Evaluation of Topology Optimization Filtering with Numeric Examples

Prepared By:

John A. Morelli, ATA Engineering Inc.

AOE 5904, Fall 2019

Prepared For:

Dr. Robert Canfield, Virginia Tech

Abstract

Commonly used filtering algorithms and settings for addressing the checkerboarding problem inherent in the continuous density-based topology optimization approach (e.g., the Solid Isotropic Material with Penalization or SIMP method) are discussed. A modification to the optimality criteria recursion relationship is shown, and an alternative formulation of optimality criteria (OC) suitable for use with a linear density filter is provided. A MATLAB implementation of a 2D compliance minimization problem using OC and the Globally Convergent Method of Moving Asymptotes (GCMMA) is used to compare the effects of using different filter weights, using a sensitivity or a linear density filter, and increasing the size of the filter. The 2D compliance problem is solved using commercially available topology optimization software including ANSYS workbench, TOSCA Structure, MSC Nastran, and Simcenter Nastran; comparisons are made to the MATLAB implementations of the problem to make inferences about the filter algorithms used in the commercial software. A 3D compliance problem is solved using Simcenter Nastran 2019.2 and TOSCA Structure 2019, and recommendations regarding element formulation and filter settings are provided.

Table of Contents

1. Introduction and Background

The topology optimization problem aims to determine the configuration of isotropic material within a prespecified spatial domain that optimally satisfy some user-defined objectives and constraints. Topology optimization is often described in contrast to *sizing* or *shape* optimization. In a sizing or shape optimization problem, the general layout of the structure is already determined, and the optimization seeks to determine the cross-sectional properties of known member locations or locations of grid points to satisfy design requirements. By contrast, the topology optimization problem seeks to determine the structure's connectivity, shape, and locations of voids without presupposing any information about the structural layout apart from the domain over which the design must exist and user-defined loads and boundary conditions.

Topology optimization is an increasingly active area of research and development which has seen a resurgence in attention as the complex designs that result from topology optimization algorithms may become actualized through advances in optimization and additive manufacturing techniques. It is well known that the topology optimization problem is ill-posed, and therefore closed form solutions for the optimal structural configuration do not exist [1]. Robust, reliable topology optimization tools are therefore critical to overcome the many challenges associated with the complex numerical solution to the topology optimization problem.

In practical applications, structural topology optimization is commonly implemented through so-called relaxed penalization (RP) approaches, sometimes referred to as *density-based* approaches, in conjunction with the finite element method. In an RP approach, the original topology optimization problem (the binary existence of either material or void, so-called 0-1 design) is *relaxed* using a continuous variable that represents the existence of material, often referred to as the material pseudo-density. The material pseudo-density is directly related to the mass and stiffness of individual finite elements in RP approaches through an interpolation scheme. The continuous material pseudo-density may be *penalized* in a way such that non-physical intermediate densities are uneconomical and therefore topology optimization solutions are incentivized towards binary 0-1 designs. The most well-known RP interpolation scheme is known as Solid Isotropic Material Penalization (SIMP) and is the basis for all methods and results discussed in this paper [2]. The SIMP approach has been applied extensively in both academia and industry because of the simplicity of its implementation and repeated success of results.

The goal of this paper is to increase the understanding of the effect of various filter types and filter parameters to addressing the well-known numerical problems of checkerboarding and mesh-dependence. Methods of filtering the material pseudo-density or filtering of the objective sensitivity are applied to a compliance minimization (stiffness maximization) example problems subject to a volume constraint. Furthermore, the way in which popular commercial topology optimization programs address the checkerboarding and mesh-independence problems is assessed, and several recommendations are provided.

The paper is organized as follows. First, a brief overview of the compliance minimization problem using the SIMP method is outlined in Section 1.1. The numeric instabilities inherent in topology optimization RP approaches are discussed in Section 1.2. The filtering methods that are used to address the checkerboarding and mesh-dependence problems are discussed in Section 2. In Section 3, an updated form the power law used in optimality criteria recursion relationship is provided. Optimality criteria for the compliance minimization problem in conjunction with a linear density filter are also derived. A brief overview of topology optimization approaches implemented in several popular commercial codes, including ANSYS, TOSCA Structure, MSC Nastran, and Simcenter Nastran, is provided in Section 4. Section 5 provides comparisons of topology optimization results and solutions times for varying filter types and parameters, including the optimality criteria with a linear density filter for a simple two-dimensional example problem implemented in a MATLAB environment. Topology optimization results for both 2D and 3D implementations of the compliance minimization problem using popular commercial software are also provided in Section 5.

1.1. Compliance Minimization Problem using SIMP

Compliance minimization is a popular choice for studying topology optimization with finite elements in part because the sensitivity of compliance is self-adjoint and therefore inexpensive to accurately calculate. Compliance minimization is also useful from a practical standpoint. Because compliance is a measure of structural flexibility, minimizing compliance implies the stiffness of the structure is being maximized. Compliance minimization is often used in conjunction with a limit on the structural volume. Thus, the compliance minimization problem answers the question: what is the stiffest possible structure that can exist within the prescribed material limit, for the prescribed loads and boundary conditions.

In the finite element method, the global stiffness matrix for a structure, $\boldsymbol{K}$, and the nodal displacement vector, D, are related to the nodal load vector R as shown in Equation 1-1. The solution for the displacement vector may be readily obtained through standard matrix operations and used to compute the elemental strain energies, u_e, as shown Equation 1-2, where $\boldsymbol{K_{0e}}$ is a sparse matrix that represents a given elemental stiffness matrix blown up to the appropriate size and degrees of freedom of the global stiffness matrix.

$$\boldsymbol{K}D = R \qquad \text{Equation 1-1}$$

$$u_e = \left({}^{1}\!/_{2}\right)D^T\boldsymbol{K_{0e}}D$$

$$\boldsymbol{K} = \sum_{e=1}^{N}\boldsymbol{K_{0e}} \qquad \text{Equation 1-2}$$

The compliance of structure, C, may be defined as the dot product of some weighting vector $\bar{\boldsymbol{R}}$ and the displacement vector, $\boldsymbol{D}$, from the finite element solution. When the compliance weighting vector is chosen to be the applied load vector, then the structural compliance is twice the element strain energies summed over all elements in the structure, as shown in Equation 1-3.

$$C = \bar{R}^T D$$

$$C = D^T\boldsymbol{K}D = \sum_{e=1}^{N} D^T\boldsymbol{K_{0e}}D = 2\sum_{e=1}^{N} u_e \quad for \quad \bar{R} = R \qquad \text{Equation 1-3}$$

The compliance minimization problem may then be posed as shown in Equation 1-4, where x_e are the pseudo-density design variables. The volume or material resource constraint, $g(x_e)$, is the dot product of the pseudo-density design variables and the nominal volume of each finite element, V_{0e}, and must be less than some user-specified target volume, V_f. N is the number of elements for which the topology optimization is performed over (the design domain). Side constraints are enforced on the pseudo-densities such that they must be between some arbitrarily small lower bound, x_{min}, in order to avoid singularities in the stiffness matrix decomposition, and one.

$$\min_{x_e} C(x_e) = 2\sum_{e=1}^{N} u_e(x_e)$$

$$s.t.: g(x_e) = \left(\sum_{e=1}^{N} (x_e V_{0e}) \Big/ V_f \right) - 1 \leq 0, \qquad \text{Equation 1-4}$$

$$0 < x_{min} \leq x_e \leq 1$$

In RP approaches, the pseudo-density design variables are used to linearly scale the volume (and, indirectly, the mass) of the finite elements, as shown in Equation 1-5, hence the term “pseudo-density”.

$$V_e = x_e V_{0e} \qquad \text{Equation 1-5}$$

The pseudo-density design variables are also used to interpolate the elemental stiffness matrices of each element within the design domain. There are several methods discussed in the literature and used in commercial software codes to perform this interpolation. By far the most popular and widespread stiffness interpolation approach is the SIMP method, which is shown in Equation 1-6. In the SIMP method, the modulus of elasticity for a given finite element, E_e, is taken as a nonlinear function of the nominal elastic modulus of the material, E_0, and the pseudo-densities, with a user-specified penalization factor, p, which must be greater than or equal to one.

$$E_e = E_0 {x_e}^p \qquad \text{Equation 1-6}$$

For linear elastic finite element analysis, the stiffness matrix of each finite element becomes a nonlinear function of the pseudo-density design variable for that element.

$$\boldsymbol{k}_e = {x_e}^p \boldsymbol{k}_{0e} \qquad \text{Equation 1-7}$$

The use of a power law interpolation method with a penalty larger than unity means that pseudo-densities between 0 and 1 are incentivized to move towards 0 or 1 solutions. Higher penalizations increase the concavity of the SIMP approximation, as shown in Figure 1-1, thereby increasing the cost of having intermediate pseudo-densities in any given iteration of the topology optimization.

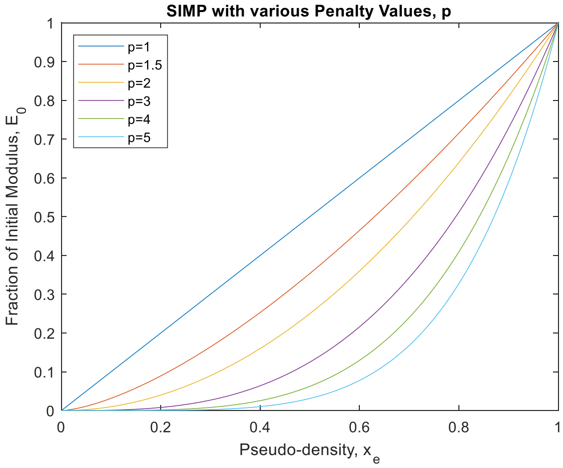


Figure 1-1. The choice of penalty value in the SIMP method increases the nonlinearity of the element stiffness with respect to the pseudo-density design variables. Increasing the penalty value makes intermediate pseudo-densities increasingly more expensive.

In order to perform topology optimization using either math programming (MP) methods or approaches based on optimality criteria (OC), the sensitivities of the objective and constraint functions with respect to the pseudo-density design variables are required. Differentiating the compliance with respect to the pseudo-density design variable yields:

$$\frac{\partial C}{\partial x_e} = \bar{R}^T \frac{\partial D}{\partial x_e} \qquad \text{Equation 1-8}$$

In Equation 1-8, the displacement vector sensitivity term may be computed directly using, for example, finite difference or complex step differentiation. However, because the number of design variables in topology optimization is typically very large, such numeric methods are inefficient, and the adjoint method is preferred. Bendsøe and Kikuchi first expressed the sensitivity of the stiffness field using the adjoint method in their seminal paper on topology optimization using a homogenization approach [3]. Sigmund later used the adjoint method to express the sensitivity of the compliance to element pseudo-densities in the form shown Equation 1-9 [4]. The $\boldsymbol{K_{0e}}$ and $\boldsymbol{K_e}$ terms in Equation 1-9 represent the un-penalized and the penalized element stiffness matrices, $\boldsymbol{k}_{0e}$ and $\boldsymbol{k}_e$, respectively, expanded into the global stiffness matrix (without contribution from any other elements). Because $\boldsymbol{k}_{0e}$ and $\boldsymbol{k}_e$ are related via SIMP interpolation law, the compliance sensitivity simplifies and the compliance sensitivity with respect to all pseudo-density design variables may be evaluated in a single operation that depends only on the finite element solution element strain energies and the current pseudo-density design variables.

$$\frac{\partial C}{\partial x_e} = -p{x_e}^{p-1}(D^T \boldsymbol{K_{0e}} D) = p{x_e}^{-1}(D^T \boldsymbol{K_e} D) = -2p{x_e}^{-1}u_e \qquad \text{Equation 1-9}$$

The sensitivity of the volume constraint with respect to the pseudo-density design variables is easily obtained and is shown in Equation 1-10 for completion.

$$\frac{\partial g}{\partial x_e} = V_{0e}/V_f \qquad \text{Equation 1-10}$$

1.1.1. Other Material Interpolation Methods

While not utilized further in this paper, another popular material interpolation method used in both literature and commercial software is the Rational Approximation of Material Properties (RAMP), which was originally developed by Stolpe & Svanberg and is shown in Equation 1-11 [5]. Note that q is typically used to denote the penalty parameter associated with the RAMP method.

$$E_e = E_0 \frac{x_e}{1 + q(1 - x_e)} \qquad \text{Equation 1-11}$$

As noted by Deaton & Grandhi, one advantage of using RAMP over SIMP is that the slope of the RAMP material modulus with respect to the pseudo-density design variables is greater than zero at very low values of the pseudo-density, as shown in the left hand side of Figure 1-2 [6]. Deaton & Grandhi state that the benefit of having non-zero slope at low pseudo-density is that if an element's pseudo-density becomes close to zero early on in an optimization, it is easier for it to return to a value greater than zero. This behavior is preferable in instances where the self-weight of the structure becomes important, such as for inertial loading or dynamics problems. These claims are not verified in this report, however.

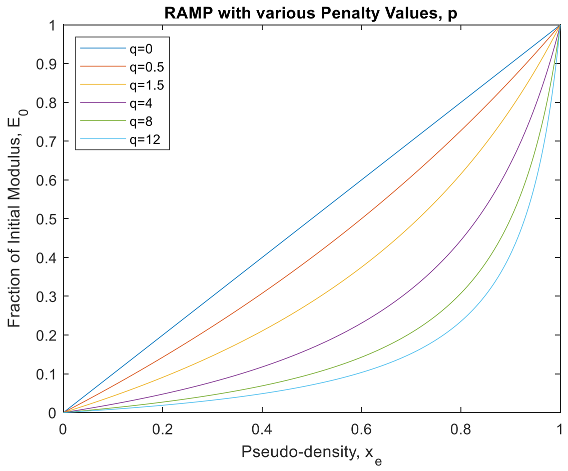


Figure 1-2. Compared to SIMP, the RAMP material interpolation method has larger gradients with respect to the pseud-density design variable at values of the pseudo-density near zero.

Other material interpolation laws, such as lattice-based interpolation, are available in software such as Simcenter Nastran 2019.2. While these methods are not well documented in the literature, the Simcenter Nastran 2019.2 documentation seems to suggest that these interpolation methods incentivize the design to proceed towards three separate states: solid (pseudo-density of 1), void (pseudo-density of 0), and lattice (intermediate pseudo-density) [7]. Examples of the lattice structures that make use of these intermediate pseudo-densities are shown in Figure 1-3. The behavior of these material interpolation methods was not investigated further in this paper.

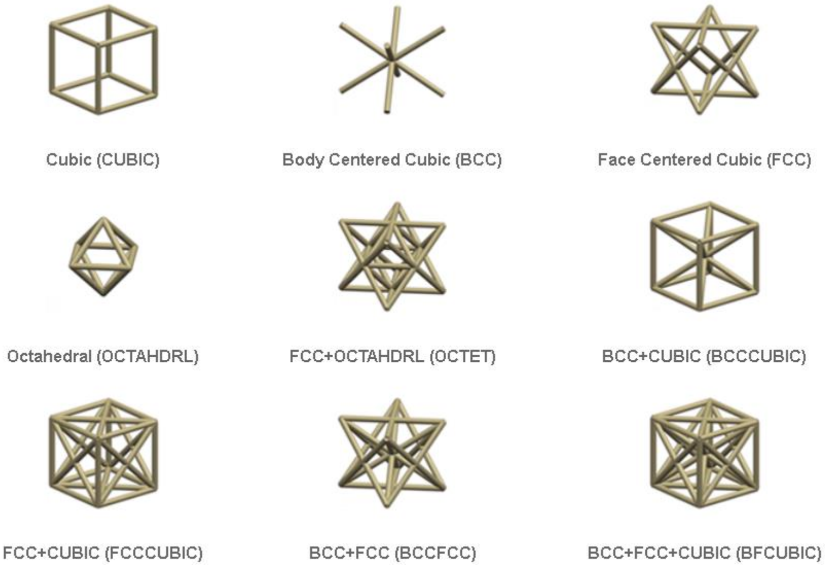


Figure 1-3. Lattice material interpolation methods allow for a third state in the topology optimization (solid, void, and lattice) are available in Simcenter Nastran 2019.2 [7][8].

1.2. Numeric Instabilities in Topology Optimization

There are three major numerical challenges associated with solutions to the topology optimization problem using RP approaches such as SIMP. First, the topology optimization problem lacks a unique solution in general. The second two numerical challenges of topology optimization problems are the loosely related issues of the so-called *checkerboard* problem and mesh dependence. All three issues are discussed in more detail in the following subsections.

1.2.1. Nonexistence

The numeric issue of nonexistence in the context of topology optimization was illustrated by Sigmund using the uniaxial bar example in Figure 1-4 [1]. Under a tensile load, if the total area for all structural members are the same, then the displacement solution for the uniaxial bar is unchanged. It can therefore be seen that an infinite number of configurations of bars may be used to represent the same displacement solution, provided that the summed area of all structural members is constant.

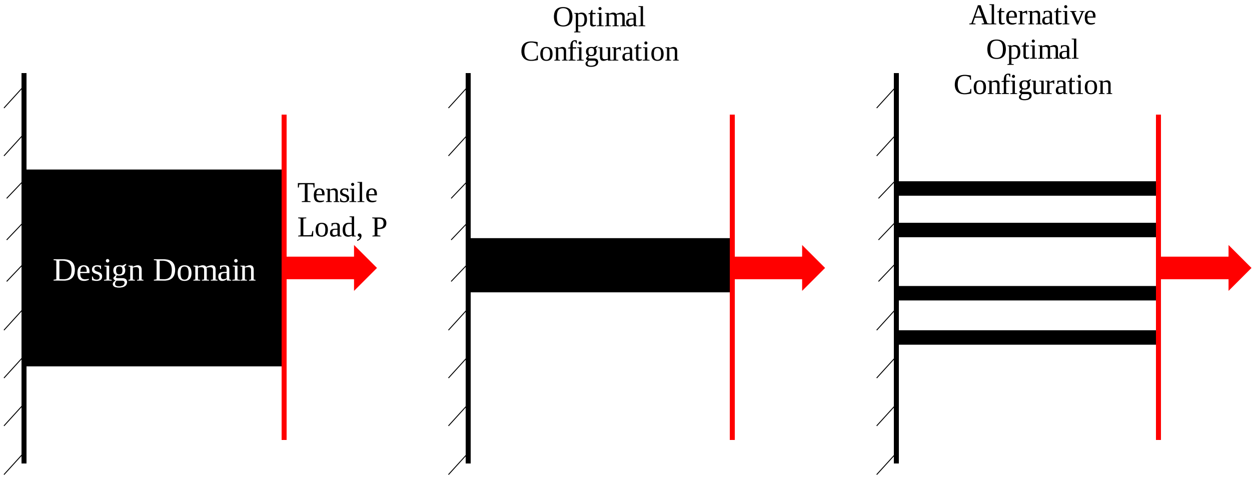


Figure 1-4. The uniaxial bar under tensile loading has an infinite number of material configurations that result in identical structural performance metrics, such as weight, displacement, and stress.

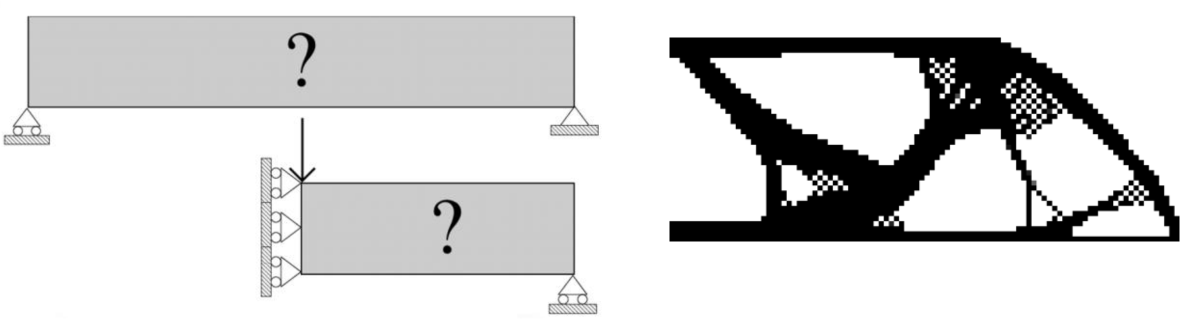
In topology optimization, proving convergence to a global optimum is generally not possible, as an infinite number of topologies may exist which exhibit similar performance. Often, it is difficult to even prove that the topology optimization converged to a local optimum. In practice, most commercially available topology optimization algorithms use convergence logic based on relatively weak convergence criteria, such as checking whether the objective function or design variables are no longer changing between iterates. A more robust convergence criteria based on the first-order gradient of the Lagrangian function is one of the Karush-Kunh-Tucker

(KKT) conditions of optimality [9]. Convergence based on the first-order gradient condition is more common in sizing optimization problems, but rarely implemented in topology optimization algorithms. In Section 5, it is shown that the KKT first-order gradient method may be satisfied in the presence of density filter for math programming methods. Section 3.1.2 describes how optimality criteria for the compliance minimization problem may be modified to account for a linear density filter which, in theory, could converge to a local optimum that satisfies the KKT first-order gradient condition.

