respective designs are shown in the second row.

In this paper, we show that with density distribution represented with B-splines, the B-spline functions effective serve as a filter and the cost for storing filtering weight factors is linear with respect to the filter size when B-spline degrees are used to control the minimum feature size. Further, when the number of knot intervals are used to control the feature size, the cost of storing the B-spline weight factors is independent from the effective filter size. The CPU time can be reduced to obtain larger size features in optimized design by using coarser knot vectors. We further reveals that this B-spline filtering can be easily extended to resolve the storage challenge in multiple density per element approach [12].

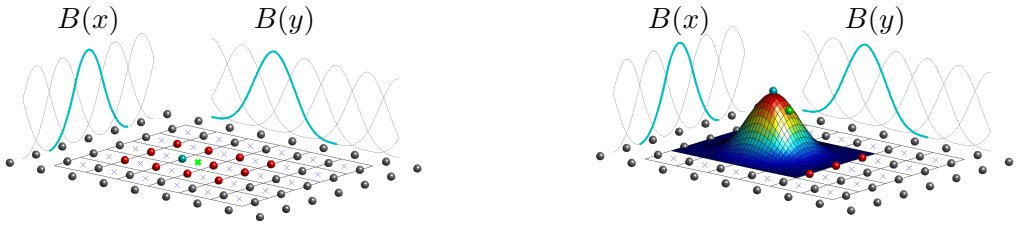
B-splines

In this section, we first briefly review the definition and properties of B-splines and show these properties can be advantageously used in topology optimization. B-splines are commonly used in data approximation and in computer-aided geometric design. For details on B-splines, refer to [16, 17]. For the use of B-splines in density representation in topology optimization, see [2].

A univariate B-spline function of degree p is defined as follows:

$$f(x) = \sum_{i=0}^{n} B_{i,p}(x) b_i, \tag{2}$$

where $b_i(x)$ is the $(i+1)$-th B-spline coefficient and $B_{i,p}$ is the corresponding basis function of degree p defined on a non-decreasing sequence of $m = n+p+1$ real numbers $\Xi = \{\bar{x}_0, \bar{x}_1, \cdots, \bar{x}_m\}$



(a) Univariate basis functions $B(x)$ and $B(y)$ (b) Bivariate basis functions $N(x, y)$

Figure 2: Obtaining bivariate basis function $N(x, y)$ through tensor product $B_p(x)B_q(y)$.

which is called a knot vector. A knot vector is said to be uniform if its knots are uniformly spaced and otherwise non-uniform. The separation between each pair of adjacent knots is referred to as knot interval or knot span δ. In this paper, uniform B-splines are used. Basis functions can be calculated recursively starting with piecewise constants:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } \bar{x}_i \leq \bar{x} < \bar{x}_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and

$$B_{i,p}(\xi) = \frac{\bar{x} - \bar{x}_i}{\bar{x}_{i+p} - \bar{x}_i} B_{i,p-1}(\bar{x}) + \frac{\bar{x}_{i+p+1} - \bar{x}}{\bar{x}_{i+p+1} - \bar{x}_{i+1}} B_{i+1,p-1}(\bar{x}). \tag{4}$$

One very important property of B-splines is local support, i.e. a p-th degree B-spline function is non-zero in at most $(p+1)$ knot intervals. The local support of B-splines has the effect of filtering in topology optimization as shown in [2]. This makes it possible to obtain optimized topological structures that are free from checkerboards, without extraneous filtering or penalty.

By means of tensor product, a bivariate B-spline $f(x, y)$ of degree p in Ξ direction and q in $\mathcal{H}$ direction can be constructed from a bidirectional net of $(n_1 + 1) \times (n_2 + 1)$ B-spline coefficients $b_{i,j}$ as:

$$f(x, y) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} B_{i,p}(x) B_{j,q}(y) b_{i,j}. \tag{5}$$

Corresponding to each coefficient $b_{i,j}$, there is a B-spline basis function $B_{i,p}$ in direction Ξ and basis function $B_{j,q}$ in direction $\mathcal{H}$. They are respectively the pth-degree, qth-degree basis functions defined on non-decreasing knot vectors $\Xi = \{\bar{x}_0, \bar{x}_1, \cdots, \bar{x}_{n_1+p+1}\}$, and $\mathcal{H} = \{\bar{y}_0, \bar{y}_1, \cdots, \bar{y}_{n_2+q+1}\}$. Fig. 2 shows the tensor-product nature of the bivariate B-spline functions: where one B-spline coefficient and the corresponding basis functions are highlighted. Such tensor-product form of B-splines suggests that the evaluation of a bivariate tensor product B-spline function does not necessarily require the storage of the bivariate basis functions. Instead, one can store two sets of univariate basis functions and dynamically compose the bivariate basis function $N(x, y) = B(x)B(y)$ during the function evaluation. This is the basis for our proposed approach to resolve the storage challenges in topology optimization.

Similarly, a tri-variate B-spline function $f(x, y, z)$ of degree p in x direction, degree q in $\mathcal{H}$ direction and degree r in $\mathcal{Z}$ direction with $(n_1 + 1) \times (n_2 + 1) \times (n_3 + 1)$ B-spline coefficients $b_{i,j,k}$ can be defined as:

$$f(x, y, z) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} \sum_{k=0}^{n_3} B_{i,p}(x) B_{j,q}(y) B_{k,r}(z) b_{i,j,k}. \tag{6}$$

Topology optimization in B-spline space

In this section, we show how B-splines can be used to represent density distribution and can be used in topology optimization.

B-spline representation of material density

We use uniform B-splines to represent the material density distribution inside the design domain, as shown in Fig. 3 for a cubic computational domain in 3D where the dots are B-spline coefficients. The grid lines show knot intervals. We can obtain the material density distribution throughout the computational domain through the definition of B-splines. If we denote the B-spline coefficients as $\boldsymbol{\rho}$, and $0 \leq \rho_i \leq 1$, due to the strong convex hull property of B-splines [16], the physical density $\tilde{\rho}$ inside the cube is strictly between 0 and 1, i.e. $0 \leq \tilde{\rho} \leq 1$. By Eq. 6, we can obtain the distribution of physical density $\tilde{\rho}$ as

$$\tilde{\rho}(x, y, z) = \sum_{\hat{i}=0}^{n_1} \sum_{\hat{j}=0}^{n_2} \sum_{\hat{k}=0}^{n_3} B_{\hat{i},p}(x) B_{\hat{j},q}(y) B_{\hat{k},r}(z) \rho_{\hat{i},\hat{j},\hat{k}}. \tag{7}$$