1.2.2. Checkerboarding and Mesh-Dependence

The second two of three numerical challenges of topology optimization problems are the loosely related *checkerboard* problem and mesh dependence problems. While the checkerboard and mesh-dependence problems are distinct from one another, they are considered a related problem because the use of *restriction methods* tends to resolve both issues simultaneously. Restriction methods generally refer to approaches used to restrict the RP problem and reduce the effect of checkerboarding and mesh-dependence in the topology optimization. Several restriction methods, including filtering methods, are discussed in more detail in Section 2.

The checkerboard problem refers to the tendency of RP approaches to converge to 0-1 design where the element material pseudo-density varies from 0 to 1 in neighboring elements, resembling a checkerboard pattern. While this configuration was originally believed to represent some optimal microstructure, Díaz & Sigmund and Jog & Haber have shown that the checkerboard patterns are a numeric artifact of the finite element discretization of the problem, and do not represent an optimal microstructure, as originally believed [10][11]. The checkerboard problem is clearly visible in two-dimensional problems, such as the 2D half MBB compliance minimization example shown in Figure 1-5. The topology optimization result shown Figure 1-5 was generated using Sigmund’s popular 99-line MATLAB implementation of the SIMP approach [4].



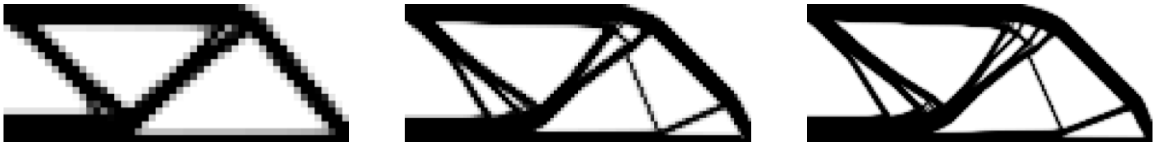
2D Half MBB
Problem Setup

Compliance Minimization Result Subject to
40% Volume Constraint
100 x 40 Elements

Figure 1-5. (Left) boundary conditions for the compliance minimization problem for the 2D half MBB compliance minimization example problem [(Sigmund 2001)]. (Right) The solution to compliance minimization problem includes the checkerboard effect.

Sigmund & Petersson noted that theoretical studies of checkerboards appearing in three-dimensional problems have not yet been carried out [1]. In Section 5.6 of this paper, a compliance minimization problem is carried out in three-dimensional space using commercial software to illustrate the checkerboarding effect for several different finite element types. Many different authors have also suggested using higher-order finite elements in the displacement solution of the topology optimization to avoid the checkerboard problem. Díaz & Sigmund showed that using 8 or 9-node quadrilateral elements can eliminate checkerboarding using the SIMP approach, but only for relatively low penalization factors [10]. Several examples showcasing the effect that higher-order finite elements have on reducing the effect of the checkerboard problem for a compliance minimization problem are provided in Section 5. In general, higher-order elements tend to reduce the checkerboarding effect at the cost of computational time. However, the amount by which higher-order elements reduce the checkerboard effect depends on the specific element formulation and finite element software.

The mesh-dependence problem is illustrated using the 2D MBB half-beam example problem in Figure 1-6. Each of the topologies shown in Figure 1-6 were generated using Sigmund's 99-line MATLAB implementation of the SIMP approach [4]. The topology solutions shown in Figure 1-6 each make use of a compliance sensitivity filter (discussed in detail in Section 2.3) to address the checkerboard problem. Ideally, increasing the mesh density ought to result in a better discretization of the same optimal structure. However, it can be seen in Figure 1-6 that increasing the mesh density results in qualitatively different optimal structures. As the mesh density increases, finer and finer structural members may present themselves in the optimal solution. Mesh dependence is also related to the nonexistence of an optimal solution. That is, the precise discretization used to represent a structural domain may converge to a locally optimal solution. However, a different discretization of the same structural domain may converge to a different locally optimal solutions.



Optimal Topology
50 x 20 Elements

Optimal Topology
100 x 40 Elements

Optimal Topology
150 x 60 Elements

Figure 1-6. The optimal topologies generated using SIMP approaches are dependent on the finite element discretization. A finer mesh discretization permits finer structural members to exist in the optimal topology.

2. Restriction Methods

Several restriction methods have been suggested in the literature to eliminate the checkerboard effect. Many of the proposed restriction methods have the secondary effect of introducing a mesh-independent length scale into the topology optimization problem, thereby reducing, but not eliminating, the effect of mesh dependency. A brief overview of several lesser-used restriction methods, including perimeter and gradient constraints, is provided in this section. Filtering methods are the focus of this paper, and a more detailed discussion of filtering methods is provided in the following subsections. Specifically, discussion is focused on the original sensitivity filter and the linear density filter. Several advanced filters have been proposed in the literature (e.g., morphological filters proposed by Sigmund [13] and filters based on Pythagorean means proposed by Svanberg & Svärd [14]) and a brief overview of these methods is provided in this paper. However, these filters are not implemented in any of the commercially available topology optimization software packages evaluated in this paper and so are not evaluated in detail in this paper.

2.1. Perimeter Control and Gradient Constraints

The general idea behind perimeter control is to constrain the sum of the lengths or areas of all inner and outer boundaries of the structural topology. When used with the SIMP approach, perimeter control amounts to constraining the *total variation* $TV(\rho_i)$ of the pseudo-density design variables, ρ_i, to be less than some prescribed perimeter value. The use of a perimeter control adds a single extra constraint to the topology optimization problem. While the addition of a single extra constraint is not prohibitive, Sigmund [13] notes that selecting an appropriate perimeter to use within the constraint is difficult. Often, continuation approaches, where the perimeter constraint value is slowly changed, are required to show convergence. The existence of solutions to the topology optimization using perimeter control was proven by Ambrosio & Buttazzo [15].

Local and global gradient constraints have also been shown to assure existence of solutions and convergence of the finite element scheme [1]. Gradient constraints have the added benefit of introducing a well-defined length scale that is effective in reducing the mesh-dependency issue.

The local gradient constraint restricts the spatial variation in the pseudo-density to be bounded by some value, G, as shown in Equation 2-1. This constraint implies that any solid-to-void transition in the structure must occur over a distance larger than $2/G$. Implementing the local gradient constraint directly is prohibitively expensive as it introduces two to three times as many extra constraints as there are design variables (elements) in the topology optimization problem. The infinity norm of the local gradient constraint shown in Equation 2-1 may be used in place of individual local gradient constraints such that a single constraint may be used in the optimization. However, in such a scenario, the value used in the constraint is unknown beforehand, and must be determined through experimentation.

$$\left|\frac{\partial \rho_i}{\partial x_j}\right|_\infty \leq G \; for \; j = 1,2,3 \qquad \text{Equation 2-1}$$

In the global gradient constraint, the norm of the material pseudo-density in Sobolov Space $H^1(\Omega)$ over the design domain, Ω, is restricted to some value M, as shown in Equation 2-2. Like the norm of the local gradient constraint, the global gradient constraint also requires careful numeric experimentation to find determine an appropriate limit M, which is problem-specific and unknown beforehand.

$$\int_{\Omega} \sqrt{(\rho_i{}^2 + |\nabla\rho_i|^2)} \mathrm{d}\Omega \leq \mathrm{M} \qquad \text{Equation 2-2}$$

Perimeter control and gradient constraints are each effective at addressing the nonexistence and checkerboarding problems inherent in topology optimization. However, each method has the drawback of requiring significant amounts of experimentation to determine the appropriate problem-specific constraint value. In problems of practical importance, this kind of numeric experimentation is prohibitively time-consuming. For these reasons, both perimeter control and gradient constraints are not implemented in commercially available software and were not evaluated further throughout this paper.

2.2. Filter Weighting

Sensitivity and density filters, which are discussed in detail in Sections 2.3 and 2.4, utilize a matrix of weighting values that incorporates the sensitivities or densities of neighboring elements. The values of elements within the weighting matrix are dependent upon a user-defined filtering radius as well as the distances between element centroids in each finite element discretization. For centroid to centroid distances greater than the user-defined filter radius, the weighting value is zero. Several forms of the weighting matrix were studied in this paper, including constant weighting, linear (or conic) weighting, and Gaussian weighting. While it was determined that the filter weighting has a relatively minor impact on the topology optimization, each weighting method is discussed in more detail below for completion. Note that because the weighting matrix is dependent only on geometry of the design domain finite element discretization, it needs calculated only once for a given topology optimization problem.

The simplest weighting implementation, referred to as constant weighting, defines a constant weighting value based on the number of elements contained within the user-defined filter radius, as shown in Equation 2-3. In Equation 2-3, R is the user-defined filter radius, $d(i,j)$ is a function that returns the centroid-to-centroid distances between the i^{th} and j^{th} elements, and N_i is the set of elements within the filtering radius for the i^{th} element, and $|N_i|$ is the number of elements within the filter radius.

$$w_{ij} = \begin{cases} \dfrac{1}{|N_i|}, & j \in N_i \\ 0, & j \notin N_i \end{cases} \qquad \text{Equation 2-3}$$

Another common method of weighting in topology optimization filters is to use linearly or conically varying values as shown in Equation 2-4; linear weighting assigns a higher weighting value for elements located near the current element being filtered.

$$w_{ij} = \begin{cases} \dfrac{R - d(i,j)}{\sum_{k \in N_i}(R - d(i,k))}, & j \in N_i \\ 0, & j \notin N_i \end{cases} \qquad \text{Equation 2-4}$$

A simple visualization of the filtering radius is shown in Figure 2-1 for a five by two finite element discretization. In Figure 2-1, the dashed circle represents the user-defined filter radius, and orange coloring indicates that a given element centroid lies within the filter radius from the centroid of element 7. The 7th row of the weighting matrix therefore contains non-zero entries in columns 2, 6, 7, and 8. For the case of a constant weighting matrix, each of these non-zero entries are a constant equal to ¼. For the case of a linear weighting matrix, the seventh column contains a larger value than columns 2, 6, and 8, and the exact values of the weighting matrix will depend on the distances between the centroid of element 7 and its neighboring elements, as well as the user-defined filter radius.

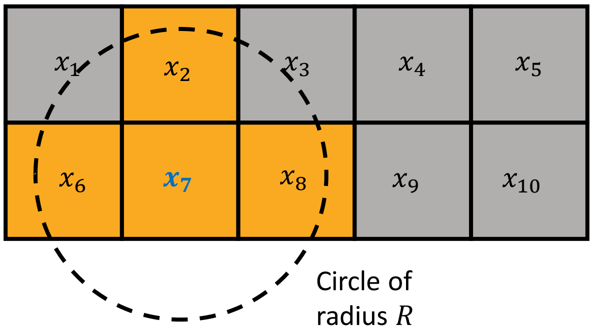


Figure 2-1. Simple example of the weighting matrix calculation for a finite element model. For the 7th row of the weighting matrix, nonzero entries exist for columns (yellow elements) whose centroid lies within the filter radius, R, from the 7th element.

A Gaussian distribution for weighting density filters shown in Equation 2-5 was also suggested by Bruns & Tortorelli [17]. The Gaussian distribution weighting matrix may be calculated using Equation 2-5; σ_d is the Gaussian variance, which may be described by either $R/2$ or $R/3$. For both choices of Gaussian variance, the filter weights approach zero for centroid-to-centroid distances greater than R. Sigmund [13] notes that no advantages were observed for the Gaussian weighting scheme compared to the linear weighting method. This observation was corroborated by the author using several test cases; consequently, the Gaussian weighting is not considered in any comparisons shown in the remainder of this paper.

$$w_{ij} = e^{\frac{-1}{2}\left(\frac{d(i,k)}{\sigma_d}\right)^2} \qquad \text{Equation 2-5}$$

2.3. Sensitivity Filters

Sensitivity filtering is a popular choice in both commercial software and literature, owing to its simple formulation and computationally inexpensive implementation [2]. Literature and test cases presented in this paper confirm that sensitivity filters are effective at eliminating the checkerboard problem and introducing a length scale that help overcome the mesh-dependence problem. In general, sensitivity filters operate by taking a weighted average (via a weighting matrix described in Section 2.2) of the sensitivity of the objective function (the compliance) with respect to the element pseudo-densities. The process of applying a sensitivity filter may be summarized as follows.

(1) The objective is calculated using finite element analysis and the objective sensitivity with respect to the design variables is calculated using any method (e.g., adjoint sensitivity).

(2) For each design variable (each finite element), a weighted average of the objective sensitivity is calculated using one of the weighting methods described in Section 2.2.

(3) The weighted average of objective sensitivities is passed on to the design updating scheme (e.g., optimality criteria or math programming) to determine a new design point.

Sensitivity filters are heuristic methods. To date, no proof or explanation of the physical meaning behind why a sensitivity filter works has been provided. Furthermore, the use of a weighted average objective sensitivity in place of the actual objective sensitivity may introduce problems when using a search to calculate the next design point or attempting to calculate the KKT gradient-condition. Nevertheless, sensitivity filtering remains a popular option for addressing both checkerboarding and mesh-dependence numeric problems. In the following subsection, the sensitivity filter used widely throughout literature is shown. Several alternative forms of the sensitivity filter were proposed in the literature and are summarized by Sigmund [13]. Each of the alternative sensitivity filters rely on a similar heuristic approach for averaging the objective sensitivities. While some alternative sensitivity filters offered subjectively increased filter performance (e.g., fewer perceived checkerboards mesh dependence), the benefits appeared to be present only in very specific example test cases. Consequently, the alternative sensitivity filters were not deemed worthwhile to consider as part of the assessments in this paper.

2.3.1. Original Sensitivity Filter

The original sensitivity is shown in Equation 2-6 [13]. The original sensitivity filter is simply a weighted average of the product of the pseudo-density and the nominal (un-filtered) objective sensitivity to the pseudo-density. When implemented, an arbitrarily small lower bound greater than zero is usually enforced on the pseudo-density, thus avoiding division by zero. The form of the sensitivity used by Sigmund in his 99-line code implementation of the compliance minimization problem makes use of a conic weighting matrix, as described by Equation 2-4 [4].

$$\frac{\widetilde{\partial f}}{\partial x_e} = \frac{\sum_j w_{ij} x_j \frac{\partial f}{\partial x_j}}{\rho_e \sum_j w_{ij}}$$

Equation 2-6

Throughout this paper, the form of the sensitivity filter shown in Equation 2-6 is used. The original sensitivity was later modified by Sigmund [13] to account for non-regular mesh densities as shown in Equation 2-7, where V_{0j} is the nominal volume of the j^{th} finite element. Because the example compliance minimization problem used throughout this paper contains a discretization with uniform element sizes, the use of Equation 2-6 instead of Equation 2-7 is not expected to impact topology optimization results.

$$\frac{\widetilde{\partial f}}{\partial x} = \frac{\sum_j w_{ij} x_j \frac{\partial f}{\partial x_j} {V_{0j}}^{-1}}{x_e {V_{0e}}^{-1} \sum_j w_{ij}}$$

Equation 2-7

2.4. Density Filters

Density filters operate by modifying the pseudo-density design variable for each element to be dependent on the pseudo-densities of neighboring elements. The modified (or filtered) pseudo-densities are commonly referred to as the *physical* design variables because they are subsequently used directly in the simulation of the underlying problem physics with, for example, finite element analysis. In the context of a density filter, un-filtered pseudo-densities have no physical meaning. However, the un-filtered pseudo-densities are still considered the design variables of the problem, and it is the un-filtered pseudo-densities that are updated by, for example, math programming methods. For that reason, the un-filtered pseudo-densities are sometimes referred to as the *optimization* design variables.

In contrast to sensitivity filters, density filters are not heuristic methods. Consequently, line search optimization algorithms as well as convergence criteria based on the KKT first-order gradient condition are both possible provided the filter and its associated sensitivity chain rule terms are properly implemented in the given density filter. The process of applying a density filter may be summarized as follows.

(1) Starting at some initial design, the nominal (un-filtered) pseudo-densities, x_e, are passed to a filtering function, $\mathcal{F}$, yielding the filtered pseudo-densities, $\widetilde{x_e}$. That is, $\widetilde{x_e} = \mathcal{F}(x_e)$

(2) Elemental mass and stiffnesses are updated using the filtered pseudo-densities, $\widetilde{x_e}$, and with, e.g., the SIMP method, and finite element analysis is performed.

(3) The objective and constraint functions are calculated using outputs from the finite element analysis as well as the filtered pseudo-densities, $\widetilde{x_e}$, if required.

(4) The objective and constraint sensitivities with respect to the un-filtered pseudo-densities, ρ_e, are calculated. In this step, it is important to account for chain rule terms associated with differentiating the filtering function with respect to the un-filtered element pseudo-densities. That is, $\frac{\partial f}{\partial x_j} = \frac{\partial f}{\partial \widetilde{x_j}} \frac{\partial \mathcal{F}}{\partial x_j}$. Depending on the nature of the density filter being considered, this step can become computationally prohibitive to implement.

(5) The objective, constraint, and sensitivities with respect to the un-filtered pseudo-densities are used to update the current design point. Math programming methods or optimality criteria (OC) may be used to determine the un-filtered pseudo-densities at the next iteration.

(6) Steps (1) through (5) are repeated until convergence is satisfied.

2.4.1. Linear Density Filter

A number of different density filters have been proposed and studied in the literature [13][14]. The most popular and widely used density filter is the aptly named linear density filter, which modifies existing element pseudo-densities to be linearly weighted based on the pseudo-densities of neighboring elements. The linear density filter was first introduced by Bruns & Tortorelli [17]. The linear density filter shown in Equation 2-8 is used frequently in the comparisons presented in this paper. This form of the linear density filter accounts for potentially non-regular mesh densities [13].

$$\widetilde{x_e} = \mathcal{F}\left(x_j\right) = \frac{\sum_j w_{ij} V_{0j} x_j}{\sum_j w_{ij} V_{0j}} \qquad \text{Equation 2-8}$$

Because of the linear form of the filter, the objective and constraint sensitivities may be readily calculated as shown in Equation 2-9 and Equation 2-10. The summations over *j* in Equation 2-9 and Equation 2-10 are required because terms associated with the all the un-filtered optimization pseudo-densities within the user-defined filter radius will be present in the physical pseudo-densities.

$$\frac{\partial f}{\partial x_e} = \sum_j \left(\frac{\partial f}{\partial \tilde{x}_j}\right)\left(\frac{\partial \tilde{x}_j}{\partial x_e}\right) \qquad \text{Equation 2-9}$$

$$\frac{\partial g}{\partial \rho_e} = \sum_j \left(\frac{\partial g}{\partial \tilde{x}_j}\right)\left(\frac{\partial \tilde{x}_j}{\partial x_e}\right) \qquad \text{Equation 2-10}$$

The derivatives of the objective and constraint with respect to the filtered (physical) design variables may be evaluated as in Equation 1-10 and Equation 1-11, except that the un-filtered design variables are replaced by the filtered design variables, as shown in Equation 2-11 and Equation 2-12. The strain energies in Equation 2-11 are based on finite element analysis where the material properties calculated using the filtered design variables, which is written as $\widetilde{\boldsymbol{u}}_e$.

$$\frac{\partial C}{\partial \tilde{x}_e} = -2p\tilde{x}_e{}^{-1}\widetilde{\boldsymbol{u}}_e \qquad \text{Equation 2-11}$$

$$\frac{\partial g}{\partial \tilde{x}_e} = V_{0e}/V_f \qquad \text{Equation 2-12}$$

The derivative of the filtered pseudo-density with respect to the unfiltered pseudo-density, referred hereafter to as the filter chain rule term, can be written as a square matrix with the number of rows and columns equal to the number of elements in the design domain and given by Equation 2-13. Because the weighting matrix is nonzero only for elements that lie within the filtering radius, the filter chain rule is generally a sparse matrix. Furthermore, it can be seen in Equation 2-13 that the filter chain rule term matrix is a function only of the initial finite element model geometry and is, in general, asymmetric. The asymmetry of the filter chain rule term matrix is a consequence of elements along the periphery of the design domain having fewer neighboring elements than elements located near the center of the design domain.

$$\frac{\partial \tilde{x}_j}{\partial x_e} = \frac{w_{je}V_{0j}}{\sum_k w_{kj}V_{0k}} \qquad \text{Equation 2-13}$$

Because both the filter chain rule terms and weighting matrices are a function of discretization of the design domain only, these matrices need calculated only once prior to performing the topology optimization. It is worthwhile to note that the linear density filter behaves similarly to linking matrices that are often used in other aspects of structural optimization, such as sizing optimization problems. The linear density filter modifies the original topology optimization problem so that the stiffness of a given element is not only dependent on the un-filtered pseudo-density associated with that element but is also dependent on a weighted average of the un-filtered pseudo-densities of elements within a given filter radius.

An example of the effect of the linear density filtering is shown in Figure 2-2. The linear density filter effectively averages the elemental pseudo-density, blurring the boundaries between solid and void. The degree to which the boundaries are depends on the both the filtering radius, and the size of the structural member predicted in the optimization.

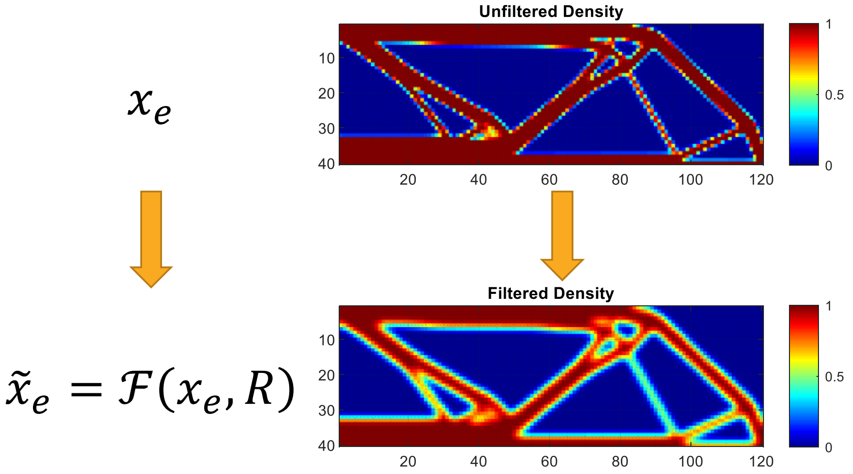


Figure 2-2. The linear density filter (R=0.12 inches) transforms the optimizer variables (top) into physical variables (bottom) that are used in finite element analysis and represent the design output by the topology optimization.

2.4.2. Other Density Filters

Other density filters based on the morphological operations used in image processing as well as filters based on Pythagorean means have been proposed by Sigmund [13] and Svanberg & Svärd [14], respectively. While none of these filters were used in example problems presented in this paper, these filters have nonetheless been proven to be more effective at addressing the checkerboarding and other numeric problems associated with topology optimization for certain problem types. A brief overview of the basic morphological filters proposed by Sigmund is provided below. The author was unsuccessful in implementing filters based on Pythagorean means, and so those filters are not discussed any further in this paper.

The morphological operations proposed by Sigmund include the erode and exponential dilate operations, which are defined by the filter operations shown in Equation 2-13 and Equation 2-14, respectively. The erode and dilate filters are continuous approximations of the min and max operators which depend on a user-defined parameter β. A β value of zero is equivalent to the linear density filter, and the filters approach the min or max operators as β increases to infinity.

$$\widetilde{x_\imath} = 1 - \frac{\log\left(\sum_j w_{ij} e^{\beta(1-x_j)}\right)}{\beta}$$

Equation 2-14

$$\widetilde{x_\imath} = \frac{\log\left(\sum_j w_{ij} e^{\beta x_j}\right)}{\beta}$$

Equation 2-15

The effect of applying the erode and dilate filters is illustrated in Figure 2-3 and Figure 2-4, respectively. As the parameter of β is increased, the filter begins to behave like a min or max operation for all elements that lie within the filter radius.

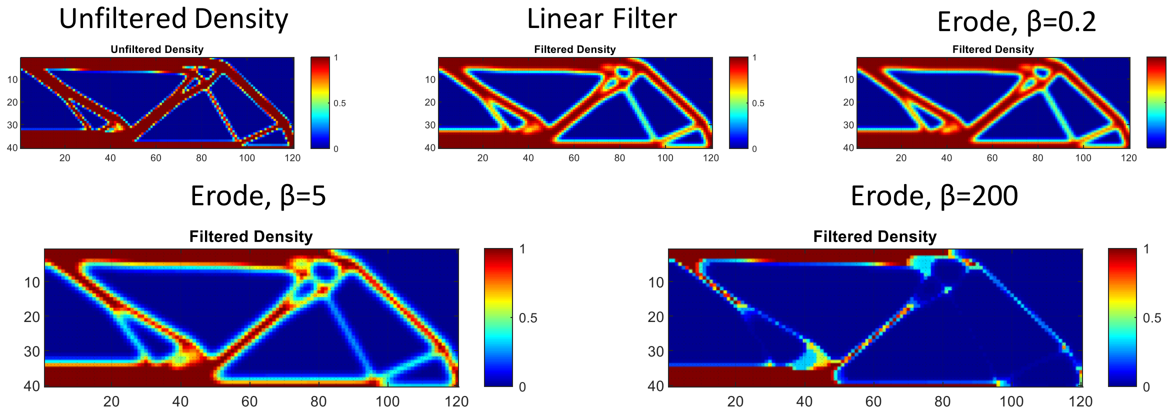


Figure 2-3. The erode filter behaves increasingly like the min operation as the filter parameter β is increased.

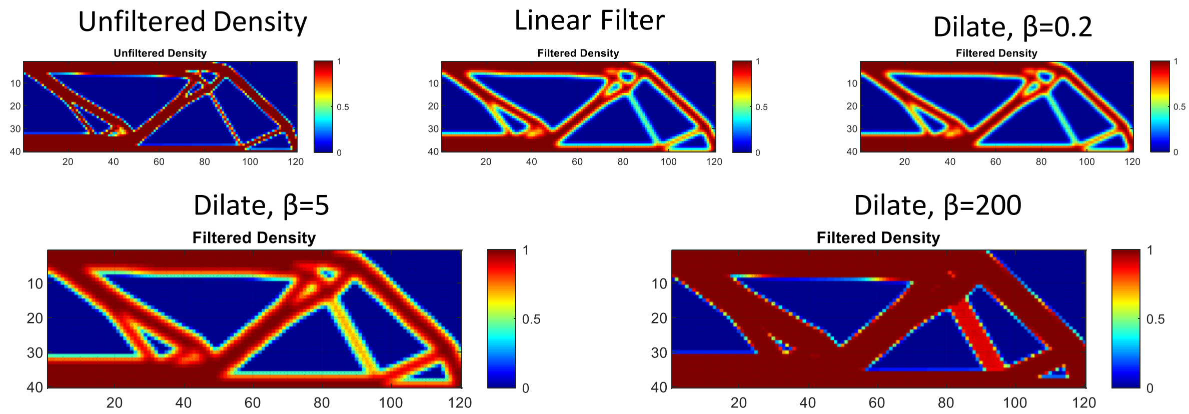


Figure 2-4. The dilate filter behaves increasingly like the max operation as the filter parameter β is increased.

Sigmund notes that the erode and dilate filters may be applied in sequences (e.g., erode followed by dilate, or dilate followed by erode) to replicate concepts from image processing. These image processing concepts can remove details smaller than the provided radius or fill in details smaller than the provided radius. The result of these operations is a topology that contains very few intermediate density elements (as is usually the case for a linear density filter) but is also effective at addressing the checkerboard problem. Unfortunately, the author was unable to implement these filters in any of the example problems. However, their potential for use as part of a larger topology density filtering package is acknowledged. Downsides of the morphological operations include added computational cost. In the case of a single erode or dilate operation, this extra cost stems from the fact that, unlike the linear density filter, the erode and dilate filter chain rule terms need to be re-calculated at each design iteration. Furthermore, using an erode or dilate operation in sequence with one another introduces an additional chain rule term in all sensitivity calculations that can quickly become a prohibitive computational challenge for larger scale problems.

3. Optimization Methods

Both optimality criteria and math programming methods remain popular options for solving the compliance minimization problem in both academic literature and commercial software implementations. Notable math programming methods used throughout this paper as well as in commercial software are mentioned in this section; however, details on math programming methods are omitted. In general, any math programming method capable of handling many design variables may be applied to the topology optimization problem.

A popular math programming choice in much of the literature is the method of moving asymptotes (MMA) or its successor, the globally convergent method of moving asymptotes (GCMMA) [20]. A MATLAB implementation of GCMMA is available, by request, from Professor Svanberg, and is used throughout this paper. Modified implementations of MMA or GCMMA are also used by many commercially available topology optimization software. For these reasons, GCMMA was used as the optimizer for all the example problems presented in this paper that were implemented in MATLAB using math programming. A summary of optimizers used for topology optimization in the commercial software evaluated in this paper include:

- Proprietary variations of MMA. For example, Simcenter Nastran SPOT optimizer, and ANSYS Sequential Convex Programming (SCP)
- Interior-point line search algorithms. For example, MSC Nastran IPOPT optimizer.
- Linear and/or quadratic programming methods. For example, Simcenter Nastran Siemens Design Optimization (SDO) and MSC Automated Design Synthesis (MSCADS).
- Optimality Criteria

3.1. Optimality Criteria (OC)

The optimality criteria (OC) method for the compliance minimization problem is based on satisfying the KKT gradient conditions directly for the Lagrangian function. The Lagrangian function and its gradient condition for the compliance minimization problem are shown in Equation 3-1 and Equation 3-2, respectively.

$$L(x_e,\lambda) = f + \lambda g = C(x_e) + \lambda g(x_e) \qquad \text{Equation 3-1}$$

$$\frac{\partial L}{\partial x_e} = \frac{\partial C}{\partial x_e} + \lambda \frac{\partial g}{\partial x_e} = 0 \qquad \text{Equation 3-2}$$

The Lagrange multiplier may be determined from the first-order gradient condition and simplifies as in Equation 3-3, where $\boldsymbol{v}_e$ is the strain energy density for a given element.

$$\lambda = \frac{-\frac{\partial C}{\partial x_e}}{\frac{\partial g}{\partial x_e}} = 2p\frac{\boldsymbol{u}_e}{x_e V_{0e}} = 2p\frac{\boldsymbol{u}_e}{V_e} = 2p\boldsymbol{v}_e \qquad \text{Equation 3-3}$$

Thus, the classical OC for compliance shows that the strain energy density of every element not at the upper and lower limits of the pseudo-density design variables should be constant. This assumption is related to the fully stressed design methodology outlined by Venkayya [21]. In order to determine how the pseudo-density design variables ought to be updated, a recursion relationship is usually performed in the spirit of the fully stressed design algorithm [22]. The recursion process begins by making an estimate of the Lagrange multiplier. Next, elements with higher strain energy densities have their pseudo-densities reduced in proportion to their strain energy density, and elements with lower strain energy have their pseudo-densities increased in proportion to their strain energy density. A power law is used to control that rate of change, as shown in Equation 3-4, which is the form of optimality used in Sigmund's 99-line MATLAB implementation of the compliance minimization problem [4]. Usually, the value of η is taken to be one half.

$$x_e^{(k+1)} = \left(\frac{-\frac{\partial C}{\partial x_e}}{\lambda \frac{\partial g}{\partial x_e}} \right)^{\eta} x_e^{(k)} = \left(\frac{2p u_e^{(k)}}{\lambda x_e^{(k)} V_{0e}} \right)^{\eta} x_e^{(k)} \qquad \text{Equation 3-4}$$

Two modifications to the original OC recursion relationship are proposed in the following subsections. The first modification attempts to show what the appropriate power, η, in the OC recursion relationship should be. The second modification accounts for the inclusion of a linear density filter in the OC recursion relationship derivation.

3.1.1. Modification to OC Recursion Power Law

A more rigorous approach may be used to derive the OC recursion relationship wherein the damping parameter, η, may be calculated. This is accomplished through the use an appropriate intermediate variable, y_e, using a reciprocal relationship, as shown in Equation 3-5, where p is the penalization value from the SIMP method.

$$y_e = x_e^{-p} \rightarrow x_e = y_e^{-1/p} \qquad \text{Equation 3-5}$$

This choice of intermediate variable approximation is based on the observation that scaling the global stiffness matrix by the power law results in scaling the displacement vector by the inverse of the power law. That is, suppose the structural volume is scaled by a factor s, such that $V = s^p V_0$. The associated stiffness matrix would then scale as $\boldsymbol{K} = s^p \boldsymbol{K_0}$. As a result, the displacement vector, shown in Equation 3-6, is made to be linear with respect to a reciprocal variable of the form used in Equation 3-5.

$$D = \boldsymbol{K}^{-1} R = (s^{\boldsymbol{p}} \boldsymbol{K_0})^{-1} R = s^{-p} \boldsymbol{K}_0{}^{-1} R = s^{-p} D_0 \qquad \text{Equation 3-6}$$

Using the intermediate variable defined in Equation 3-5, the compliance at a new intermediate variable iteration may be approximated using a first order expansion about the current intermediate variable, as shown in Equation 3-7.

$$\hat{C}^{(k+1)} = {C_o}^{(k)} + \left(\frac{\partial C}{\partial y_e}\right)^{(k)} \left({y_e}^{(k+1)} - {y_e}^{(k)}\right) \qquad \text{Equation 3-7}$$

where the sensitivity of the compliance with respect to the intermediate design variable may be evaluated using the chain rule, as shown in Equation 3-8.

$$\left(\frac{\partial C}{\partial y_e}\right)^{(k)} = \left(\frac{\partial C}{\partial x_e}\frac{\partial x_e}{\partial y_e}\right)^{(k)} = \left(\left(-2p{x_e}^{-1}\boldsymbol{u}_e\right)\left(\frac{-1}{p}{x_e}^{p+1}\right)\right)^{(k)} \qquad \text{Equation 3-8}$$

$$= \left(2u_e{x_e}^{p}\right)^{(k)}$$

Next, the Lagrange function for the compliance minimization problem is constructed using the approximate form for compliance, shown in Equation 3-9. The first-order gradient condition may also be evaluated, as shown in Equation 3-10. Implicit in the linear approximation of compliance in the intermediate design variable space is the idea that the design is “frozen” in the intermediate design space at iteration k. Therefore, the derivatives of ${C_o}^{(k)}$ and ${y_e}^{(k)}$ with respect to x_e are each zero.

$$\hat{L}(x_e, \lambda) = \hat{C}\left(y_e(x_e)\right) + \lambda g(x_e) \qquad \text{Equation 3-9}$$

$$\frac{\partial \hat{L}}{\partial x_e} = \frac{\partial \hat{C}}{\partial x_e} + \lambda\frac{\partial g}{\partial x_e} = \left(\frac{\partial C}{\partial y_e}\right)^{(k)} \left(\frac{\partial y_e}{\partial x_e}\right)^{(k+1)} + \lambda\frac{\partial g}{\partial x_e} \qquad \text{Equation 3-10}$$

$$= \left(2u_e{x_e}^{p}\right)^{(k)}\left(-p{x_e}^{-p-1}\right)^{(k+1)} + \lambda V_{0e} = 0$$

Finally, the first-order gradient condition shown in Equation 3-10 may be rearranged to form the recursion relationship, as shown in Equation 3-11. Noting the similarities between Equation 3-4 and Equation 3-11, for $p = 1$, the power law associated with the recursion relationship is exactly ½, as was suggested by Sigmund (2001). However, for $p > 1$, the recursion power should take a value equal to $(p + 1)^{-1}$.

$${x_e}^{(k+1)} = \left(\frac{2p{u_e}^{(k)}}{\lambda {x_e}^{(k)} V_{0e}}\right)^{\frac{1}{p+1}} {x_e}^{(k)} \qquad \text{Equation 3-11}$$

Numeric experiments using the proposed recursion power law with $\eta = (p + 1)^{-1}$ have been carried out and compared to the case where $\eta = \frac{1}{2}$ in a similar fashion to the examples shown in Section 5.4 for the case where no filtering was applied. The results from this evaluation are shown in Figure 3-1 for several different SIMP penalty values. The use of the $\eta = (p + 1)^{-1}$

recursion power law resulted in slightly more required iterations to achieve convergence based on the Lagrangian gradient condition for all penalty values assessed. However, the impact on the final objective function values were only marginally affected, and the per iteration solution time was completely unaffected.. For these reasons, all MATLAB topology optimization results shown in 5.4 make use of the $\eta = (p+1)^{-1}$ recursion power law.

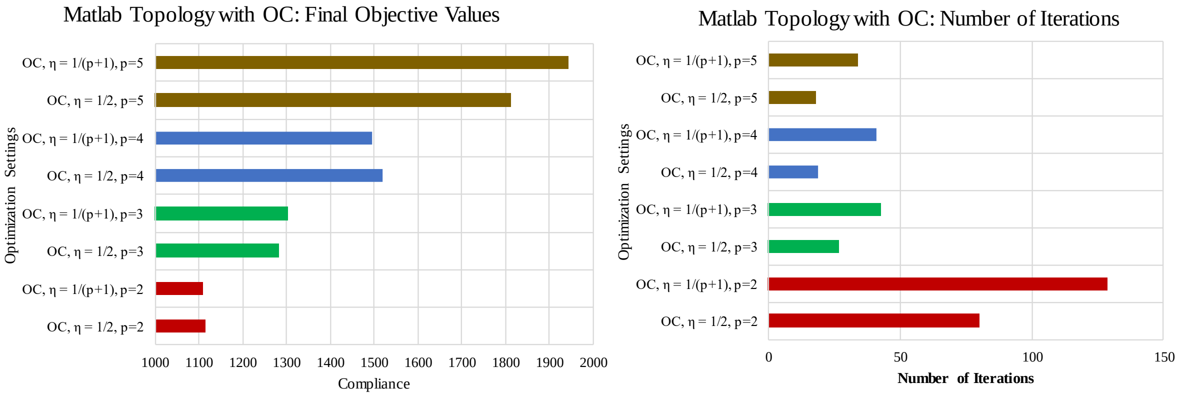


Figure 3-1. The final objective values and the number of iterations required to convergence based on the Lagrangian gradient condition for MATLAB topology using OC recursion using $\eta = \frac{1}{2}$ and $\eta = (p+1)^{-1}$ power laws for various SIMP penalty factors (density and sensitivity filtering was not applied).

3.1.2. Modification to OC Recursion for the Linear Density Filter

A similar derivation to the one shown in Section 3.1.1 may be applied by instead considering intermediate variables based on the filtered pseudo-density design variables, as shown in Equation 3-12.

$$\tilde{y}_e = \tilde{x}_e^{-p} \rightarrow \tilde{x}_e = \tilde{y}_e^{-1/p} \qquad \text{Equation 3-12}$$

Following nearly identical steps as in Section 3.1.1, the approximate form of compliance may be derived. The first-order gradient condition in the presence of a linear density filter is shown in Equation 3-13. Equation 3-13 resembles Equation 3-10, except that compliance and constraint sensitivities must include the chain rule terms and summations discussed in 2.4.1.

$$\begin{aligned} \frac{\partial \hat{L}}{\partial x_e} = \frac{\partial \hat{C}}{\partial x_e} + \lambda \frac{\partial g}{\partial x_e} \\ = \sum_{j \in N_e} \left(\frac{\partial \hat{C}}{\partial \tilde{x}_j}\right)\left(\frac{\partial \tilde{x}_j}{\partial x_e}\right) + \lambda \sum_{j \in N_e} \left(\frac{\partial g}{\partial \tilde{x}_j}\right)\left(\frac{\partial \tilde{x}_j}{\partial x_e}\right) \\ = 0 \end{aligned} \qquad \text{Equation 3-13}$$

The expression in Equation 3-13 can simplify even further, as shown in Equation 3-14. This form of the first-order gradient term is a function of only the finite element geometry, the SIMP penalty value, the un-filtered design variables, and the element strain energies (evaluated using the filtered/physical design variables).

$$\frac{\partial \hat{L}}{\partial x_e} = \sum_j \left((-2p{x_e}^p \tilde{u}_e)^{(k)} ({x_e}^{-p-1})^{(k+1)} \right) \left(\frac{\partial \tilde{x}_j}{\partial x_e} \right) + \lambda \textstyle\sum_j (V_{0j}) \left(\frac{\partial \tilde{x}_j}{\partial x_e} \right) = 0$$

Equation 3-14

Unlike the recursion relationship derived in Section 3.1.1, the first-order gradient condition in Equation 3-14 cannot immediately be used to determine the next iteration of the design variables. This is because each element of the Lagrangian gradient is dependent on multiple design variables. Therefore, the problem is re-arranged into a form that may easily be solved using standard matrix operations, as shown in Equation 3-15. In Equation 3-15, singularities in the matrix decomposition process will occur as the strain energy in an element approaches a small number. This situation often occurs for elements in regions that the optimizer has determined should be a void. Singularities are easily remedied by enforcing an arbitrarily small, yet greater than zero, minimum value for the strain energy of every element in the design domain. Through the numerical examples performed in this paper, it was found that setting the minimum strain energy to be three orders of magnitude lower than the minimum pseudo-density design variable, x_{min}, is adequate to eliminate MATLAB warning messages associated with singularities in the solution for z_e.

$$\frac{\partial \hat{L}}{\partial x_e} = \left[{\tilde{u}_e}^{(k)} \frac{\partial \tilde{x}_j}{\partial x_e} \right] \left\{ {x_e}^{p(k)} \Big/ \lambda {x_e}^{p+1(k+1)} \right\} = \frac{1}{2p} \left\{ \sum_j (V_{0j}) \left(\frac{\partial \tilde{x}_j}{\partial x_e} \right) \right\}$$

Equation 3-15

For brevity, the solution to Equation 3-15 is defined as z_e, which may be written as in Equation 3-16. The values for z_e correspond to a ratio of the design variables between two iterations, scaled by the Lagrange multiplier. The solution to this system of equations presented in Equation 3-15 is typically about as expensive as the finite element solution. Therefore, calculation of z_e should be limited as much as possible. Because z_e includes the Lagrange multiplier, this solution needs to be generated once per design iteration.

$$z_e = \left\{ {x_e}^{p(k)} \Big/ \lambda {x_e}^{p+1(k+1)} \right\} = \left[{\tilde{u}_e}^{(k)} \frac{\partial \tilde{x}_j}{\partial x_e} \right]^{-1} \left(\frac{1}{2p} \left\{ \sum_j \left(V_{0j} \right) \left(\frac{\partial \tilde{x}_j}{\partial x_e} \right) \right\} \right) \qquad \text{Equation 3-16}$$

Finally, the OC recursion relationship in the presence of a linear density filter may be written as in Equation 3-17. The procedure for determining the next design point is identical to the original OC derivation. Namely, a Lagrange multiplier is first estimated, and then recursively updated with, for example, the bisection method, until convergence for the Lagrange multiplier is achieved. Noting the similarities between Equation 3-11 and Equation 3-17, the z_e term may be regarded as a sort of filtered strain energy density term.

$${x_e}^{(k+1)} = {x_e}^{(k)} \left(\lambda {x_e}^{(k)} z_e \right)^{-1/(p+1)} \qquad \text{Equation 3-17}$$

Example problems solved using the OC recursion relationship derived in this section are shown in Section 5.4. In general, topologies and convergence histories resulting from the OC recursion relationship derived in this section tend to resemble results from GCMMA. The most apparent drawback to the approach outlined in this section is the requirement for an additional matrix decomposition. However, it is shown in Section 5.4 that despite this extra computational burden, solution times for the OC recursion relationship with the linear density filter and GCMMA are comparable.