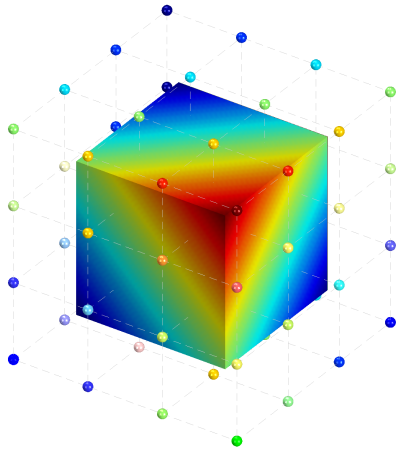
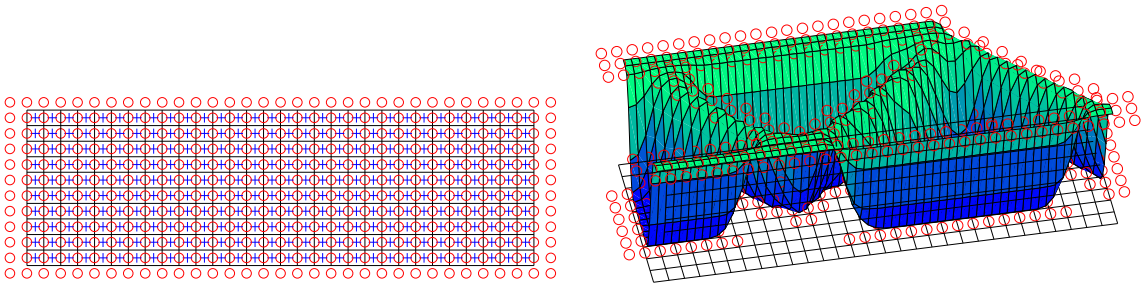


Figure 3: Representing density distribution in 3D by B-splines. The color in B-spline coefficients corresponds to material density values.

Figure 4 gives a 2D illustration of the distribution of B-spline coefficients and finite elements where the optimized distribution of density in a minimal compliance problem. In finite element analysis, the physical density $\tilde{\rho}_e$ in element e can be obtained by sampling the physical density field $\tilde{\rho}(\mathbf{x})$ at the element e's center $\mathbf{x}_e = (x_e, y_e, z_e)$ as

$$\tilde{\rho}_e = \tilde{\rho}(\mathbf{x}_e) = \sum_{\hat{i}=0}^{n_1} \sum_{\hat{j}=0}^{n_2} \sum_{\hat{k}=0}^{n_3} B_{\hat{i},p}(x_e) B_{\hat{j},q}(y_e) B_{\hat{k},r}(z_e) \rho_{\hat{i},\hat{j},\hat{k}}, \tag{8}$$

where $\mathbf{x}_e = (x_e, y_e, z_e)$ representing the center position of element e. For the physical density $\tilde{\rho}_e$ at element e, there are at most $(p+1)$ non-zero entries in $B_{\hat{i},p}(x_e)$, $(q+1)$ non-zero entries in $B_{\hat{j},q}(y_e)$,



(a) Distribution of B-spline coefficients and finite element.

(b) 3D view of B-spline representation of the density distribution $\tilde{\rho}(x, y)$ and B-spline coefficients ρ_i.

Figure 4: Optimized density representation from degree $p = 3, q = 2$ B-splines of 30×10 knot intervals with analysis done by 30×10 quadrilateral linear elements [2]. (a) Distribution of 33×12 B-spline coefficients ρ_i (red circle), ((b) 3D view of density distribution $\tilde{\rho}(\boldsymbol{x})$ and the B-spline coefficients ρ_i.

and $(r+1)$ non-zero entries in $B_{\hat{k},r}(z_e)$, and corresponding $(p+1)(q+1)(r+1)$ entries in design variables $\boldsymbol{\rho}$. Therefore, (8) can be noted as

$$\tilde{\rho}_e = \sum_{i \in \mathbb{N}_e} N_i(\mathbf{x}_e)\rho_i, \tag{9}$$

where the index set $\mathbb{N}_e$ contains $(p+1)(q+1)(r+1)$ entries and the triplet $(\hat{i}, \hat{j}, \hat{k})$ is mapped to the global index i. The weight factor $N_i(\mathbf{x}_e)$ can be evaluated as

$$N_i(\mathbf{x}_e) = B_{\hat{i},p}(x_e)B_{\hat{j},q}(y_e)B_{\hat{k},r}(z_e). \tag{10}$$

The B-spline based density equation (9) has the same form as the density filter (1). This can be seen by viewing $N_i(\mathbf{x}_e)$ as B-spline weight functions, i.e.

$$H_{ei} = N_i(\mathbf{x}_e). \tag{11}$$

Due to the partition of unity property of B-splines, i.e. $\sum_{i \in \mathbb{N}_e} H_{ei} = \sum_{i \in \mathbb{N}_e} N_i(\mathbf{x}_e) = 1$. Therefore, (1) essentially becomes

$$\tilde{\rho}_e = \sum_{i \in \mathbb{N}_e} H_{ei}\rho_i,$$

which has the same form as (9). Thus, we also refer to (9) as a *B-spline filter* that computes physical density $\tilde{\rho}_e$ by weighting design variables (i.e. B-spline coefficients) ρ_i, $i \in \mathbb{N}_e$, with B-spline function $N_i(\mathbf{x}_e)$.

Topology optimization in B-spline space

With the above B-spline representation of density distribution as shown in Fig. 4, an arbitrarily shaped design domain Ω is embeded into a B-spline domain $\bar{\Omega}$. This B-spline function $\tilde{\rho}(x), x \in$

$\bar{\Omega}$, represents the density distribution that are parameterized by a finite number n_c of B-spline coefficients ρ_i, where $i = 1, 2, \cdots, n_c$. As shown in [2], this B-spline representation is independent from how the physical domain is discretized into analysis elements. In this paper, element center based approximation and evenly distributed multiple density points in each finite element are used. For how this approach can be applied to curved design domain and alternative analysis methods such as isogeometric analysis, see [2].

Two different kinds of problems are solved in this paper: a cantilever beam problem to minimize the mean compliance, and a heat conduction problem to minimize the mean temperature. Their nested formulations can be written as:

$$\min_{\boldsymbol{\rho}} : c(\boldsymbol{\rho}) = \mathbf{L}^T\mathbf{U} \tag{12a}$$

$$s.t. : \sum_{e=1}^{n_e} \tilde{\rho}_e \mathrm{v}_e \leq V^* \tag{12b}$$

$$: \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1} \tag{12c}$$

where $\boldsymbol{\rho}$ is the set of design variable(i.e. B-spline coefficients), and c is the compliance. Equation (12b) is a volume constraint where v_e is the volume of element e and V^* is the allowed material volume. Equation (12c) is the box constraint for design variables, ensuring the resulting physical density $\tilde{\rho}$ is within $[0, 1]$.