4. Commercial Software Implementations

The Messerschmitt-Bölkow-Blohm (MBB) beam is a classic problem used throughout topology optimization literature. In this problem, a simply supported beam is representative of a support beam from a civil aircraft produced by Messerschmitt-Bölkow-Blohm GmbH. The beam supports a vertical load at its mid-span and is optimized for compliance subject to a restriction on the available material [23]. Further details on the modeling approach for the MBB beam is provided in Section 5.1. Four popular commercial software packages were used to implement the 2D MBB compliance minimization problem throughout this paper. These packages include: ANSYS Workbench 2019 R1, Simulia TOSCA Structure 2019 with Abaqus 3DEXPERIENCE R2019x, MSC Nastran 2018, and Simcenter Nastran (formerly known as NX Nastran) 2019.2. In addition, Simcenter Nastran 2019.2 and TOSCA Structure 2019 were also used to solve the 3D

MBB compliance minimization problem. Another popular topology optimization software package, Altair's Optistruct, was not evaluated in this paper.

Each commercial software package utilizes one or more MP or OC methods to solve the topology optimization problem. Each software also offers at least one method to control and eliminate the checkerboard effect. However, exact details on the algorithms used to perform optimization as well as address the checkerboarding and mesh dependence issues are considered proprietary, and therefore are not covered in detail in the software companion documentation. Nevertheless, certain aspects of how checkerboard control is implemented (e.g., whether it is a binary toggle switch, always on by default, or has some user-defined parameter related to the filter radius) is documented in this section. Table 4-1 summarizes the key aspects of each topology optimization software package. The features of each piece of software are discussed in further detail in the following subsections.

Table 4-1. Key features of four popular commercially available topology optimization software

Feature	ANSYS Workbench	TOSCA Structure	MSC Nastran	Simcenter Nastran
Finite Element Solver(s)	-ANSYS	-Abaqus/Standard -Abaqus/Explicit -MSC Nastran -ANSYS	-MSC Nastran	-Simcenter Nastran
Optimization Methods	-SCP (MMA-like) -OC	-General sensitivity -Controller-based (OC-like)	-IPOPT (line-search) -MSCADS (linear/quadratic programming)	-SPOT (MMA-like) -GCMMA -SDO (linear programming)
Checkerboard Control Options	-Always on	-Always on, may toggle between "Standard" filter and "Low" filter	-Toggle control types (see below)	-Toggle On/Off
Checkerboard Control Type	- Undocumented	-Undocumented	-Filtering algorithm -Density constraint	-Localized density filter
Initial Density	-User specified constant	-Arbitrary user-specified densities	-User specified constant	-Full density (always)

4.1. ANSYS Workbench

ANSYS Workbench topology optimization is developed and maintained by ANSYS Inc. The ANSY implementation of topology optimization may only be performed in the ANSYS workbench environment. Unlike other topology optimization software, ANSYS workbench requires a geometric CAD representation of the design domain in order to proceed. RP approaches have been implemented in ANSYS for many years. Beginning in ANSYS 2019 R3, a level-set implementation of the topology optimization problem (an alternative to RP approaches) was also implemented. This approach was not evaluated in this paper.

ANSYS offers an MMA-like optimizer for general topology optimization problems, and an implementation of OC for the compliance minimization problem. For RP methods, a filtering algorithm is on by default, with no ability to toggle the control or adjust filter parameters. Presumably, the software contains logic to control the size of the filter based on the mesh density. However, this is speculation; details on the exact implementation of the filtering algorithm are considered proprietary and are not provided in the ANSYS documentation. Additional constraints are available to limit, for example, the minimum member size obtained by the topology optimization solution.

In ANSYS, the initial topology may be initialized to a constant user-specified density over the entire design domain. By default, the initial density is automatically selected to be equal to the volume constraint for volume-constrained problems. Convergence in ANSYS is satisfied when changes in the objective and constraint functions are within a user-defined tolerance, or when a maximum number of iterations has occurred.

4.2. TOSCA Structure (Abaqus/Standard)

Unlike the other software evaluated in this paper, TOSCA Structure (along with its fluid-based optimization software, TOSCA Fluid), are not directly tied to any given finite element software. TOSCA Structure was originally developed by FE-DESIGN GmbH and was intended as a general optimization framework with the ability to couple with multiple physics simulation software. TOSCA Structure can address sizing, shape, and topology optimization problems. As a result, TOSCA Structure can use multiple finite element solvers, including Abaqus, ANSYS, and MSC Nastran. FE-DESIGN GmbH was acquired by Dassault Systemes in 2013, and later integrated into the Simulia and Abaqus analysis packages. TOSCA Structure is now coupled with Abaqus/CAE and is used by all Abaqus optimization tasks. However, TOSCA Structure may also be used in a stand-alone configuration and coupled with finite element solvers besides Abaqus.

TOSCA Structure utilizes a “general sensitivity” optimization algorithm for most topology optimization problems. By default, the “general sensitivity” algorithm utilizes an implementation of the MMA for solving topology optimization problems. TOSCA Structure also has the option of using a Convex Separable Approximation (CSA) for problems where the objective or constraints change rapidly with changes in the design variables. For the “general sensitivity” algorithm, convergence is satisfied when one or both changes in objective and constraint functions are within a user-defined tolerance, or when a maximum number of iterations has occurred

For the compliance minimization problem subject to a volume constraint, a “controller-based” algorithm is available in TOSCA Structure. The “controller-based” algorithm is unique among all commercial software packages evaluated in this paper in that it does not require sensitivity information of the objective or constraint functions [26]. Instead, the “controller-based” program uses a controller to update the design; the only requirement for the “controller-based” algorithm is that nodal von Mises stresses from the design domain be accurate and available. Convergence of the compliance minimization problem with no intermediate pseudo-densities occurs within 15 iterations, and convergence is *not* based on changes in the objective or constraint functions. The “controller-based” algorithm proceeds in a fashion more like a nonlinear static solution than it is like the various RP approaches discussed for other software discussed in this paper.

In TOSCA Structure, the initial topology may be initialized to arbitrary user-specified densities, which is a feature that is either not present or difficult to implement in all other commercial software evaluated in this paper. By default, the initial density is selected to be equal to the volume constraint for volume-constrained problems, and equal to 0.50 for problems that do not contain a volume constraint. A filtering algorithm is used by the “general sensitivity” algorithm, and is always on. TOSCA Structure offers an alternative to the default “standard” filtering algorithm in the form of the “low” filter, which is intended to be used with particularly coarse mesh densities [26]. For the “general sensitivity” algorithm, the default value for the filter radius is 1.3 times the average element edge length in the design domain. This edge length corresponds to including roughly all attached elements in the filtering algorithm. The default filter radius value may be manually overridden in the TOSCA Structure GUI or in the TOSCA Structure parameter (*.par) file, but there are no options for controlling the TOSCA Structure filter from within the Abaqus/CAE GUI. For the “controller-based” algorithm, a means to control the checkerboarding may be implicit in within the controller. However, this is speculation; details on the exact implementation of the filtering algorithm as well as details on the controller-based algorithm are considered proprietary and are not provided in the Simulia documentation. Additional constraints are available to limit, for example, the minimum member size obtained by the topology optimization solution.

4.3. MSC Nastran

MSC Nastran topology optimization (SOL 200) offers a wide variety of optimization algorithms and parameters. Using the default settings, the MSC Nastran interior-point optimizer (IPOPT) is the default for most topology optimization problems and contains only a limited set of parameters that may be adjusted. For non-topology optimization problems and topology optimization problems with many constraints, MSC Automated Design System (MSCADS) is used instead. MSCADS offers significant control over the specifics of the optimization algorithms. However, because it is not recommended for most topology optimization problems, MSCADS was not evaluated further in this paper.

The means by which MSC Nastran addresses the checkerboarding and mesh dependence problems is selected by the TCHECK bulk data entry. The available choices for TCHECK are the "filtering algorithm" (TCHECK value of 1), the "density constraints" (TCHECK value of 2), "no control" (TCHECK value of 0), or automatic selection based on performance (TCHECK value of –1, the default). It is unclear how the default "automatic selection" logic selects the appropriate filtering type. When "no control" is specified, the effect of checkerboarding is clearly visible in the topology optimization results. It is speculated that the "filtering algorithm" corresponds to an implementation of a sensitivity filter, whereas the "density constraint" corresponds to an implementation of a linear density filter.

When either the "filtering algorithm" or "density constraint" is selected, MSC Nastran references the bulk data entry TDMIN. TDMIN may be thought of as the diameter of the filter (twice the filter radius). In numeric experiments, it was concluded that if TDMIN is either unspecified or is smaller than the mesh density, MSC Nastran will adjust TDMIN to be equal to the mesh density. In this way, at least one element on any side of a given element is included in the filtering algorithm. Specifying a TDMIN value greater than the mesh density adjusts the filter radius accordingly. Unlike every other software package evaluated, MSC Nastran does not include separate manufacturing constraints for the minimum member size. Instead, MSC Nastran relies on the value specified for TDMIN to indirectly introduce a mesh-independent length scale into the problem, thereby loosely enforcing a minimum member size.

In MSC Nastran, the initial topology may be initialized to a user-specified constant density; by default, the initial density is automatically selected to be equal to the volume constraint for volume-constrained problems. Convergence in MSC Nastran is satisfied when changes in the objective and constraint functions are within a user-defined tolerance, or when a maximum number of iterations has occurred. In addition, MSC Nastran uses undocumented logic to determine whether a design is “sufficiently black and white,” meaning that the software has determined that there is a sufficiently low number of elements that contain intermediate pseudo-densities. If the “sufficiently black and white” logic is triggered, the optimization algorithm automatically adjusts the objective and constraint tolerances to a value of 0.005. In numeric experiments carried out, this adjustment has the apparent effect of prematurely terminating the optimization. It is unclear whether this logic may be disabled or overridden.

4.4. Simcenter Nastran

Simcenter Nastran (formerly known as Siemens NX Nastran) topology optimization (SOL 200) is a relatively new capability that was introduced by Siemens in 2018. Prior to 2013, NX Nastran utilized aspects of TOSCA Structure to perform topology optimization using SOL 200. After the acquisition of FE-DESIGN GmbH by Simulia in 2013, Siemens began in-house development of topology optimization capabilities.

By default, Simcenter Nastran topology optimization utilizes an MMA-like optimizer known as SPOT for problems with a relatively small number of constraints. When many constraints are present, a linear programming-based optimizer known as Siemens Design Optimizer (SDO) is automatically selected instead. The automatic optimizer choices may be overridden in SOL 200 by modifying the Nastran system cell 425. While not advertised in the Simcenter Nastran documentation, assigning a system cell 425 value of 2 forces SOL 200 to use GCMMA.

In Simcenter Nastran, the means to address the checkerboarding problem is controlled via a the CHBC field on the DCON bulk data entry. By default, CHBC is turned on (value greater than 0) at the tenth iteration. The iteration at which CHBC is enabled may be adjusted by the user via the BDMNCON parameter. CHBC may be turned off for all iterations by entering a negative value in the CHBC field of the DCON bulk data entry. The exact algorithm used by Simcenter Nastran checkerboard control is undocumented. However, numeric experiments have revealed that the control algorithm appears to be heuristic and possible enforced only at a local level. That is, if the software detects a checkerboard pattern, it addresses the checkerboard pattern for region where the pattern is detected only. Because of this, the algorithm is incapable of introducing a mesh-dependent minimum length scale in the same way that other commercial software does. Simcenter Nastran includes a separate minimum member size constraint, which is intended to introduce a mesh-dependent length scale. In practice, the Simcenter Nastran checkerboard control is effective at eliminating checkerboard patterns, but often results in “jumpy” behavior of the constraint or objective functions. Because the filtering algorithm is heuristic, it is speculated that sensitivity information fed to the optimizer is inaccurate and thus the optimizer guides the solution in inaccurate directions, resulting in the “jumpy” convergence behavior.

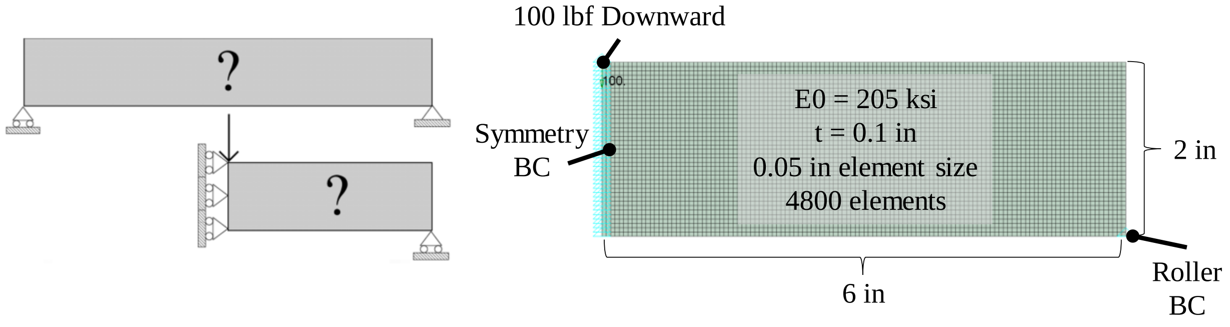
In Simcenter Nastran, the initial topology is always set to be “full” density (all design variables equal to one). For volume-constrained problems, this means that the design will always start out as infeasible. It is well understood that solutions from topology optimization are sensitive to the initial density assumption. Therefore, the lack of this capability is considered a shortcoming of Simcenter Nastran topology optimization. It is anticipated that the ability to specify an initial design density will be introduced in future Simcenter Nastran releases. However, in its current form, Simcenter Nastran appears to be the only popular commercial software without this capability. Convergence in Simcenter Nastran is satisfied when changes in the objective and constraint functions are within a user-defined tolerance, or when a maximum number of iterations has occurred.

5. Results and Discussion

The final designs (topologies and objectives and constraint values) as well as solution times for MATLAB-based and commercial software implementations of the compliance minimization problem are presented in this section. Section 5.1 describes the compliance minimization problems referenced throughout this section. A brief overview of the MATLAB implementations of the topology optimization problem is discussed in Section 5.2. In Section 5.3, the effect of filter weighting types, filter types (sensitivity or linear density), and filter radii is discussed for a MATLAB implementation of the topology optimization problem using GCMMA as the optimizer. In Section 5.4, topology optimization results are shown using the OC recursion relationships discussed in Section 3.1 as the optimizer. Section 5.5 shows results for the 2D MBB compliance minimization problem implemented in the four commercial software packages described in Table 4-1. Finally, Section 5.5.1 shows the results for the 3D MBB compliance problem implemented in Simcenter Nastran and TOSCA Structure commercial software packages.

5.1. Compliance Minimization Example Problems

Throughout this section, the compliance minimization problem is applied to a specific set of finite element geometry and load representative of the well-known MBB beam problem [23]. For all examples shown using 2D elements, a consistent 120 by 40 element mesh is used with regularly sized (0.05 inch) quadrilateral elements, unless otherwise indicated. The nominal material modulus is assumed to be 205 ksi, and the thickness is assumed to be 0.1 inches. A single point load of 100 lbf is applied in the vertical downward direction at the upper left most node. Symmetry boundary conditions are enforced on the left-hand side (the plane of symmetry), and a roller boundary condition is enforced on the bottom right corner, as shown in Figure 5-1. When implemented in commercial finite element software, the out-of-plane translation of the beam was also restrained. For all 2D MBB results presented in this paper, the SIMP penalty is assumed to be a constant value of 3. In all 2D MBB example results, the volume constraint is selected to be 40% of the volume of the design domain.



2D Half MBB Problem Setup

2D Half MBB Finite Element Model

Figure 5-1. Problem set-up for the 2D MBB compliance minimization problem referenced throughout this paper.

Because of the regular mesh size of the 2D MBB example problem, the size of the filter radius is easily visualized for all elements in the design domain. Throughout this paper, different filter sizes are used to apply the sensitivity and linear density filters. Four different filter sizes are shown in Figure 5-2. The smallest possible filter captures, at a minimum, the four elements that border a given interior element. The filter radius may be expanded to include elements connected to the corners of a given element and when further expanded includes an increasingly large number of elements in the filter operation.

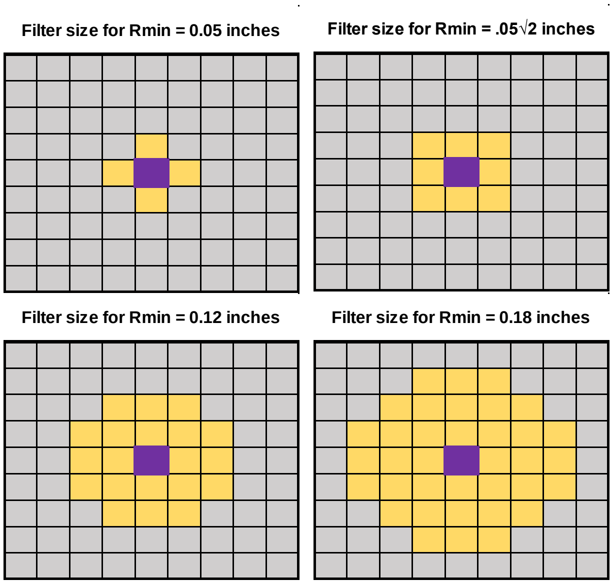

Filter size for Rmin = 0.05 inches
Filter size for Rmin = .05√2 inches
Filter size for Rmin = 0.12 inches
Filter size for Rmin = 0.18 inches

Figure 5-2. Four different filter radii (filter sizes) used throughout this paper. Individual squares represent a region of the design domain finite element model. The purple element represents the element for which the filtering operation is performed. Elements highlighted yellow are included in the purple element's filtering operation.

The problem setup for the 3D MBB compliance minimization problem solved later in this paper is shown in Figure 5-3. While the element type used to represent the 3D MBB beam vary between examples, a consistent 0.05-inch mesh density is maintained. The thickness of the 3D beam was set to 1 inch, and the total downward load, which uniformly acts on all elements at the upper center of the beam, is 210 lbf. For all 3D MBB results presented in this paper, the SIMP penalty is assumed to be a constant value of 3. In all 3D MBB example results, the volume constraint is selected to be 20% of the volume of the design domain.

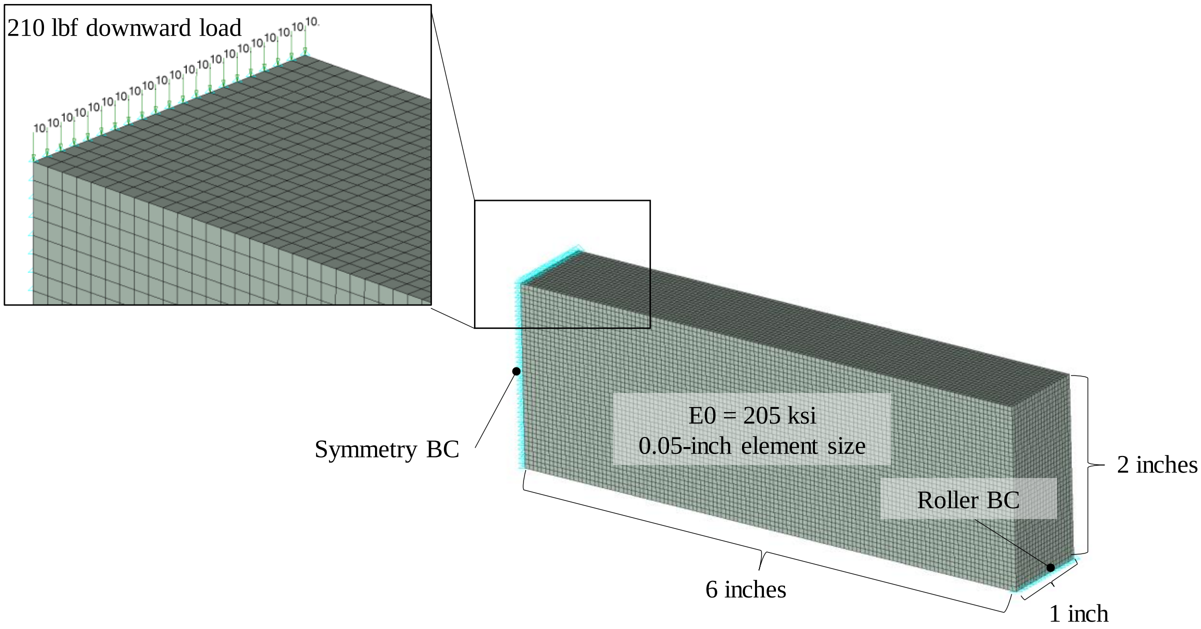


Figure 5-3. Problem set-up for the 3D MBB problem referenced in Section 5.6.

5.2. MATLAB Topology Implementation Overview

A MATLAB implementation of the 2D compliance minimization was used to calculate the optimal topologies for differing filter parameters. Figure 5-4 shows a flow-chart of the operations and convergence logic used for each MATLAB implementation of the topology optimization problem. Within each following subsection, care was taken to ensure a consistent converge parameters ($TolF$, $TolG$, $TolX$, $TolOpt$, and $MaxIter$) were used, set to 1.0E-4, 1.0E-6, 1.0E-6, 1.0E-4, and 1.0E+4, respectively, for all MATLAB topology optimization results presented in this paper. Additionally, move limits of 0.20 for the design variables (enforced as an additive, rather than multiplicative value of the current design variables) were used for all MATLAB results presented in this paper. For the 2D MBB compliance minimization problem, the density is initialized as a uniform constant over the design domain such that the volume constraint is equal to zero. For all MATLAB implementations, finite element analysis is performed using the FEA toolbox developed by Dr. Canfield of Virginia Tech [24].

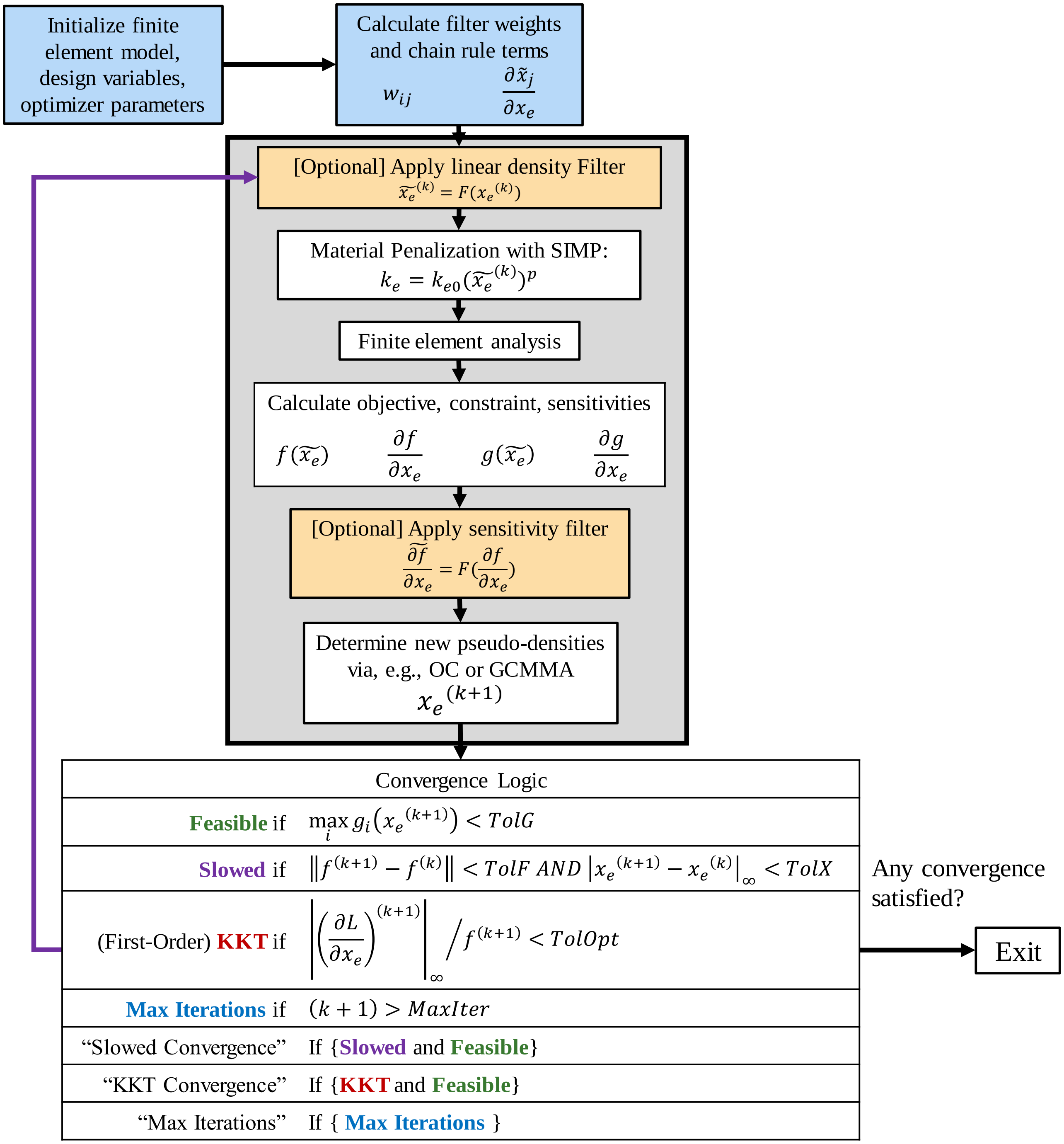

Initialize finite element model, design variables, optimizer parameters
Calculate filter weights and chain rule terms
w_{ij}
$\frac{\partial \tilde{x}_j}{\partial x_e}$
[Optional] Apply linear density Filter
$\widetilde{x_e}^{(k)} = F({x_e}^{(k)})$
Material Penalization with SIMP:
$k_e = k_{e0}(\widetilde{x_e}^{(k)})^p$
Finite element analysis
Calculate objective, constraint, sensitivities
$f(\widetilde{x_e})$
$\frac{\partial f}{\partial x_e}$
$g(\widetilde{x_e})$
$\frac{\partial g}{\partial x_e}$
[Optional] Apply sensitivity filter
$\widetilde{\frac{\partial f}{\partial x_e}} = F(\frac{\partial f}{\partial x_e})$
Determine new pseudo-densities via, e.g., OC or GCMMA
${x_e}^{(k+1)}$
Convergence Logic
Feasible if $\max_i g_i({x_e}^{(k+1)}) < TolG$
Slowed if $\|f^{(k+1)} - f^{(k)}\| < TolF\ AND\ |{x_e}^{(k+1)} - {x_e}^{(k)}|_\infty < TolX$
(First-Order) KKT if $\left| \left(\frac{\partial L}{\partial x_e}\right)^{(k+1)} \right|_\infty / f^{(k+1)} < TolOpt$
Max Iterations if $(k+1) > MaxIter$
"Slowed Convergence" If {Slowed and Feasible}
"KKT Convergence" If {KKT and Feasible}
"Max Iterations" If { Max Iterations }
Any convergence satisfied?
Exit

Figure 5-4. Flow-chart of operations used to perform topology optimization in MATLAB environment along with convergence logic. Note that the linear density and sensitivity filters are never applied simultaneously.

5.3. MATLAB Topology Optimization using GCMMA

Because each of the following subsections focuses on the impact of various filter parameters used in the topology optimization, the optimized topology for the 2D MBB example problem without any filters are shown first in Figure 5-5. The results in Figure 5-5 make use of the

GCMMA optimizer using the SIMP approach with a penalty parameter of 3. The effect of checkerboarding is clearly visible in the final topology of Figure 5-5. While some intermediate density elements are present in Figure 5-5, these intermediate densities may be removed by either increasing the SIMP penalty parameter, decreasing the optimization convergence parameters, or a combination of the two. Finally, it must be noted that this is a very specific numeric example which has been run with one specific optimization method (GCMMA). It is known that topologies and topology optimization iteration histories are sensitive to the specific optimizer algorithm and optimizer settings employed. Therefore, the conclusions presented in this section example must be viewed with the understanding that they are applicable to a very specific numeric example, and the general behavior of filter settings may differ from the conclusions presented herein.

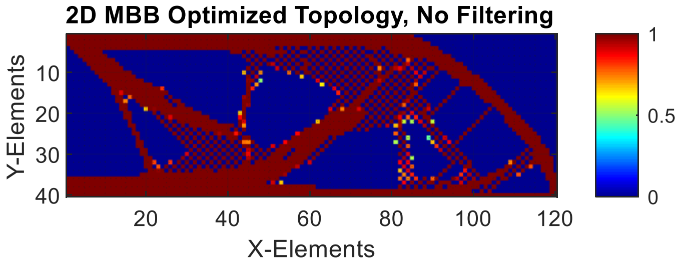


Figure 5-5. Contour showcasing the final element pseudo-density design variable values from topology optimizing using GCMMA, where no sensitivity or density filtering has been applied (SIMP p=3).

Figure 5-6 shows comparisons of constant and conic weighting matrices for various filter radii using a sensitivity filter. Figure 5-7 shows comparisons of constant and linear weighting matrices for various filter radii using a linear density filter. Figure 5-8 shows comparisons of the final objective value from the topology optimization for each of the filter settings investigated in this section. Figure 5-9 shows comparisons of both the total number of iterations required for convergence and the average user time spent on each iteration. This numeric example was performed on a desktop PC with 32 GB of RAM available, and multiple solutions were run concurrently. Admittedly, the iteration times may have been altered because multiple solutions were run at once. Nevertheless, the average per iteration solution time provide a rough order of magnitude difference in the computational cost between each filtering method assessed. Finally, Figure 5-10 and Figure 5-11 show the first-order gradient condition (the infinity norm of the

Lagrangian gradient, normalized by the objective) iteration history for the case of sensitivity filters and linear density filters, respectively.

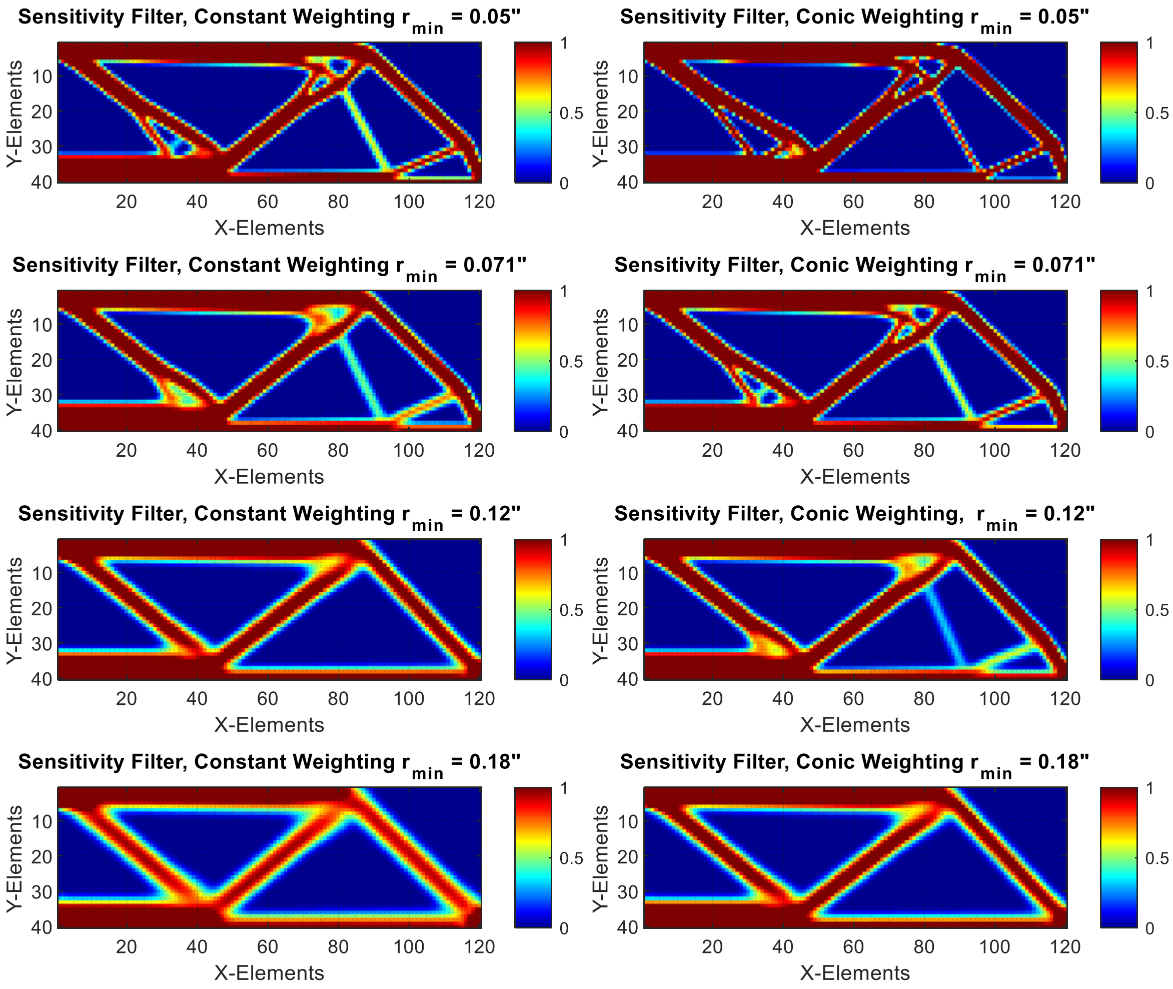
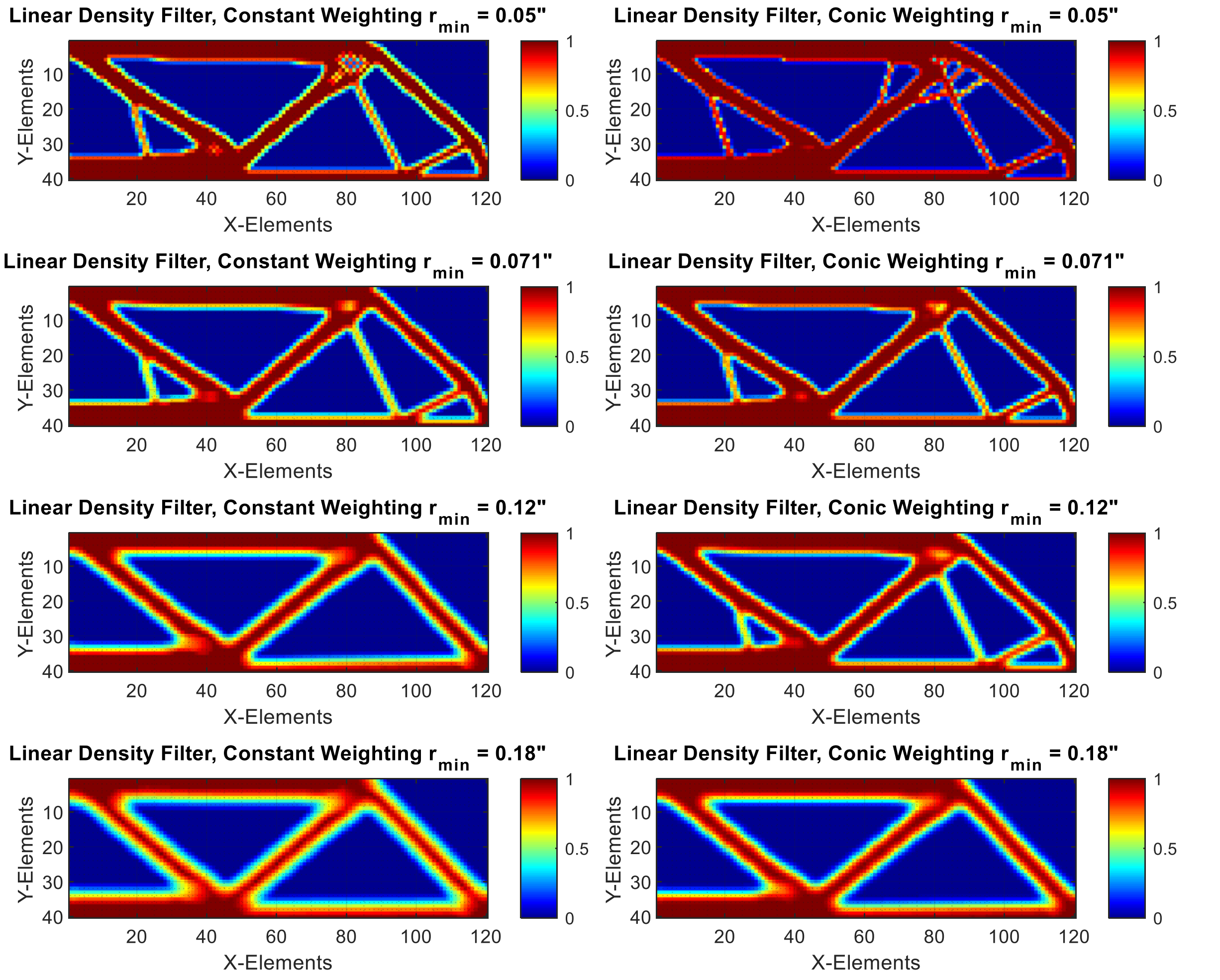


Figure 5-6. Final topologies for various filter sizes using a sensitivity filter. (Left) Constant sensitivity filter weights, (right) conic/linear sensitivity filter weights.



Linear Density Filter, Constant Weighting r_{min} = 0.05"

Linear Density Filter, Conic Weighting r_{min} = 0.05"

Linear Density Filter, Constant Weighting r_{min} = 0.071"

Linear Density Filter, Conic Weighting r_{min} = 0.071"

Linear Density Filter, Constant Weighting r_{min} = 0.12"

Linear Density Filter, Conic Weighting r_{min} = 0.12"

Linear Density Filter, Constant Weighting r_{min} = 0.18"

Linear Density Filter, Conic Weighting r_{min} = 0.18"

Figure 5-7. Final topologies for various filter sizes using a linear density filter. (Left) Constant sensitivity filter weights, (right) conic/linear sensitivity filter weights.

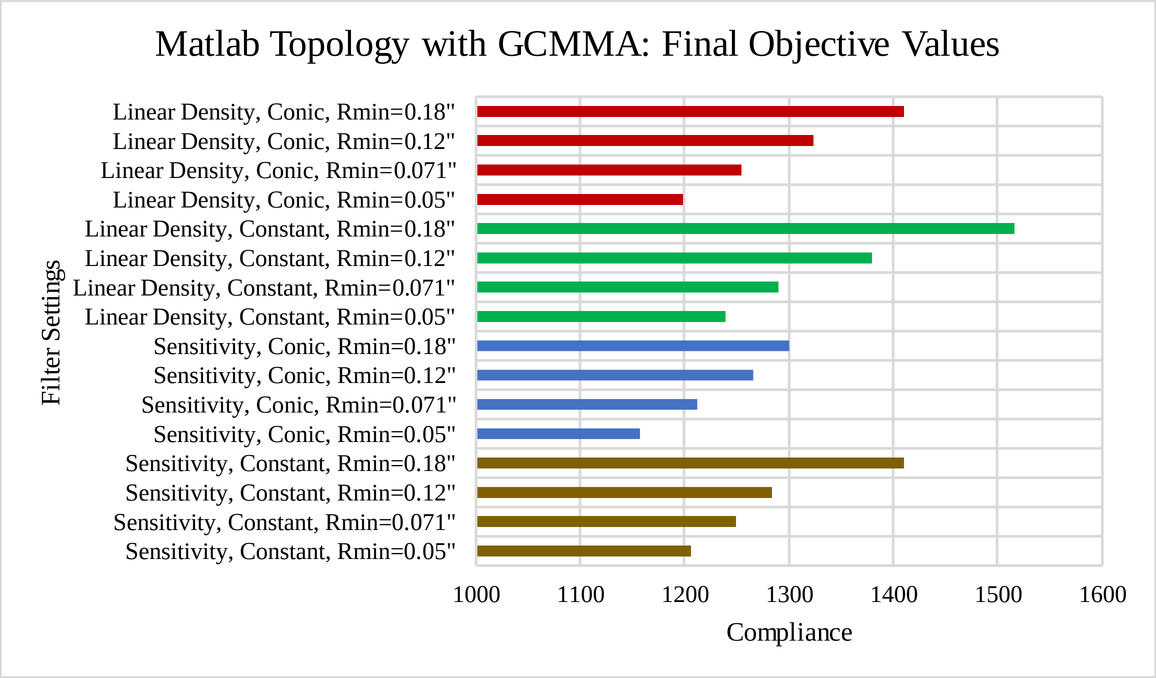


Figure 5-8. The final objective values for MATLAB topology using GCMMA for the various filter settings assessed in this section.