In the nested formulation, at each iteration, the state variables $\mathbf{U}$ are solved from the equilibrium equation

$$\mathbf{K}\mathbf{U} = \mathbf{F}, \tag{13}$$

where $\mathbf{K}$ is the global stiffness or conductance matrix, $\mathbf{F}$ is the force vector, and $\mathbf{U}$ is the vector of state variables, i.e. displacement for elasticity and temperature for heat conduction problem. The global stiffness or conductance matrix $\mathbf{K}$ is obtained by assembling all the n_e number of element stiffness or conductivity matrix $\mathbf{K}_e$ noted as follows:

$$\mathbf{K} = \sum_{e=1}^{n_e} \mathbf{K}_e(E_e). \tag{14}$$

where $\mathbf{K}_e = E_e\mathbf{K}_e^0$. $\mathbf{K}_e^0$ is the element stiffness matrix for element with solid phases. E_e is the material stiffness or heat conductance obtained by using the solid isotropic material interpolation with penalization (SIMP) given as

$$E_e = E_{\min} + \tilde{\rho}_e^s(E_0 - E_{\min}) \tag{15}$$

where E_0 is the stiffness or conductivity for solid material, s is the penalization power and $\tilde{\rho}_e$ is the physical density in element e.

The sensitivity of the objective function with respect to physical density $\tilde{\rho}_e$ can be calculated from adjoint analysis:

$$\frac{\partial c}{\partial \tilde{\rho}_e} = \lambda^T \frac{\partial \mathbf{K}}{\partial \tilde{\rho}_e}\mathbf{u} \tag{16}$$

where λ is the adjoint variable determined from $\mathbf{K}\lambda = \mathbf{L}$ where $\mathbf{L}$ is the adjoint load. By applying the chain rule, the sensitivity of the objective function with respect to design variable ρ_i can be computed as

$$\frac{\partial c}{\partial \rho_i} = \sum_{e \in \mathbb{N}_i} \frac{\partial c}{\partial \tilde{\rho}_e} \frac{\partial \tilde{\rho}_e}{\partial \rho_i} \tag{17}$$

where $\mathbb{N}_i$ is the set of finite elements $\{e\}$, of which the physical density $\tilde{\rho}_e$ is affected by the design variable ρ_i through a non-zero weight. By substituting equation (16) into equation (17), we can obtain the sensitivity of the objective function with respect to design variables that are required for gradient based optimization. Both sensitivities of the volume constraint and the cost function involves the computing the term $\frac{\partial \tilde{\rho}_e}{\partial \rho_i}$. Based on (9), we have

$$\frac{\partial \tilde{\rho}_e}{\partial \rho_i} = N_i(\mathbf{x}_e). \tag{18}$$

The update of physical density $\tilde{\rho}_e$ in each iteration in (15) also involves the above weight factor $N_i(\mathbf{x}_e)$.

It should be pointed out that the equations shown in this section have the same form as the usual density filter based topology optimization. The difference lies in the meanings of the design variables and the weighting factors that relate the design variables ρ_i to the physical density $\tilde{\rho}_e$. In the usual density filter, the design variable ρ_i represents the density in each element and the weight factor is based on the center-to-center distance $\mathbb{D}(e, i)$ as shown in (1). In B-spline filter, the design variable ρ_i represents the B-spline coefficient. The weight factor is B-spline functions.

Once we know how to compute and evaluate the B-spline function $N_i(\mathbf{x}_e)$, the remainder of the numerical procedures for optimization is then identical to the usual density filter based topology optimization. Therefore, in the next section, we focus on evaluating and storing the B-spline function $N_i(\mathbf{x}_e)$.

Computing and storing B-spline weights

The weight function $N_i(\mathbf{x}_e)$ in the B-spline filter and H_{ei} in the density filter are respectively used in both density evaluation and in computing the sensitivity of the cost function in each optimization iteration. The sensitivity of volume constraint is usually computed once and pre-stored. Here we focus on cost of computing the sensitivity (17) and the cost of storing the filtering weights. The cost of computing the sensitivity (17) can be decomposed into two parts. The first part consists of computing the sensitivity of cost function with respect to physical density a shown in (16) and this part is noted as sensitivity analysis (SA) in time and storage comparison, e.g. in Fig. 5. The second part consists of computing the sensitivity of the physical density with respect to design variables, i.e. $\frac{\partial \tilde{\rho}_e}{\partial \rho_i}$. This term is noted as filter (FIL) in density filter and B-spline evaluation (BE) in B-spline filter.

Computing B-spline weights

We first show algorithmically how to compute the B-spline weights. Algorithm 1 gives the procedure for computing B-spline weight factors $N_i(\mathbf{x}_e)$. The first step is to identify the knot intervals for which the element center $\mathbf{x}_e$ is located based on the following relationships,

$$x_e \in [\bar{x}_{l_e^x}, \bar{x}_{l_e^x+1}], \quad y_e \in [\bar{y}_{l_e^y}, \bar{y}_{l_e^y+1}], \quad z_e \in [\bar{z}_{l_e^z}, \bar{z}_{l_e^z+1}]. \tag{19}$$

The remaining steps include computing univariate B-spline basis functions $B_{\hat{i}}(x_e)$, $B_{\hat{j}}(y_e)$, and $B_{\hat{k}}(z_e)$, and then invoking the tensor product form (10) to obtain the B-spline weight factors.

Although the above procedure for computing the B-spline weight factors is relatively simple, it is still expensive to compute them in each optimization iteration. Since these weights do not

Algorithm 1 Computing weight factors $\mathbf{N}(\mathbf{x}_e)$ for physical density $\tilde{\rho}_e$

Input: Element e's center coordinate $\mathbf{x}_e$

Output: Weight factors $N_i(\mathbf{x}_e)$ of physical density $\tilde{\rho}_e$

1: From $\mathbf{x_e}$, find the knot interval index l_e^x, l_e^y, l_e^z based on (19).
2: Compute $p+1$ basis functions $B_{\hat{i}}(x_e)$ in $[\bar{x}_{l_e^x}, \bar{x}_{l_e^x+1}]$based on (4)
3: Compute $q+1$basis functions $B_{\hat{j}}(y_e)$ in$[\bar{y}_{l_e^y}, \bar{y}_{l_e^y+1}]$ based on (4)
4: Compute $r+1$ basis functions $B_{\hat{k}}(z_e)$ in $[\bar{z}_{l_e^z}, \bar{z}_{l_e^z+1}]$ based on (4)
5: $\forall i \in \mathbb{N}_e$, compute $N_i(\mathbf{x}_e)$ via tensor product (10)

depend on optimization variables ρ_i and they only depend on the location of the element center $\mathbf{x}_e$ with respect to the knot vectors, these weights are thus invariant in each iteration. They can then be pre-computed and stored before optimization. The challenge is that the memory usage for storing weight factors can be prohibitively expensive for three-dimensional problems. We analyze below the cost for storing B-spline weights and compare it with the density filter.