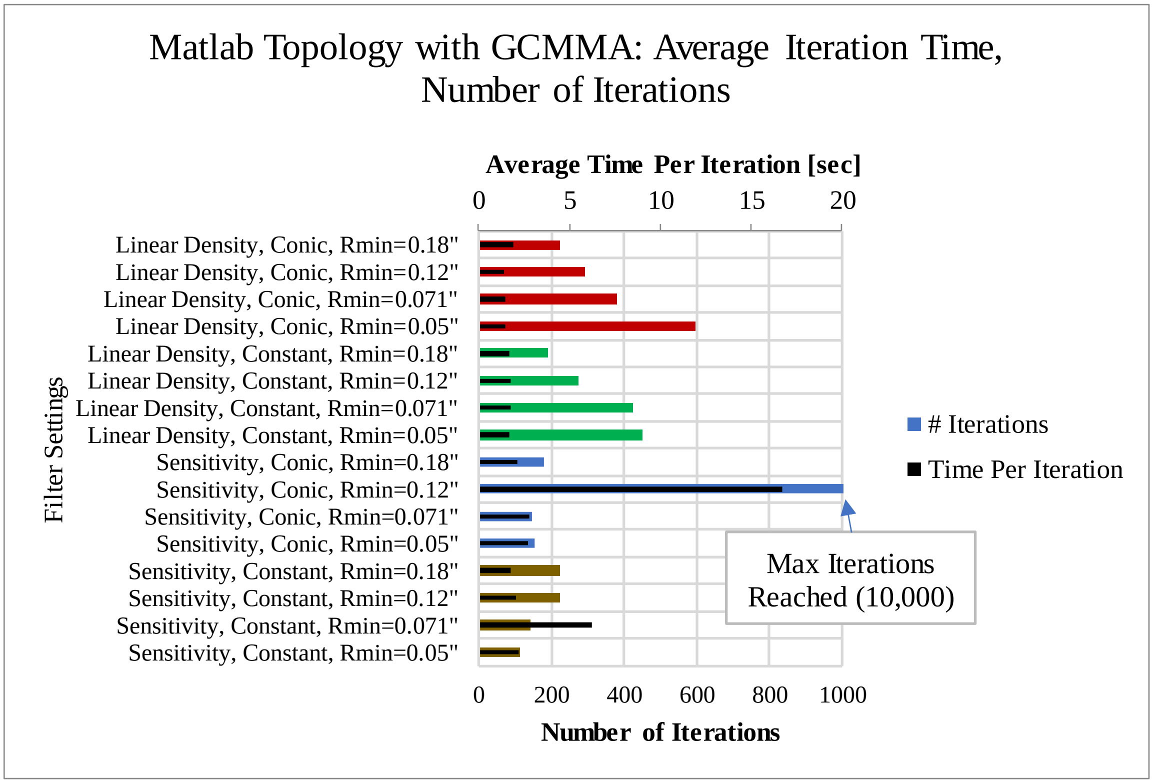


Figure 5-9. The number of iterations required to reach convergence and the average per iteration time, in seconds, for MATLAB topology using GCMMA for the various filter settings assessed in this section.

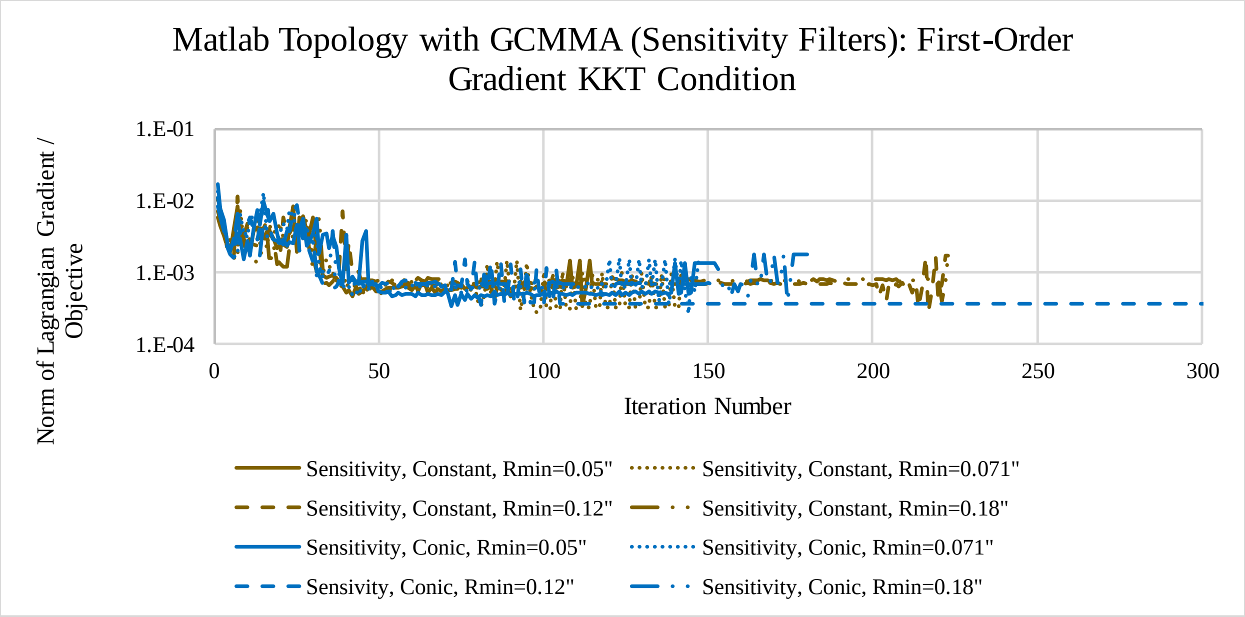


Figure 5-10. The first-order gradient KKT condition iteration history for MATLAB topology using GCMMA and a sensitivity filter for various filter radii and weighting matrices.

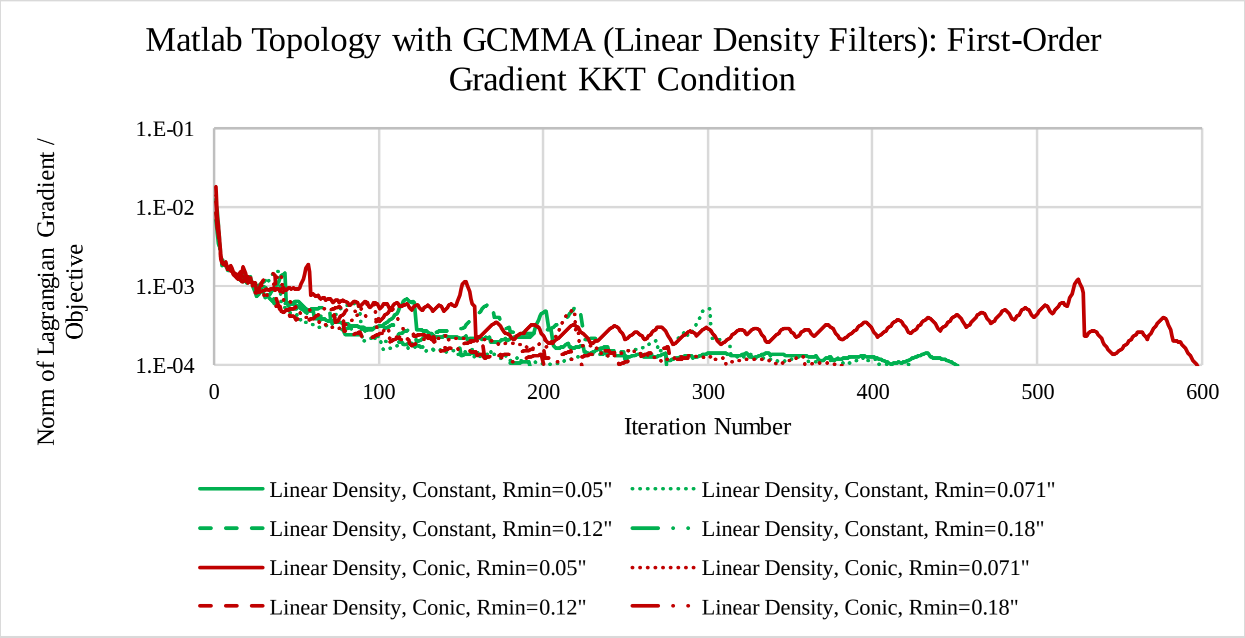


Figure 5-11. The first-order gradient KKT condition iteration history for MATLAB topology using GCMMA and a sensitivity filter for various filter radii and weighting matrices.

5.3.1. Weighting Matrix Choice

The choice of either a constant or a conic weighting matrix is discussed in this section. In general, when conic weighting matrices are used, the final topologies appear sharper and contain fewer intermediate density elements, particularly along the boundaries of the optimized topologies. This effect makes sense intuitively, as a constant (non-weighted) average over a spatial area tends to obfuscate and blur all topological features when compared to weightings that are based on proximity to a given element. The increased region of intermediate densities is usually undesirable in the optimized topology design

While both weighting matrix choices are mostly sufficient for addressing the checkerboard problem, the constant weighting matrix may exhibit patterns similar to checkerboards when the topological features are of the same order of size of the filter radius, as is the case for the upper-left most image in Figure 5-7. Furthermore, the choice of weighting matrix has the potential to alter the overall final topology; such a situation occurred in the case of the linear density filter with 0.05” and 0.12” filter radii. Regardless of the filter type, a conic weighting matrix tends to produce an overall lower objective than an identical optimization set up to use a constant weighting matrix. Finally, the choice of filter weighting matrix does not appear to strongly affect either the rate of convergence, the ability to satisfy the first-order gradient KKT condition, or the overall solution speed.

As mentioned in Section 2.3, the sensitivity filter described by Sigmund [4] makes use of a conic weighting matrix. Furthermore, engineering judgment supports the idea that for a given filter, elements near the element that is undergoing the filter operation should be weighted more heavily. For these reasons, it is speculated that all commercial software that make use of filtering algorithms will utilize a conic weighting matrix or, perhaps, a modified version of the conic weighting matrix that otherwise accounts for element distances in the weighting values. All subsequent MATLAB-based results presented in Section 5.4 of this paper will also make use of a conic weighting matrix.

5.3.2. Sensitivity versus Linear Density Filtering

The choice of a sensitivity versus a linear density filter is discussed in this section. In general, the final topologies predicted using either sensitivity or linear density filtering are quite similar. When difference in the final topologies are present, they are typically minor changes, such as one or more struts positioned in slightly different locations or orientations. Topologies produced with sensitivity filters tend to have slightly lower final objectives than similar topologies produced with a linear density filter. However, the difference in these final objectives is quite small. The most significant differences between the two filter types are primarily a tradeoff between solution times and convergence accuracy.

Because sensitivity filtering heuristically modifies the objective sensitivity, the first-order gradient terms are unlikely to be satisfied when using a sensitivity filter. Indeed, in Figure 5-10, the first-order gradient term never drops below a value of about 5.E-4 throughout the optimization. Furthermore, the first-order gradient term exhibits “jumpy” behavior that may be a direct result of poor estimates of the Lagrange multiplier which are themselves a consequence of feeding a heuristically modified sensitivity to the GCMMA algorithm. With sensitivity filtering, convergence via GCMMA was achieved by satisfying “slowed” convergence criterion. This convergence criterion is weaker than convergence based on the first-order gradient KKT condition, and thus proving convergence to a local minimum using a sensitivity filter seems unlikely. One of the examples using the sensitivity filter (conic weighting with a 0.12” filter radius) resulted in an excessive number of iterations and a very high per iteration solution time. This behavior may be attributed to the inaccurate sensitivity information being fed to the GCMMA algorithm, which is apparently unable to reach any convergence criteria prior to reaching the maximum number of iterations. While this specific example may be able to reach convergence by carefully tuning the parameters of the GCMMA optimizer, it nonetheless highlights one of the expected deficiencies of sensitivity filtering when used in conjunction with math programming optimization algorithms.

On the other hand, the linear density filter achieves convergence based on the first-order gradient KKT condition for all the filter parameters assessed in this paper. As expected, in Figure 5-11, the first-order gradient term decreases monotonically with the number of iterations, which suggests the optimization is at the very least converging to a local optimum. Linear density filtering does tend to require a larger number of iterations to reach convergence. However, the larger number of iterations leads to a design that may be proven to be a local optimum. Sensitivity filtering, on the other hand, may converge more quickly via the “slowed” criterion, but this convergence may also be a result of the optimizer failing to identify an improved design based upon the heuristically modified sensitivities it is fed.

The time required per iteration is similar between the sensitivity and linear density filter solutions. While there are several outliers, such as the constant weighting matrix for sensitivity filtering, it is not expected that the choice of filter strongly impacts the per iteration solution time.

5.3.3. Effect of Filter Size

The impact of filter radius on the final design is discussed in this section. Using a filter radius that encompasses a minimal set of elements (one on each side of a given element) is enough to remove the majority of checkerboarding from the final topology. The use of larger filter radii seems to introduce a mesh-independent length scale into the topology optimization problem, which is a conclusion supported by findings by Lazarov, Wang, & Sigmund [25]. As the filter radius is increased to 0.12” and 0.18”, the topologies resulting from both sensitivity and linear density filters approach a simple triangular structure.

Larger filter radii also increase the amount of the final design in which intermediate densities are present. The presence of more intermediate densities and larger structural members collectively act to increase the final objective. In general, the final objective from topology optimization increases as the filter radius is increased. For linear density filtering, the size of the filter is also related to the convergence speed. Smaller filter radii result in a larger number of iterations required to achieve convergence, while larger filter do the opposite. This conclusion makes sense intuitively, as the optimizer has a larger amount of freedom of generating finer-sized topological features when the filter radius is small. Larger filter radii indirectly act to limit the freedom of the optimizer. For both sensitivity and linear density filtering, the filter radius was not strongly associated with changes to the per iteration solution time.

5.4. MATLAB Topology Optimization using OC

In this section, topologies results were generated using the OC recursion power law relationship derived in Section 3.1.1. Comparisons are made between OC recursion using a sensitivity filter, OC recursion using the linear density filter modification discussed in Section 3.1.2, and GCMMA using a linear density filter. In this section, only conic weighting matrices and two filter radii are considered.

Figure 5-12 shows comparisons of OC with sensitivity filtering, OC with linear density filtering, and GCMMA with linear density filtering using conic weighting matrices with a filter radius of 0.05” and a filter radius of 0.12”. Figure 5-13 shows comparisons of the final objective value from the topology optimization for each method and filter radius. Figure 5-14 shows comparisons of both the total number of iterations required for convergence and the average user time spent on each iteration. Finally, Figure 5-15 shows the first-order gradient condition (the infinity norm of the Lagrangian gradient, normalized by the objective) iteration history for each method.

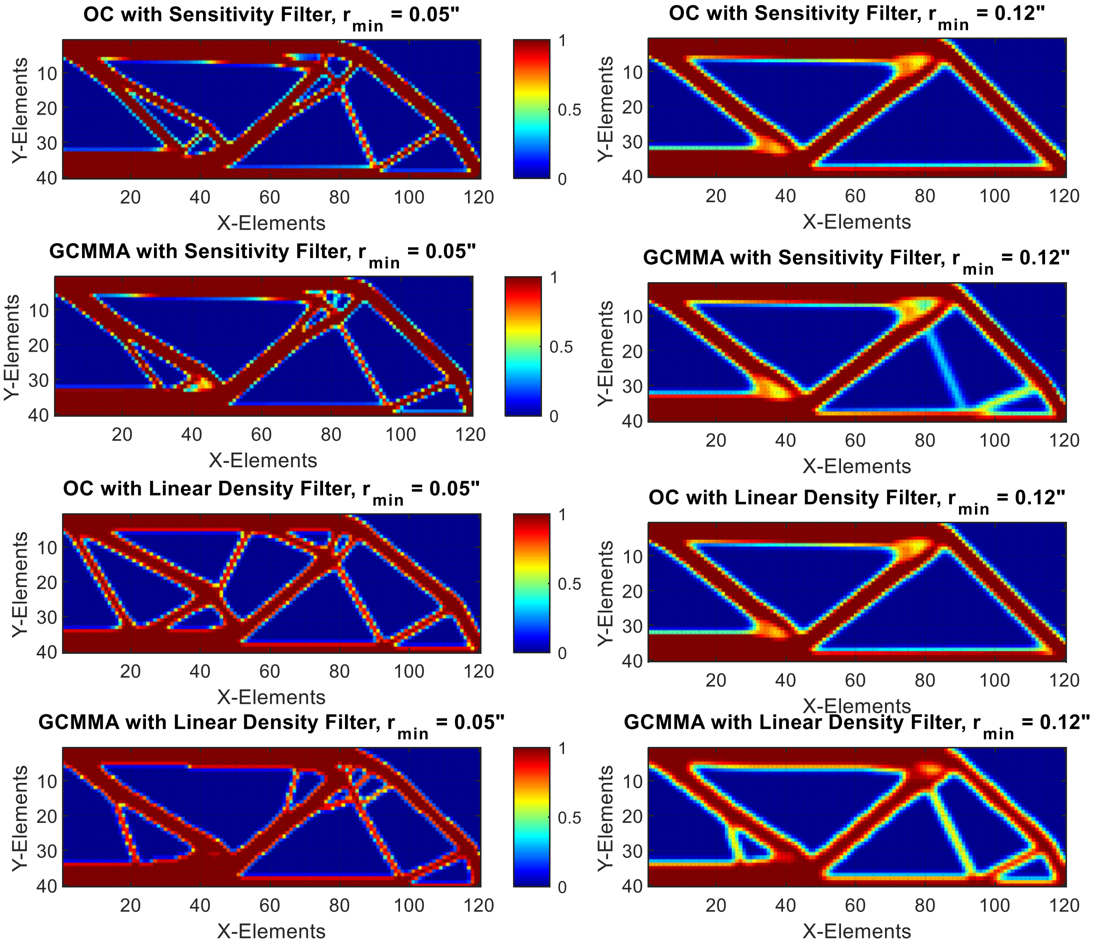


Figure 5-12. Final topologies for OC with sensitivity filtering, OC with linear density filtering, and GCMMA with linear density filtering. (Left) Rmin = 0.05” (right) Rmin=0.12”.

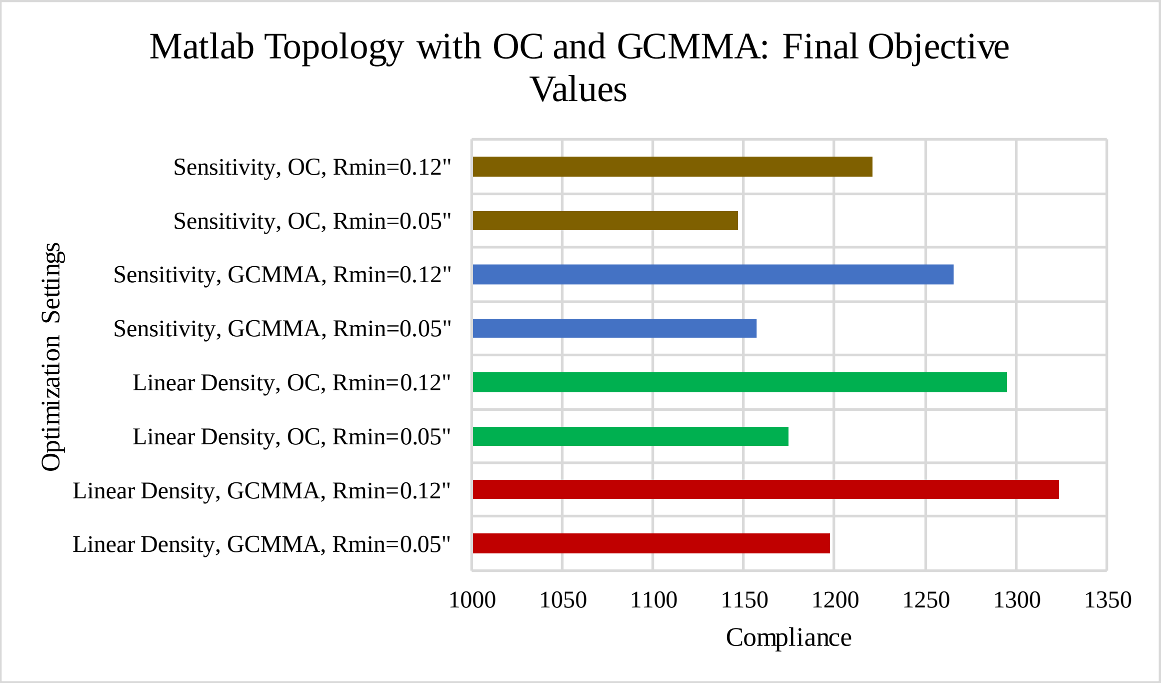


Figure 5-13. The final objective values for MATLAB topology using OC with sensitivity and linear density filtering and GCMMA with linear density filtering.