Comparing the cost of storing filtering weights

In the introduction section, we have shown that, with the density filter, the weight factors H_{ei} is cubic with respect to the filter size. In MTOP, the storage cost is also cubic with respect to the number of elements m per direction. In this section, we show the storage cost of the B-spline weight factors $N_i(\mathbf{x}_e)$ can be made linear with respect to the filter size when B-spline degrees are used to control the filter size and the cost can be made even more compact when the number of B-spline knot intervals is used to control the filter size.

Here, an analysis of memory required for storing weights in the density filter and in the B-spline filter is given below. We assume x86-64bits system is used and the size of integer is 4 bytes, and size of double is 8 bytes.

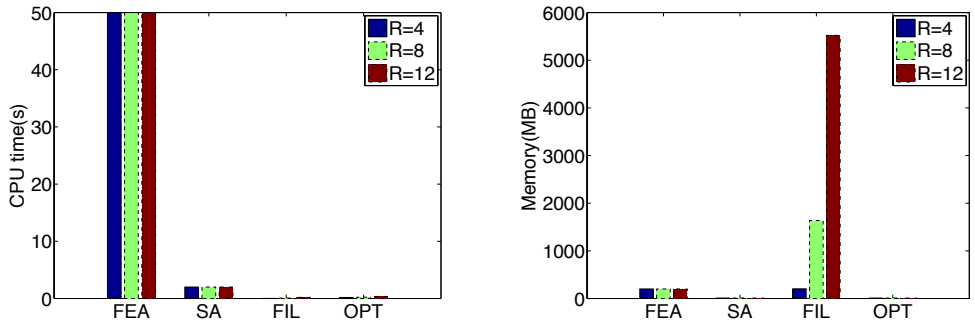
Cost for storing weight factors in the density filter

For density filter, the storage of each weight factor H_{ei} includes a tuple $\{e, i, H_{ei}\}$, where e and i are the indicies and H_{ei} is the weight. They can be stored in the format of COOrdinate list(COO) of (row, column, value) tuples. The weight factor H_{ei} is symmetric with respect to indicies e and i, thus only half needs to be stored. For each physical density $\tilde{\rho}_e$, the number of neighboring elements $n_{\mathbb{N}_e}$ that fall in the filtering region can be estimated from the volume of the sphere of filter radius R as $n_{\mathbb{N}_e} = 4/3\pi\lceil R\rceil^3$, where R is the filter radius. If there are a total number of n_e elements, then the total memory usage of the weight factors is:

$$\begin{aligned} S^d &= (4+4+8)n_e n_{\mathbb{N}_e}/2 = 8 n_e n_{\mathbb{N}_e} \\ &= 32 n_e \pi \lceil R\rceil^3/3\text{bytes}. \end{aligned} \tag{20}$$

We can see that the storage complexity of the density filter is $O(R^3)$.

In Fig. 5, the memory usage and computation time required for density filter based topology optimization with 100,000 linear hexahedra elements for minimum compliance problem is plotted. The tasks compared include finite element analysis (FEA), sensitivity analysis (16), filtering (17) and optimization. The CPU time on a 2.3 GHz computer is also plotted on the left. We can see that finite element analysis takes 90% of the time in each iteration in the optimization. In contrast,



(a) CPU time

(b) Memory usage

Figure 5: The CPU and memory usage of density filter with different filter radius R for four tasks in each optimization iteration.

the density filter takes most of the memory especially at large filter radius. The memory usage for storing density weighting factors exceeds that of the stiffness matrix and the finite element analysis.

Cost for storing evaluated B-spline weight factors

For B-splines, the storage of each evaluated B-spline weight factor $N_i(\mathbf{x}_e)$ includes the tuple $\{e, i, N_i(\mathbf{x}_e)\}$, where e and i are the indicies and $N_i(\mathbf{x}_e)$ is the B-spline weight. However, the B-spline weight $N_i(\mathbf{x}_e)$ is not symmetric with respect to the indicies e and i. Thus the storage cost for all B-spline weights is

$$\begin{aligned} S_e^b &= (4+4+8)n_e(p+1)(q+1)(r+1) \\ &= 16n_e(p+1)(q+1)(r+1)\text{bytes}, \end{aligned} \tag{21}$$

where we have used the fact that the number of entries in $\mathbb{N}_e$ is $(p+1)(q+1)(r+1)$. We can see that with the evaluated B-spline weight factor $N_i(\mathbf{x}_e)$, the storage complexity of the B-spline filter is $O(p^3)$ assuming equal filter size in all axes, i.e. $p = q = r$.

Cost for storing tensor form of the B-spline weight factors

In storing the evaluated B-splines, for each physical density $\tilde{\rho}_e$, one needs to store $n_{\mathbb{N}_e} = (p+1)(q+1)(r+1)$ entries of tuples $\{e, i, N_i(\mathbf{x}_e)\}$. In contrast, when storing each pre-evaluated, tensor form of the weight factor, one needs to only store $p+1$ entries of univariate B-spline basis function $B_{\hat{i}}(x_e)$ along x direction, $q+1$ entries of B-spline function $B_{\hat{j}}(y_e)$ along y direction, and $r+1$ entries of B-spline function $B_{\hat{k}}(z_e)$ along z direction. The evaluated B-spline weight factors $N_i(\mathbf{x}_e)$, $i \in \mathbb{N}_e$, can be obtained from these univariate B-spline functions via (10). Thus, the cost for storing univariate B-spline basis functions in all direction for each $\tilde{\rho}_e$ is $8(p+q+r+3)$bytes.

The $n_{\mathbb{N}_e} = (p+1)(q+1)(r+1)$ entires of indicies $\hat{i}, \hat{j}, \hat{k}$ used in evaluating (10) can be recovered dynamically in each optimization iteration from the indices of the knot intervals, i.e. l_e^x, l_e^y, l_e^z, based on (19). Therefore, the cost for storing the indicies for each $\tilde{\rho}_e$ is 3×4 bytes.