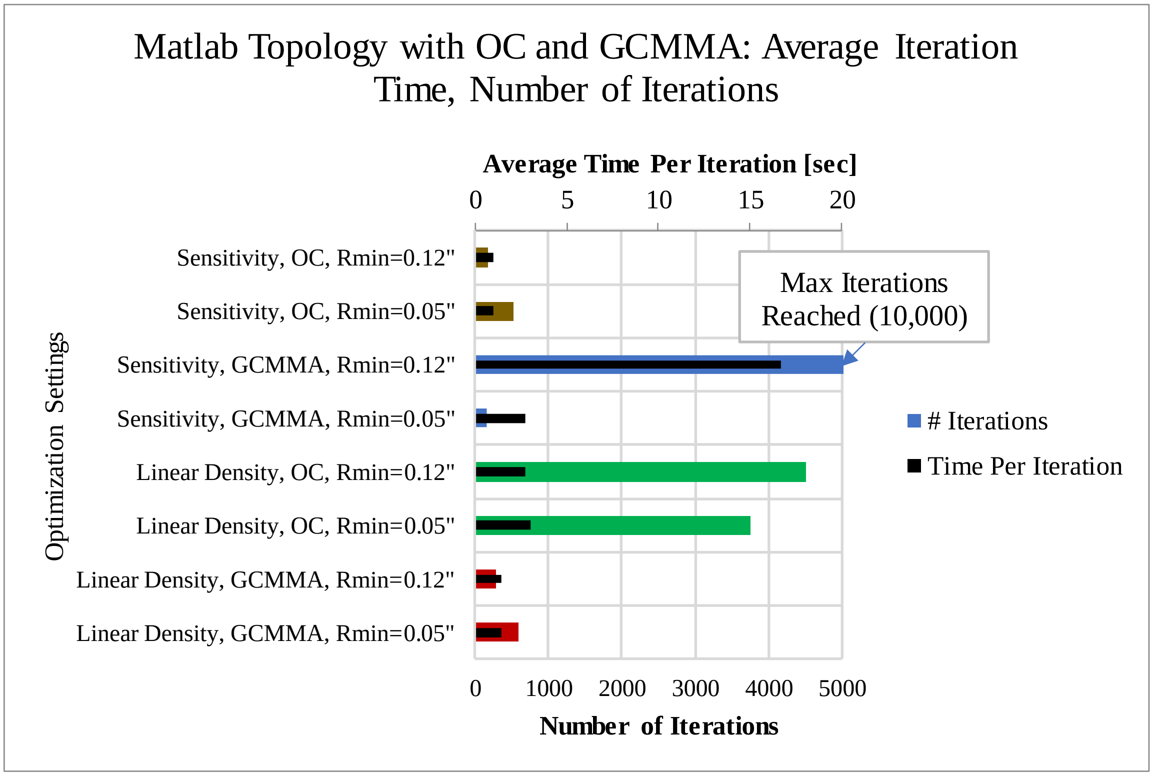


Figure 5-14. The number of iterations required to reach convergence and the average per iteration time, in seconds, for OC with sensitivity and linear density filtering and GCMMA with linear density filtering.

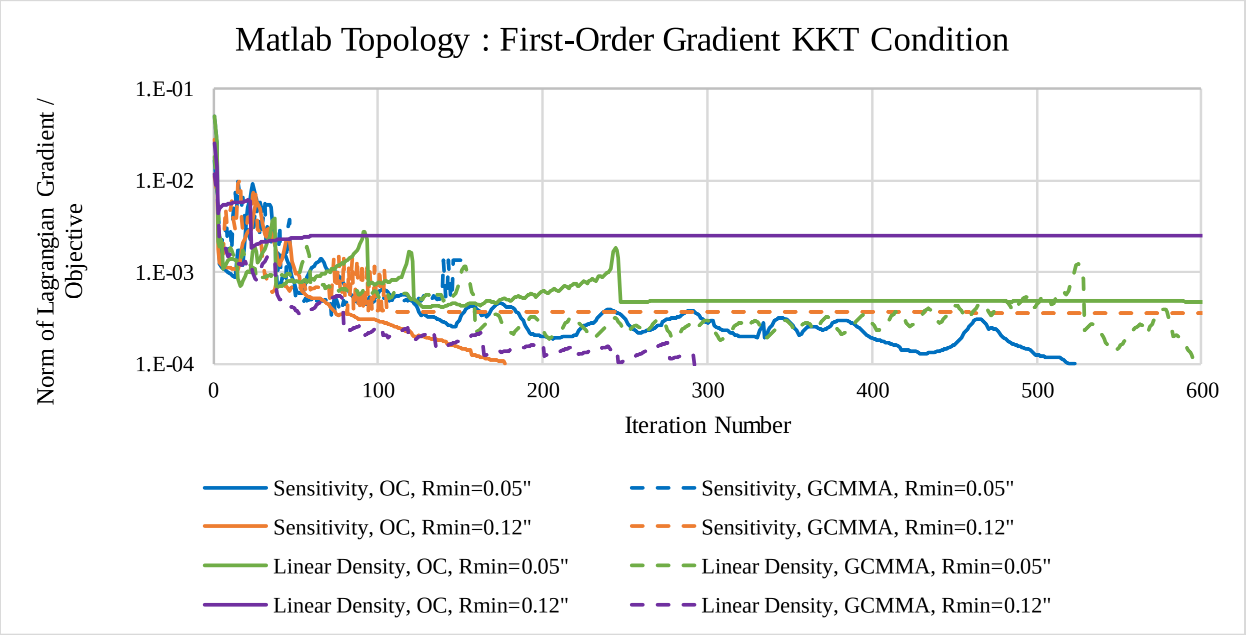


Figure 5-15. The first-order gradient KKT condition iteration history for OC with sensitivity and linear density filtering and GCMMA with linear density filtering.

In general, realistic topologies may be generated using either OC or GCMMA approaches with either sensitivity of linear density filtering. Topology results for OC and GCMMA using a sensitivity filter are nearly indistinguishable. Differences are apparent in the OC and GCMMA results using a linear density filter, which may suggest further refinements are required for the implementation of OC using the linear density filter. In terms of final compliance values, using a sensitivity filter in conjunction with either OC or GCMMA update techniques tend to slightly outperform results using a linear density filter.

More substantial differences between OC and GCMMA implementations arise when considering the convergence behavior and solution times. The use of OC with a sensitivity filter is typically the least expensive implementation. Surprisingly, the use OC with a sensitivity filter also provides sufficiently accurate approximations of the Lagrange multiplier, and convergence based on satisfying the first-order gradient of the Lagrangian is achieved for each of the filter radii investigated when a sensitivity filter is used with OC. Use of GCMMA with a linear density filter incurs a slightly higher per iteration time and required number of iterations compared to OC with a sensitivity filter, but is also able to satisfy the first-order gradient condition. The use of a linear density filter with OC incurs a per iteration computational cost associated with the decomposition operation shown in Equation 3-16. The examples of OC with the linear density filter presented in this paper each converge based upon “slowed” criteria and were unable to satisfy the first-order gradient of the Lagrangian. It is currently unknown why convergence based on the first-order gradient condition was unable to be achieved using OC with a linear density filter, and further investigation is required.

5.5. Commercial Software Results: 2D MBB

In general, making accurate comparisons between different commercial solvers is a difficult, if not impossible task. The lack of documentation with regards to details on element formulation, optimization algorithms, filtering algorithms, etc. mean that fair comparisons may very rarely be made. Nonetheless, Figure 5-16 shows a comparison between each of the four commercial software programs evaluated and the MATLAB topology implementation using GCMMA. Figure 5-16 may be used to draw several general conclusions. First, most of the commercial software documentation indicates that some modified form of MMA or GCMMA is used as the optimizer for topology optimization problems. Several solvers also offer alternatives based on OC or linear programming methods, but methods based on MMA are more commonly used as default optimizers for topology optimization problems. Next, all four commercial software evaluated use convergence criteria based on changes to either the objective function, the norm of the design variables, or both. Were a linear density filter implemented in any case, it would seem reasonable that convergence based on the Lagrangian gradient might be an option as well, but that is not the case for any commercial software evaluated. In Figure 5-16, topology results produced using ANSYS, TOSCA Structure, and MSC Nastran all closely resemble results from the MATLAB implementation of GCMMA using a sensitivity filter with a filter radius that encompasses only elements immediately adjacent to a given element. These comparisons suggest that it is possible that ANSYS, TOSCA Structure, and MSC Nastran each use a sensitivity filter when their MMA-like optimizer is selected. Furthermore, it seems likely that each filtering algorithm has its filtering radius set such that the filter includes a single layer of elements connected to a given element. This assumption about the filter radius is known to be true for TOSCA Structure. Simcenter Nastran appears to be an outlier compared to the other commercial software; possible reasons for this discrepancy are discussed in Section 5.5.4.

Additional details and conclusions from the evaluation of individual commercial software are provided in the following subsections. The convergence criteria used for each example presented in this section are based on changes to the objective and the design variables satisfying a prescribed tolerance value. However, the objective and design variable change tolerances are not necessarily consistent between the examples presented in this section.

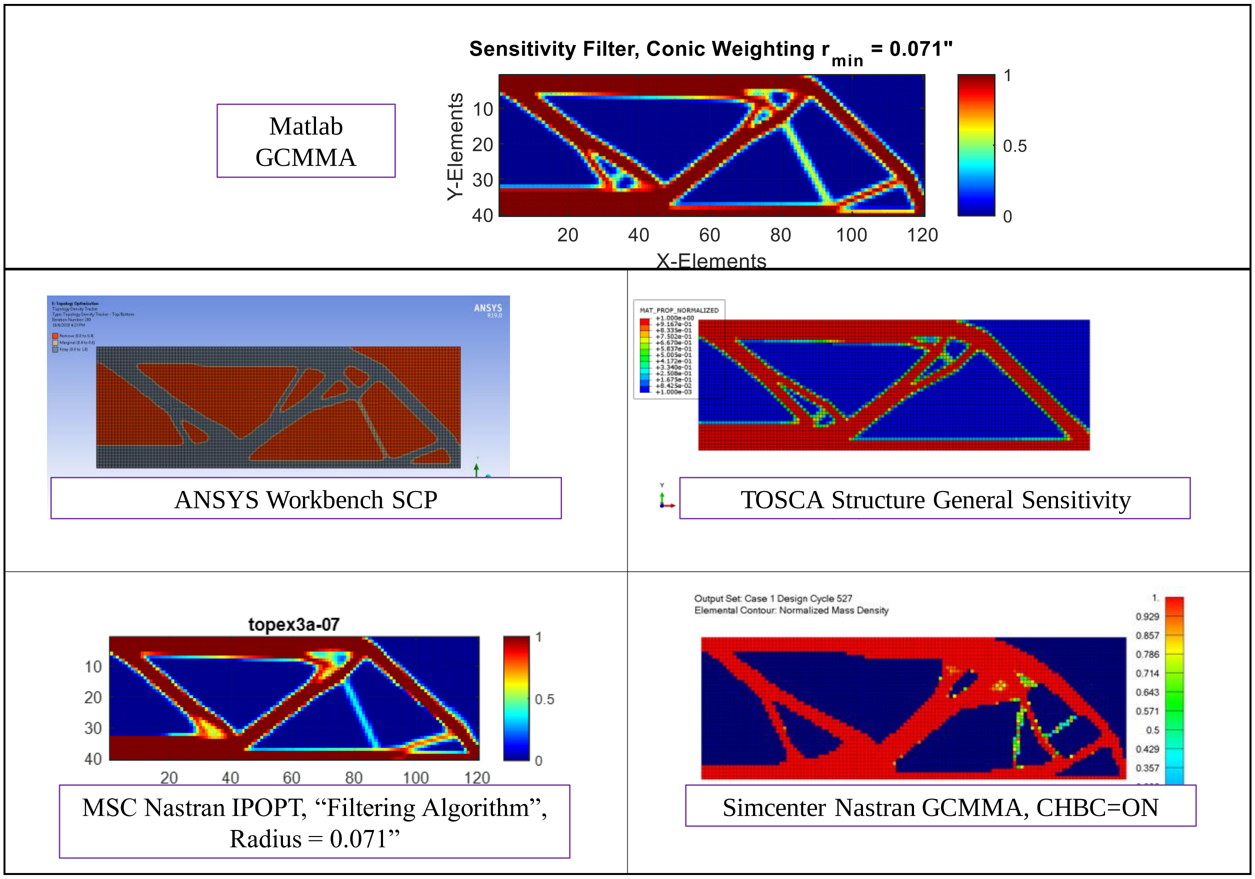


Figure 5-16. Comparisons between MATLAB GCMMA results using a sensitivity filter and similar topology designs for commercial software implementations.

5.5.1. ANSYS Workbench

Results for the 2D MBB compliance problem produced using the SCP and OC optimization algorithms of ANSYS workbench are shown in Figure 5-17. In both cases, SIMP is used with a penalty value of 3. Both optimization algorithms produce topologies that resemble results produced using MATLAB implementations of GCMMA and OC. Because the ANSYS filtering algorithm is always enabled, no further investigation was performed regarding checkerboarding control for ANSYS.

ANSYS workbench produces output formats that are somewhat different compared to the other commercial software evaluated in this paper. For example, by default, ANSYS will produce "remove", "marginal", and "keep" regions of the topology results based on user-supplied input values. However, a means to output or plot raw elemental pseudo-density values in ANSYS was not identified. Furthermore, ANSYS reports the objective iteration histories as a fractional improvement with regards to the initial objective. Combined, these effects made it difficult to make more detailed comparisons between ANSYS and other commercial software.

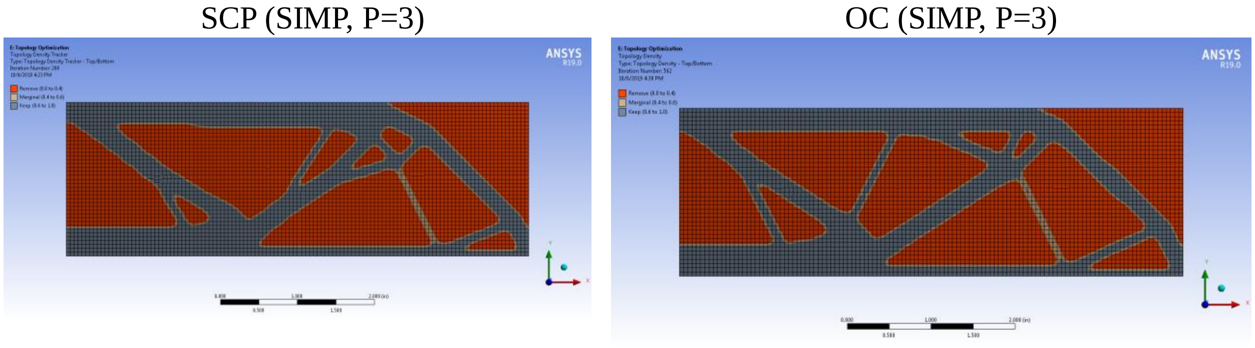


Figure 5-17. ANSYS topology optimization results for the 2D MBB compliance problem using the SCP and OC optimizers.

5.5.2. TOSCA Structure

Results for the 2D MBB compliance problem using the controller-based and general sensitivity optimization algorithms of TOSCA Structure using Abaqus/Standard FEA are shown in Figure 5-18. In both cases, SIMP is used with a penalty value of 3. The general sensitivity algorithm produces topologies that resemble results produced using MATLAB implementations of GCMMA and OC when the sensitivity filter radius is set to be 0.07”. Results from the controller-based algorithm were particularly promising. The final objective for the controller-based algorithm was lower than the same problem solved using the general sensitivity algorithm, it converged in only 15 iterations, and the final topology was completely black and white (contained no intermediate density elements). The controller-based algorithm is acknowledged as an incredibly useful tool for solving compliance minimization problems in both 2D and 3D cases. Furthermore, the controller-based algorithm scales up quite well for problems with many degrees of freedom, as it can fully converge with only 15 total finite element analyses required. The default filter settings were used for all results presented in this section (that is, the “standard” filter with a filter radius equal to 1.3 times the average element edge length). Because the TOSCA Structure filtering algorithm is always enabled, no further investigation was performed regarding checkerboarding control for TOSCA Structure.

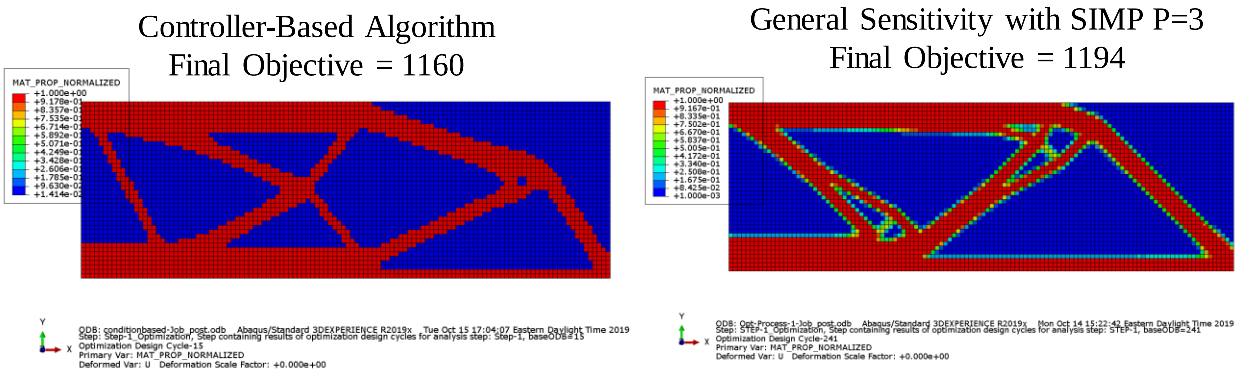

Controller-Based Algorithm
Final Objective = 1160
MAT_PROP_NORMALIZED
+1.000e+00
+9.178e-01
+8.357e-01
+7.535e-01
+6.714e-01
+5.892e-01
+5.071e-01
+4.249e-01
+3.428e-01
+2.606e-01
+1.785e-01
+9.630e-02
+1.414e-02
Y
X
ODB: conditionbased-Job_post.odb Abaqus/Standard 3DEXPERIENCE R2019x Tue Oct 15 17:04:07 Eastern Daylight Time 2019
Step: Step-1_Optimization, Step containing results of optimization design cycles for analysis step: Step-1, baseODB=15
Optimization Design Cycle-15
Primary Var: MAT_PROP_NORMALIZED
Deformed Var: U Deformation Scale Factor: +0.000e+00
General Sensitivity with SIMP P=3
Final Objective = 1194
MAT_PROP_NORMALIZED
+1.000e+00
+9.167e-01
+8.335e-01
+7.502e-01
+6.670e-01
+5.837e-01
+5.005e-01
+4.172e-01
+3.340e-01
+2.508e-01
+1.675e-01
+8.425e-02
+1.000e-03
Y
X
ODB: Opt-Process-1-Job_post.odb Abaqus/Standard 3DEXPERIENCE R2019x Mon Oct 14 15:22:42 Eastern Daylight Time 2019
Step: STEP-1_Optimization, Step containing results of optimization design cycles for analysis step: STEP-1, baseODB=241
Optimization Design Cycle-241
Primary Var: MAT_PROP_NORMALIZED
Deformed Var: U Deformation Scale Factor: +0.000e+00

Figure 5-18. TOSCA Structure topology optimization results for the 2D MBB compliance problem using the controller-based and sensitivity-based optimizers.

5.5.3. MSC Nastran

Because MSC Nastran offers two different methods of addressing the checkerboarding problem, these methods were assessed separately. For all evaluations of MSC Nastran topology optimization, SIMP is used with a penalty value of 3.

Table 5-1 shows eight different sets of input parameters used to solve the 2D MBB compliance problem using the MSC Nastran “filtering algorithm,” along with the final objective, constraint, and convergence behavior. The topologies associated with each of the cases presented in Table 5-1 are shown in Figure 5-19. It was also observed that, in many cases, MSC Nastran terminated the optimization at a relatively low number of iterations, issuing a “sufficiently black and white” message. It appears that MSC Nastran topology optimization includes an independent logical check to determine whether it appears the topology is converging and, if detected, overrides any user-defined objective convergence tolerances.

When no filtering algorithm is present, checkerboarding is prevalent in the final topology. When the “filtering algorithm” is enabled, the checkerboarding problem is addressed. However, the filter radius (TDMIN value) does not impact the topology optimization results until it reaches a value of 0.15 inches. It appears likely that the TDMIN represents the filtering diameter (twice the filter radius discussed elsewhere in this paper). Enforcing a TDMIN value of 0.15 corresponds to a filter that includes the eight elements adjacent to a given element. A TDMIN value between 0.01 and 0.06 all result in identical topologies as computed by MSC Nastran. Therefore, it seems likely that by default, MSC Nastran will include, at a minimum, the four elements immediately adjacent to a given element in its filtering algorithms. As the TDMIN value is further increased, the topologies from MSC Nastran begin to resemble the MATLAB OC and GCMMA results for increasingly large filter radii.

Table 5-1. Parameter settings used to evaluate various filter radii for the MSC Nastran “filtering algorithm” (TCHECK=1).