Therefore, the total cost of storing the tensor form of B-spline weights is the sum of the cost for storing univariate B-spline basis functions in all directions and the the storage for the indicies,

as

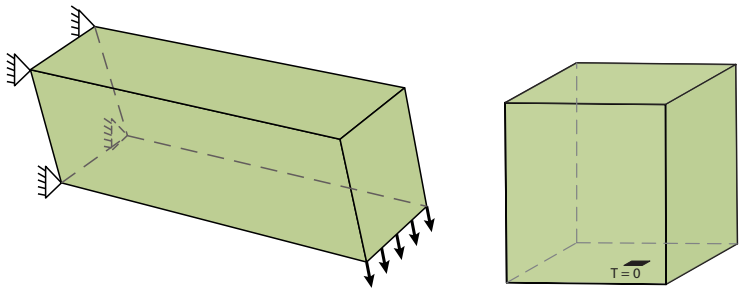
$$\begin{aligned} S_t^b &= n_e\,(8(p+q+r+3)+12) \\ &= 8n_e(p+q+r+4.5)\text{bytes.} \end{aligned} \tag{22}$$

Thus, the total cost for storing the B-spline weights is linear with respect to B-spline degrees p, q and r.

It is important to note that the storage cost for both evaluated form (21) and tensor form (22) of B-spline weights only depends on the number of finite elements n_e and B-spline degrees p, q, r, but not on the number of knots. We can use this property to make B-spline filtering even more efficient when large filter size is desired in topology optimization. That is, we can simply use fewer knots so each knot interval δ becomes larger since the total length of knot intervals must equal the domain size. The domain size of the optimization problem does not change and fewer knots lead to fewer but larger knot intervals δ. Thus, with fewer knots, the effective filter size $\Delta = (p+1)\delta$ in each axial direction becomes larger. We will show such compact storage requirement of B-spline filters in the following examples.

3D numerical examples

We use two 3D optimization problems to illustrate the computational advantages of B-spline based topology optimization. The specifications of the two problems are shown in Fig. 6. In (a), a cantilever beam with evenly distributed loads exerted on the lower edge of the other side. The volume constraint of the solid phase is 15% of the whole domain volume. The Young's modulus of solid and void phases are $E_{\text{solid}} = 1$ and $E_{\text{void}} = 1e-9$. The Poisson's ratio is $\nu = 0.3$. The problem is analyzed with $28 \times 36 \times 112$ eight-node linear hexagonal elements with element size $h = 1$. In Fig. 6(b), a cubic domain is considered. There is uniform heat generation within the design domain. On the center of the lower surface, there is a heat sink with constant temperature. All other boundary surfaces are adiabatic. The volume constraint of the solid phase is 30% of the whole domain volume. The goal is to maximize the heat transfer over the domain. The conductivity of the solid and void phase are $E^H_{\text{solid}} = 1$ and $E^H_{\text{void}} = 1e-3$. The problem is analyzed with $100 \times 100 \times 100$ hexagonal eight-node linear elements.



(a) Minimum compliance

(b) Minimum mean temperature

Figure 6: Problem specification of (a) cantilever beam; (b) heat conduction.

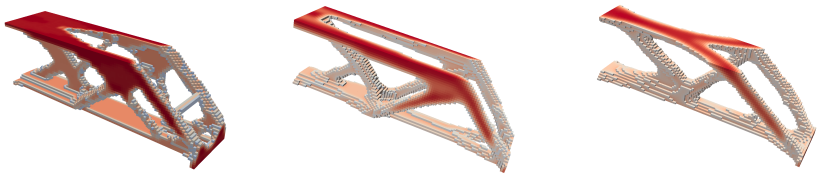
The implementation is based on a combination of C and Python. To solve the finite element equations, Symmetric Successive Over-Relaxation(SSOR) preconditioned conjugate gradient with

error tolerance $1e{-}6$ is used. Sparse matrix solver [18] is used. MMA [19] is used as the optimizer. The optimized results are rendered by Paraview with density threshold 0.5 unless otherwise noted. The optimization convergence criteria is that either the maximum change of design variables less than 0.01 or the maximum iteration number is 800. A 2.3 GHz Intel CPU laptop is used. Depending on the number of elements, it may take several hours to a day to complete the optimization. The density filter results are based on the implementation in [20].

Minimum compliance: cantilever beam

With density represented with B-splines, both degrees p, q, r, and knot intervals $\delta_x, \delta_y, \delta_z$ can be used to control the filter size, which is characterized by $\Delta_x = (p+1)\delta_x$ in x direction with similar expressions for other direction[2]. We demonstrate the storage cost of using both degrees and knot intervals to control the feature size.

Controlling filter size via B-spline degrees



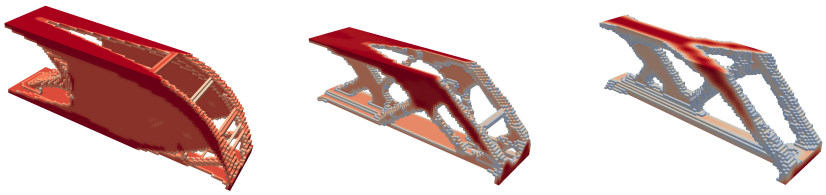
(a) p2, I326, c6470 (b) p6, I583, c5884 (c) p10, I942, c6325

Figure 7: Topology optimization results using different degrees of B-splines at $14 \times 18 \times 56$ knot intervals.

In Fig. 7, the optimized results using different B-spline degrees 2, 6, 10 are shown. The density field is represented by B-splines of $14 \times 18 \times 56$ uniform knot intervals. The number of linear elements used in FE analysis are $28 \times 36 \times 112$. We see that by increasing B-spline degrees, while fixing B-spline knot interval numbers, the minimum feature size becomes larger. Both the tensor-product form and evaluated form of B-spline weights are tested and compared. Both forms of B-spline weights lead to identical optimization results. But the storage efficiency of tensor-product form is much more different, with minor increased cost in computing the B-spline weights from the univariate entires (10). In Table 1, the memory for storing the B-spline weights and CPU time for for computing (17) from physical density's senstivity $\frac{\partial c}{\partial \tilde{\rho}_e}$ are shown for both forms of storing B-spline weights, with degrees ranging from $p = 2$ to $p = 10$. As can be seen from the table, with the increase of B-spline degrees, the cost for storing tensor-product form of B-spline weights increases linearly as predicted in (22) and the cost for storing evaluated B-spline weights increases cubicly as predicted in (21). The time cost of evaluating tensor-product form of B-spline weights increases faster than that of evaluated B-spline weights, although they are of the same complexity, i.e. cubic with respect to B-spline degrees. Thus, B-splines of tensor-product form provides overall efficient filtering. We therefore use tensor-product form of B-spline weights in all subsequent examples.

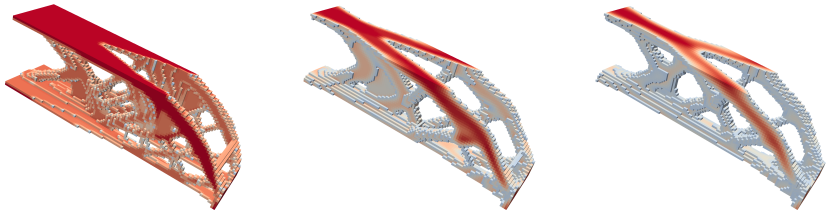
Table 1: CPU Time and memory usage comparison for using different forms of storing B-spline weights for cantilever beam problem.

	Tensor-product form		Evaluated form	
p	S_t^b(Es/Re)	time(s)	S_e^b(Es/Re)	time(s)
2	9.0/9.0MB	0.014	48.8/48.8MB	0.008
6	19.4/19.4MB	0.016	619.6/619.6MB	0.01
10	31.2/31.2MB	0.05	2404/2404MB	0.02
p	$8n_e(3p+4.5)$	$O(p^3)$	$16n_e(p+1)^3$	$O(p^3)$



(a) $\Delta 3$, I460, c5230 (b) $\Delta 6$, I326, c6470 (c) $\Delta 12$, I378, c7238

Figure 8: 3D results optimized with degree 2 B-splines of different B-spline knot intervals.



(a) $R1.5$, I452, c5324 (b) $R3.5$, I367, c6389 (c) $R5.0$, I675, c7357

Figure 9: 3D results optimized with the density filter with analysis mesh $28 \times 36 \times 112$.

Controlling filter size via the number of knot intervals

Figure 8 shows the B-spline based optimized results using the degree $p = q = r = 2$ B-splines of different number of knot intervals. The number of analysis elements is still $28 \times 36 \times 112$ in $x \times y \times z$ directions. In the paper, we use a prefix 'D' before a triple of numbers to represent the number of B-spline knot intervals and the prefix Δ before a number to indicate the effective filter size $(p+1)\delta$. The cost function values and optimization iteration number are shown in the figure with prefix 'c' and 'I'. The B-spline degree is shown with prefix 'p'. From the obtained topology results, we can see that the number of knot intervals controls the feature size in the optimized designs. With fewer but larger intervals $\delta_x = \delta_y = \delta_z$ from Fig. 8(a) to (c), the knot intervals change from $D28 \times 36 \times 112$ to $D14 \times 18 \times 56$ and to $D7 \times 9 \times 28$ and the corresponding effective filter sizes changes from $\Delta 3$ to $\Delta 6$ to $\Delta 12$. The minimum feature size becomes larger and larger. The objective function increases as the number of knot intervals decreases.

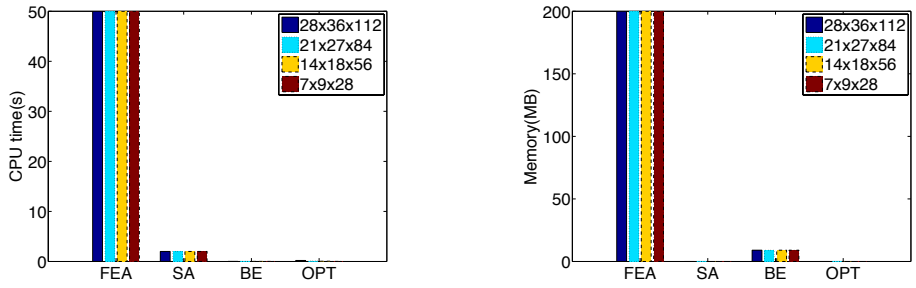
For comparison purpose, we also show the topology optimization results with the density filters from the same number of analysis elements, but of different filter radius R. The results are shown in Fig. 9. As we can see, with the increase of the density filter radius R from 1.5 element size to 5 element size, the compliance becomes larger and the minimal feature becomes larger.

Table 2: Memory usage and time comparison between different filter radius with the density filter and different knot intervals with the B-spline filter ($p = 2$) for cantilever beam.

density filter			B-spline filter		
R/h	S^d(Es/Re)	time(s)	Δ/h	S_t^b(Es/Re)	time(s)
1.5	29.3/29MB	0.006	3	9.0/9.0MB	0.014
2	29.3/29MB	0.006	4	9.0/9.0MB	0.014
3.5	231/233MB	0.011	6	9.0/9.0MB	0.014
5	450/452MB	0.018	12	9.0/9.0MB	0.014
R/h	$32n_e\pi\lceil R\rceil^3/3$	$O(R^3)$	Δ/h	$8n_e(3p+4.5)$	$O(p^3)$

In Table 2, the time for computing the sensitivity (17) and estimated and real storage cost for density filter and for storing tensor-product form of B-spline weights are given. We record the time and storage cost for each optimization iteration and list the average value in the table. It can be seen that the estimated storage cost for the density filter is close to actual storage cost since we approximate the filtering region as a sphere. On the other hand, the estimated storage cost and actual cost for B-spline filters are identical since we know precisely the number of B-splines entries to store. For the density filter, with the increase of radius R, the resulting features become larger and the storage cost has increased from 29MB to 450MB. With B-spline filtering, with the decrease of the number of knot intervals from $28 \times 36 \times 112$ to $7 \times 9 \times 28$ with the corresponding effective filter size Δ/h changing from 3 to 12, the features become larger and the storage cost remain constant 9MB.

The CPU time and memory usage for B-spline based topology optimization are also plotted in Fig. 10 in terms of four tasks in topology optimization: finite element analysis, optimization, sensitivity analysis (16), and B-spline evaluation(BE) (17) by dynamically evaluating $N_i(\mathbf{x}_e)$ with different knot interval numbers are given in Fig. 10 and Table 2. For B-spline based topology



(a) CPU time

(b) Memory usage

Figure 10: The plot of CPU time and memory usage of B-spline representation for cantilever beam problem with different number of knot intervals.

optimization, the degrees are fixed at $2 \times 2 \times 2$, only the number of B-spline knot intervals change. From Fig. 10, we can see that with the decrease of knot intervals from $D28 \times 36 \times 112$ to $D7 \times 9 \times 28$, the feature size in optimized structures become larger. However, the cost for storing B-spline weights remain the same as shown in Fig. 10 and in Table 2. This is evident based on (22). We see that since the B-spline degrees are the same, changing the number of B-spline knot intervals does not affect the memory usage or CPU time.

In comparison, with the density filter, the memory and CPU time for storing and computing filtering weights is shown on the left column of Table 2. We can see that the memory usage is proportional to R^3 and CPU time is proportional to R^3.

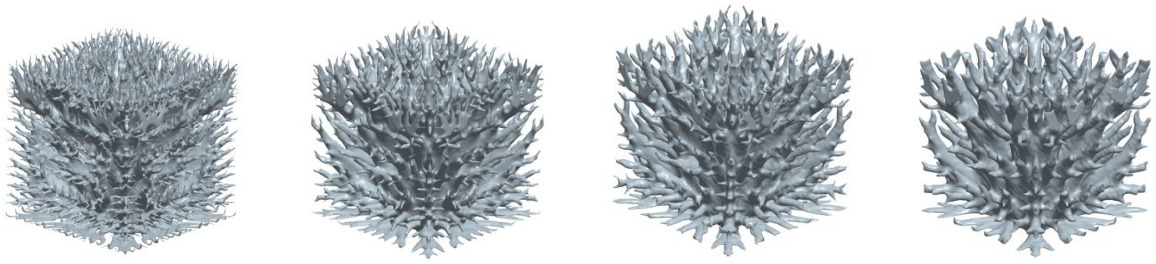
Heat conduction problem

Figure 11 shows optimized distribution of heat conduction materials with quadratic ($p = q = r = 2$) B-splines of different number knot intervals. One million linear 8-node hexagonal elements are used for FE analysis. The obtained topologies are very complex. To see it more clearly, surface smoothing is used to render the designs more clearly. As we can see, with the reduction of the number knot intervals, i.e. the increase of effective filter size $\Delta_x = (p+1)\delta_x$, the feature size becomes larger. Also listed is the memory required for storing tensor product form of B-spline weights. They are the same, 84MB, regardless the number of intervals, as predicted in (22) by substituting the analysis element number 1 million and degree 2.

The CPU time and memory usage for B-spline evaluation is shown in Fig. 12. As in minimum compliance problem, the changing of the number of knot intervals does not affect the memory usage and CPU time. The CPU time and memory usage of each components of topology optimization is well balanced to achieve different feature size. Optimization based on density filter on 1 million analysis elements is not completed since there is not sufficient memory to store the weights.

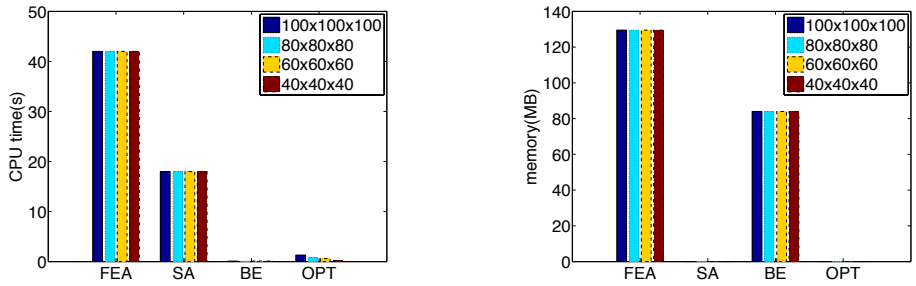
Multiple density per element scheme

Due to its efficiency in filtering, our B-spline based design representation shows significant efficacy for topology optimization schemes where significant filtering is involved. One such scheme is multi-density topology optimization[12] where it was shown that using multiple densities at each finite



(a) $p2, D100 \times 100 \times 100$, memory:84MB (b) $p2, D80 \times 80 \times 80$, memory:84MB (c) $p2, D60 \times 60 \times 60$, memory:84MB (d) $p2, D40 \times 40 \times 40$, memory:84MB

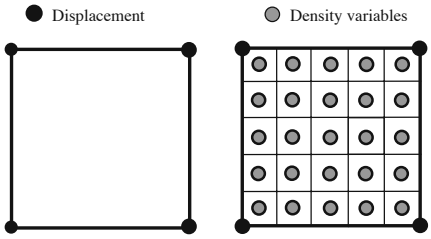
Figure 11: Topology optimization results of heat conduction problem with degree 2 B-splines of different knot interval numbers. The analysis mesh is $100 \times 100 \times 100$.



(a) CPU time (b) Memory usage

Figure 12: The plot of CPU time and memory usage of B-spline representation with different number of knot intervals for heat conduction problems.

element leads to much smoother boundary in the optimized designs. In Fig 13, a quadrilateral 4 nodes finite element with 5×5 multiple density point on each element(Q4/M25) is shown [12]. However, by packing a large number of density points, the memory cost for filter is proportional to the quardartic order of the number of density m in each direction. Table 3 compares the estimated storage of the cost of using density filter and B-spline filter. With the density filter, the MTOP scheme requires storage that is quadratic with respect to the number of density variables M per element where $M = m^3$ assuming same number of density variables m in each axial direction. This is because there is M times increase in the number of density variables and M times increase in each density variable's neighboring variables. On the other hand, with B-spline filter, the storage cost is independent from the number of neighboring variables and the storage cost is linear with respect to M. For 1 million finite elements, using cubic eight-node finite elements with $4 \times 4 \times 4 (M = 64)$, the memory usage comparison between density filter and tensor-product form of B-splines by using the same degree $p = q = r = 2$ and different knot intervals is shown. We assume that for B-spline filter, the effective interval sizes $\Delta = (p + 1)\delta$ is adjusted so they have the same effective filter size $(2R)$ in density filter. As shown in Table 3, when the filter radius R increase from 3 to 10, the memory usage increase so fast(proportional to the cubic of filter radius R) and a common desktop computer cannot meet this kind of memory requirement. Using B-spline representation, however, the storage only scales linearly with respect to M.



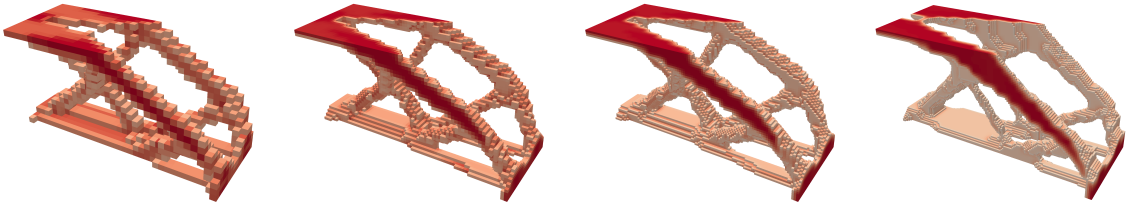
(a) MTOP scheme

Figure 13: Illustration of multiple density Q4/M25 [12].

Table 3: For 1 million finite elements, using cubic eight-node finite elements with $4 \times 4 \times 4 (M = 64)$, the memory usage comparison between density filter and tensor-product form of B-splines ($p = q = r = 2$) by using the same degree and different knot intervals that achieve the similar feature size.

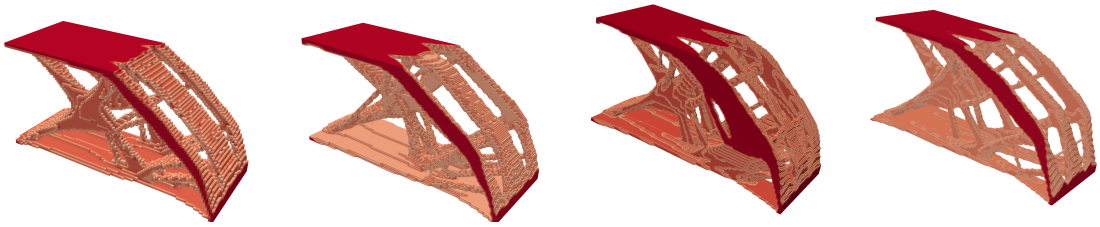
density filter		B-spline	
R/h	S^d	Δ/h	S_t^b
3	3.7TB	6	5.3GB
5	17TB	10	5.3GB
10	137TB	20	5.3GB
R/h	$32 n_e \pi R^3 M^2/3$	$(p+1)\delta/h$	$8(p+q+r+4.5) n_e M$

The optimization results using density filter are shown in Fig. 14 with display threshold 0.1. Analysis mesh $14 \times 18 \times 36$ is used. This is the finest analysis mesh we can use before out of memory issue happens. We can see that with more density variables per element, the boundary becomes



(a) $M = 1$, c=5656 (b) $M = 2^3$, c=5777 (c) $M = 3^3$, c=6479 (d) $M = 4^3$, c=7543

Figure 14: Density filter with analysis mesh: $14 \times 18 \times 36$. $R/h = 2$.



(a) M=1,c=3697 (b) M=2^3,c=3637 (c) M=3^3,c=3884 (d) M=4^3,c=4126

Figure 15: Using quadratic B-splines with analysis mesh $42 \times 54 \times 108$.

smoother. In Fig. 15, optimization results using B-spline representation are shown. Much finer analysis mesh $42 \times 54 \times 108$ can be used. By using finer mesh, more detailed structures with lower compliance are obtained. This example demonstrates B-spline filtering can demonstrate the full advantages of MTOP without encountering filter storage challenge that are otherwise prohibitively expensive with the usual density filter.

Conclusions and extensions

In this paper, we have extended the use of B-spline based density representation to three-dimensional topology optimization problems, including both minimal compliance and heat conduction. Our study shows that there is no checkerboard in optimized structures with proper choice of B-spline knots and degrees. The feature size can be controlled via B-spline degrees and knot interval numbers. Further, topology optimization with B-spline based density representation can be both memory and CPU efficient due to the tensor product nature of density distribution. Via dynamic reconstruction of the density through the tensor product, the usual cubic order of storage cost with respect to the filter size is reduced to linear order. It thus avoids the usual challenge of storing the large neighborhood distance matrix for filtering the density variables. This B-spline based representation is thus particularly useful for topology optimization methods such as multi-density per element scheme where the storage of the large number of neighboring distance between the large number of density variables have been a challenge.

The proposed B-spline based filtering of density variables can be readily extended to third-part topology optimization software. As illustrated in Algorithm 1, the required input is the center positions of elements, which can be easily obtained from most FEA and topology optimization

software. The output are $(p+1)$, $(q+1)$ and $(r+1)$ univarite B-splines weights, respectively in u, v and w directions for each element. The procedure for computing such univariate B-splines weights is relatively straightforward, with Matlab source code available online [21]. Tensor product of such univariate weights leads to the B-spline weights in 3D domain.

Acknowledgment

This work is supported in part by the NSF grant #1200800 and grant #1435072. The authors are also thankful for helpful comments from anonymous reviewers. The authors also acknowledge the help from graduate student Jing Li for producing Figure 1.

References

[1] M.P. Bendsøe and N. Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197–224, 1988.

[2] X. Qian. Topology optimization in B-spline space. *Computer Methods in Applied Mechanics and Engineering*, 265(0):15 – 35, 2013.

[3] M.P. Bendsøe. Optimal shape design as a material distribution problem. *Structural and Multidisciplinary Optimization*, 1(4):193–202, 1989.

[4] M. Zhou and G. Rozvany. The coc algorithm, part ii: topological, geometrical and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1):309–336, 1991.

[5] M.P. Bendsøe and O. Sigmund. *Topology optimization: theory, methods, and applications*. Springer Verlag, 2003.

[6] B. Bourdin. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50(9):2143–2158, 2001.

[7] O. Sigmund. Design of material structures using topology optimization. *Report S69, Danish Center for Applied Mathematics and Mechanics, Technical University of Denmark, Lyngby, Denmark*, 1994.

[8] R.B. Haber, C.S. Jog, and M.P. Bendsøe. A new approach to variable-topology shape design using a constraint on perimeter. *Structural and Multidisciplinary Optimization*, 11(1):1–12, 1996.

[9] T. Borrvall. Topology optimization of elastic continua using restriction. *Archives of Computational Methods in Engineering*, 8(4):351–385, 2001.

[10] E. Andreassen, A. Clausen, M. Schevenels, B.S. Lazarov, and O. Sigmund. Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16, 2011.

[11] B.S. Lazarov and O. Sigmund. Filters in topology optimization based on helmholtz-type differential equations. *International Journal for Numerical Methods in Engineering*, 86(6):765–781, 2011.

[12] T.H. Nguyen, G.H. Paulino, J. Song, and C.H. Le. A computational paradigm for multiresolution topology optimization (mtop). *Structural and Multidisciplinary Optimization*, 41(4):525–539, 2010.

[13] A.V. Kumar and A. Parthasarathy. Topology optimization using B-spline finite elements. *Structural and Multidisciplinary Optimization*, 44:1–11, 2011.

[14] Jiaqin Chen and Vadim Shapiro. Optimization of continuous heterogeneous models. In Alexander Pasko, Valery Adzhiev, and Peter Comninos, editors, *Heterogeneous objects modelling and applications*, volume 4889 of *Lecture Notes in Computer Science*, pages 193–213. Springer Berlin Heidelberg, 2008.

[15] Jiaqin Chen, Vadim Shapiro, Krishnan Suresh, and Igor Tsukanov. Shape optimization with topological changes and parametric control. *International journal for numerical methods in engineering*, 71(3):313–346, 2007.

[16] L.A. Piegl and W. Tiller. *The NURBS Book*. Springer Verlag, 1997.

[17] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2001.

[18] Pysparse. `http://pysparse.sourceforge.net`. Accessed: 09-24-2014.

[19] K. Svanberg. The method of moving asymptotes–a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987.

[20] W. Hunter. *Predominantly solid-void three-dimensional topology optimisation using open source software*. PhD thesis, Stellenbosch: University of Stellenbosch, 2009.

[21] Matlab source code for topology optimization in B-spline space. `http://cdm.me.wisc.edu/code/btop85.htm`. Accessed: 10-10-2014.