Run	DOPTPRM									CONVERGENCE			
	DESMAX	CTMIN	GMAX	DELX	DXMIN	CONV1	CONVDV	TCHECK	TDMIN	f_final	g_final	Type	Iteration
1	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	0	0.010	1,282.3	9.937E-09	HARD	25
2	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	1	0.010	1,188.3	9.997E-09	HARD	33
3	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	1	0.060	1,188.3	9.997E-09	HARD	33
4	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	1	0.090	1,188.3	9.997E-09	HARD	33
5	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	1	0.120	1,188.3	9.997E-09	HARD	33
6	**1000**	**1.00E-06**	**1.00E-06**	**2.00E-01**	**1.00E-08**	**1.00E-06**	**1.00E-06**	**1**	**0.150**	**1,176.8**	**9.444E-09**	**HARD**	**77**
7	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	1	0.180	1,199.6	9.759E-09	HARD	73
8	1000	1.00E-06	1.00E-06	2.00E-01	1.00E-08	1.00E-06	1.00E-06	1	0.210	1,236.0	9.995E-09	HARD	94

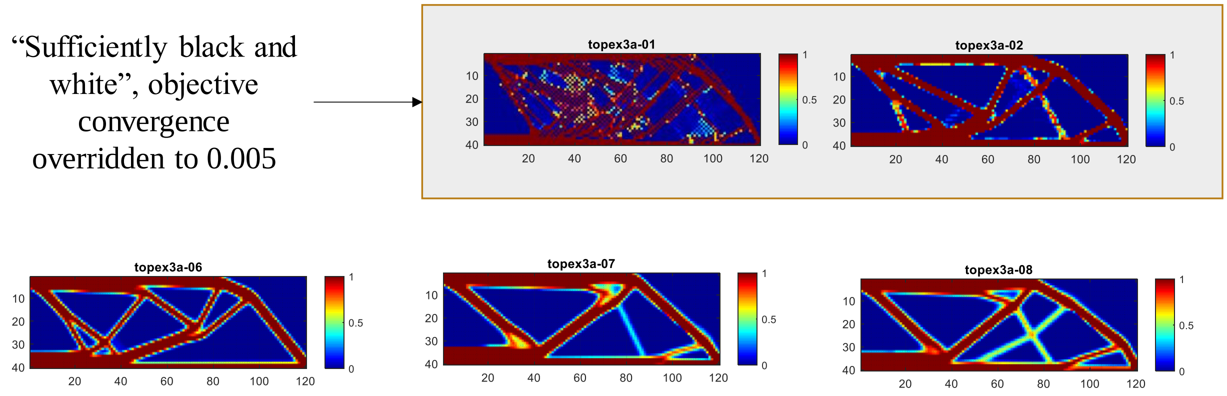

“Sufficiently black and white”, objective convergence overridden to 0.005

Figure 5-19. MSC Nastran topology optimization results for the 2D MBB compliance problem IPOPT optimizer and the “filtering algorithm” with various filter radii.

Table 5-2 shows five different sets of input parameters used to solve the 2D MBB compliance problem using the MSC Nastran “density constraint” algorithm, along with the final objective, constraint, and convergence behavior. Interestingly, for low values of TDMIN using the “density constraint” algorithm, the optimization proceeds for hundreds of iterations before reaching the maximum scratch disk space capacity. These analyses were performed on a Linux high performance computing cluster, and so it is presently unknown what internal operations resulted in the MSC Nastran topology optimization runs encountering scratch disk space issues. For larger values of TDMIN, the resulting topologies for the “density constraint” begin to resemble the results from the MATLAB implementation of GCMMA and OC when either a linear density or a sensitivity filter is implemented.

Therefore, it is suspected that the MSC Nastran “filtering algorithm” is an implementation of a sensitivity filter, whereas the “density constraint” is an implementation of a linear density filter, and is not an optimization constraint, as the name might imply.

Table 5-2. Parameter settings used to evaluate various filter radii for the MSC Nastran “density constraint” (TCHECK=2).

Run	DOPTPRM									CONVERGENCE			
	DESMAX	CTMIN	GMAX	DELX	DXMIN	CONV1	CONVDV	TCHECK	TDMIN	f_final	g_final	Type	Iteration
10	2500	1.00E-06	1.00E-06	1.00E-01	1.00E-10	1.00E-06	1.00E-06	2	0.120			FATAL	1632
11	2500	1.00E-06	1.00E-06	1.00E-01	1.00E-10	1.00E-06	1.00E-06	2	0.150			FATAL	1632
12	2500	1.00E-06	1.00E-06	1.00E-01	1.00E-10	1.00E-06	1.00E-06	2	0.180			FATAL	1632
13	2500	1.00E-06	1.00E-06	1.00E-01	1.00E-10	1.00E-06	1.00E-06	2	0.210	1,330.9	1.000E-08	HARD	627
14	2500	1.00E-06	1.00E-06	1.00E-01	1.00E-10	1.00E-06	1.00E-06	2	0.500	1,666.9	5.115E-09	HARD	273

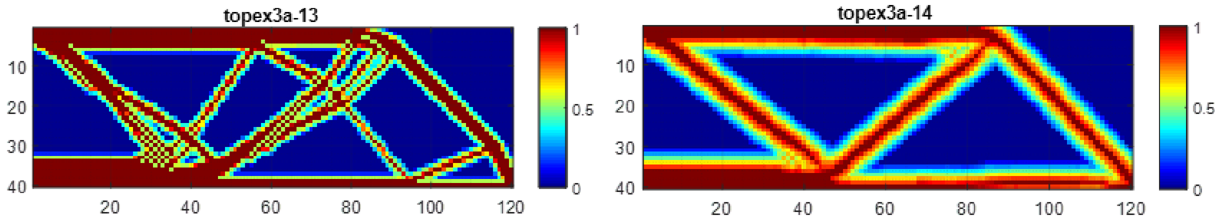
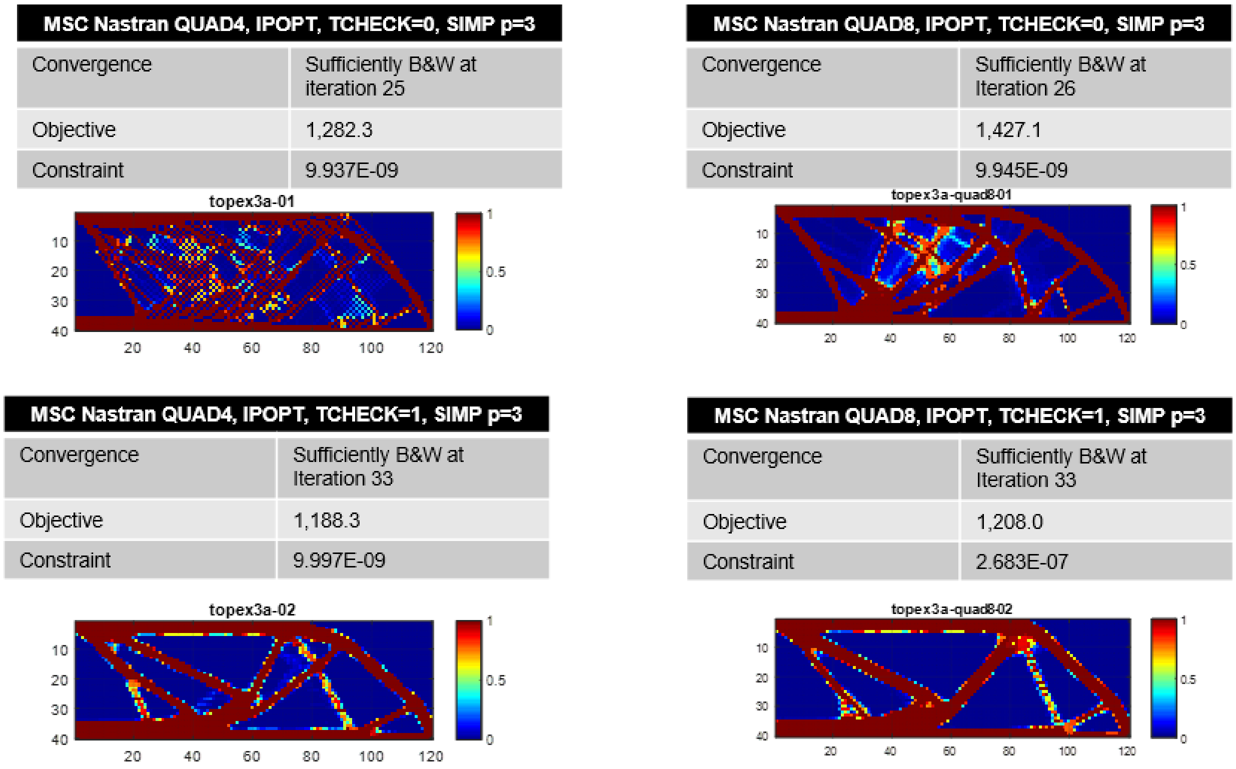


Figure 5-20. MSC Nastran topology optimization results for the 2D MBB compliance problem IPOPT optimizer and the “density constraint” with various filter radii.

As a final evaluation of the filtering algorithms in MSC Nastran, four separate test cases of the 2D MBB compliance problem were set up using linear and parabolic quadrilateral elements and toggling on and off the MSC Nastran “filtering algorithm,” as shown in Figure 5-21. From these results, the use of parabolic quadrilateral elements in place of linear quadrilateral elements greatly reduces the prevalence of checkerboarding when no other filtering algorithm is present. In many cases, the use of parabolic quadrilateral elements alone is adequate to address the checkerboarding problem. Nonetheless, including the “filtering algorithm” appears to impact the topology optimization results even when parabolic quadrilateral elements are included, albeit to a lesser extent.



MSC Nastran QUAD4, IPOPT, TCHECK=0, SIMP p=3	
Convergence	Sufficiently B&W at iteration 25
Objective	1,282.3
Constraint	9.937E-09

MSC Nastran QUAD8, IPOPT, TCHECK=0, SIMP p=3	
Convergence	Sufficiently B&W at Iteration 26
Objective	1,427.1
Constraint	9.945E-09

MSC Nastran QUAD4, IPOPT, TCHECK=1, SIMP p=3	
Convergence	Sufficiently B&W at Iteration 33
Objective	1,188.3
Constraint	9.997E-09

MSC Nastran QUAD8, IPOPT, TCHECK=1, SIMP p=3	
Convergence	Sufficiently B&W at Iteration 33
Objective	1,208.0
Constraint	2.683E-07

Figure 5-21. MSC Nastran topology optimization results for the 2D MBB compliance using linear and parabolic quadrilateral elements, without filtering and with the default “filtering algorithm” filter settings.

5.5.4. Simcenter Nastran

Because Simcenter Nastran (formerly known as NX Nastran) offers a toggle that controls its implementation of a filtering algorithm, results from Simcenter Nastran for the 2B MBB compliance problem were assessed both with and without the filtering algorithm active. For all evaluations of Simcenter Nastran topology optimization, SIMP is used with a penalty value of 3.

Results for the 2D MBB compliance problem produced using the SPOT and GCMMA optimization algorithms of Simcenter Nastran both with and without the checkerboard control toggle enabled are shown in Figure 5-22. When the checkerboard control toggle is disabled, the topologies produced using both SPOT and GCMMA optimizers resembles the topologies obtained using the MATLAB implementation of GCMMA without a filter. However, the Simcenter Nastran topologies obtained when the checkerboard control toggle is enabled tend to differ from not only all MATLAB implementations presented herein, but also all evaluations of other commercially available topology optimization software.

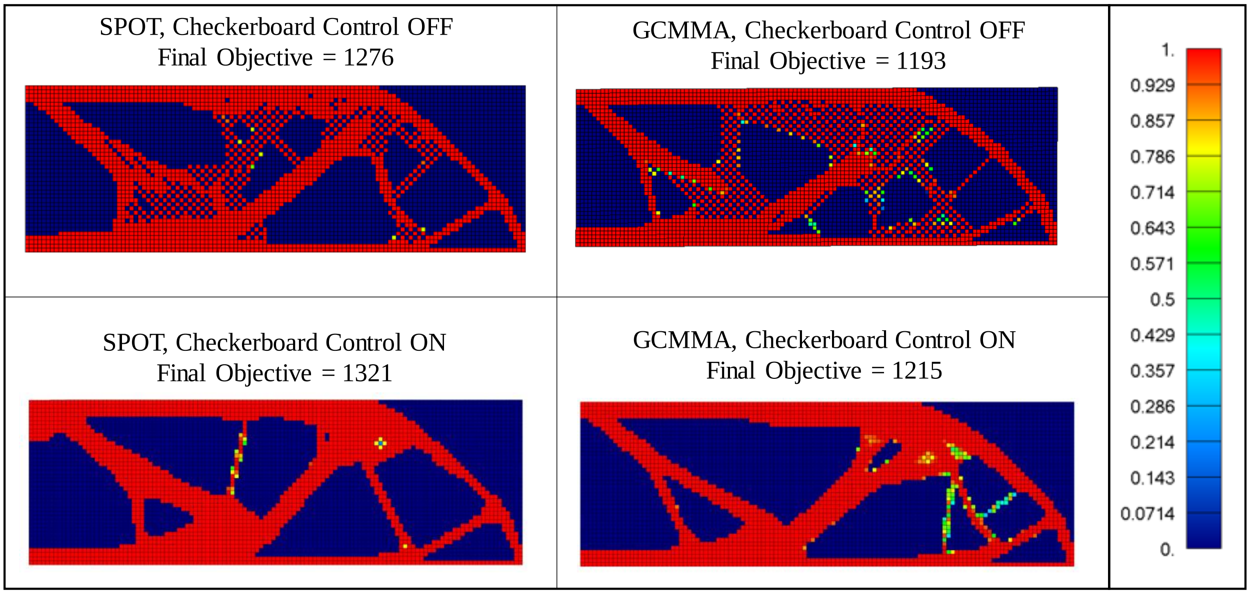


Figure 5-22. Simcenter Nastran topology optimization results for the 2D MBB compliance problem using the default SPOT optimizer and the GCMMA optimizer, with and without the checkerboard control toggle enabled.

Although the Simcenter Nastran checkerboard control toggle is quite effective at removing the checkerboarding problem, the uniqueness of its behavior warranted further investigation. Figure 5-23 shows a comparison of the objective and constraint iteration histories for both Simcenter (NX) Nastran GCMMA with checkerboard control enabled, and MSC Nastran with the "filtering algorithm" enabled. Two conclusions may be drawn from Figure 5-23.

First, the inability to enforce an initial density assumption in Simcenter Nastran results in an initially infeasible design. Therefore, the first four iterations of the Simcenter Nastran 2B MBB compliance problem are spent increasing the objective so that the volume constraint may be satisfied. Starting at an infeasible design point is considered inefficient from a computational standpoint. The first several iterations of Simcenter Nastran topology optimization subject to a volume constraint must necessarily be spent bringing the problem to a feasible design point without lending any insight into the optimal topology of the problem. It is also well-understood that the initial density distribution assumption will often impact the final topologies obtained. Therefore, it is possible that a major source of the difference noted between Simcenter Nastran and MATLAB and other commercial software implementations might be attributed to differences in the initial density distribution assumption.

Next, strange “jumpy” behavior may be observed in the Simcenter Nastran constraint iteration history when the checkerboard control toggle is enabled. This behavior was not observed in any instances where the checkerboard control toggle was disabled and was also not observed when the problem was solved with similar settings in other commercial software. Therefore, it was concluded that the Simcenter Nastran checkerboard control is the source of this behavior. One likely explanation for the behavior is that the Simcenter Nastran checkerboard control is a heuristic method that diverges from the sensitivity or density filtering algorithms implemented elsewhere. For example, the checkerboard control toggle could be acting as a density filter on a localized level, or a sensitivity filter on a localized level, such that inaccuracies are introduced to the sensitivities being fed into the optimizer.

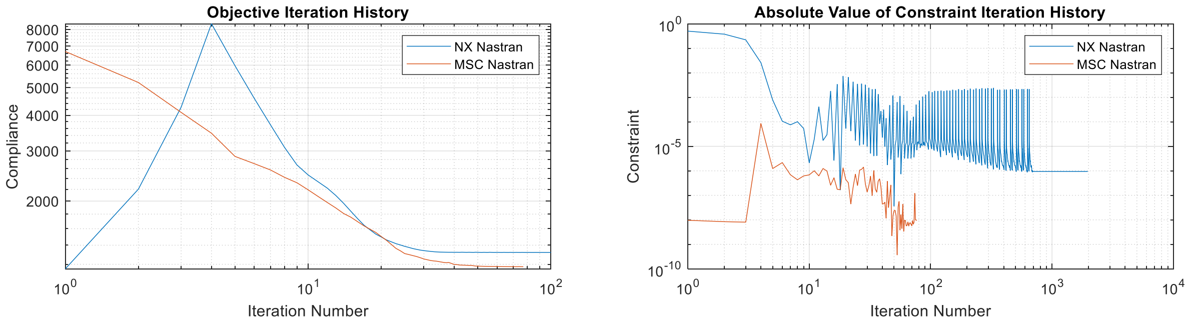
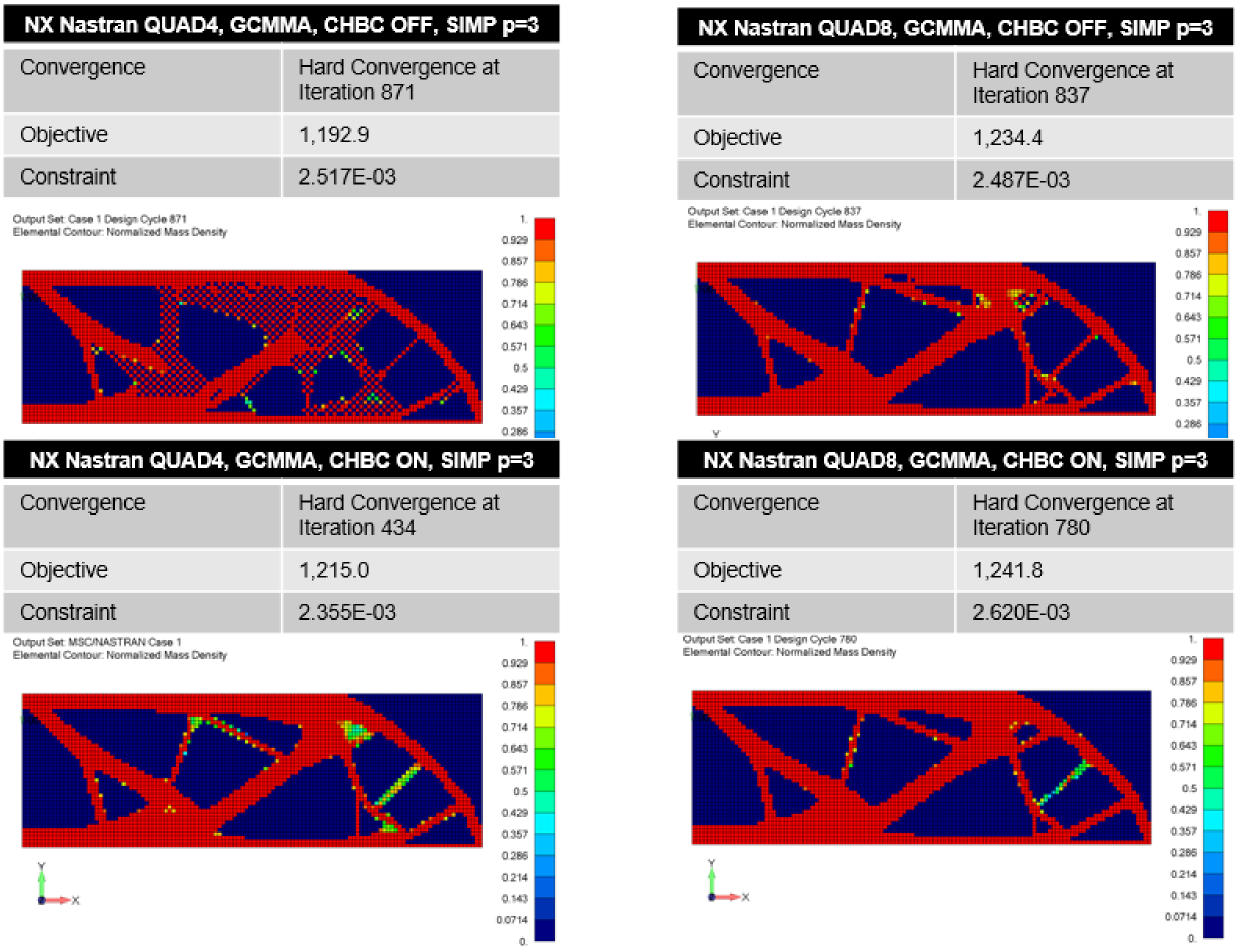


Figure 5-23. With the Simcenter Nastran checkerboard control toggle enabled, “jumpy” behavior in the constraint iteration history is observed; this behavior does not exist when the toggle is disabled and does not exist in MSC Nastran.

As a final evaluation of the checkerboard control algorithm in Simcenter Nastran, four separate test cases of the 2D MBB compliance problem were set up using linear and parabolic quadrilateral elements and toggling on and off the Simcenter Nastran checkerboard control, as shown in Figure 5-24. These results are consistent with similar findings for MSC Nastran that were discussed in Section 5.5.3. Namely, the use of parabolic quadrilateral elements in place of linear quadrilateral elements greatly reduces the prevalence of checkerboarding when no other filtering algorithm is present. In many cases, the use of parabolic quadrilateral elements alone is enough to address the checkerboarding problem. Nonetheless, including the checkerboard control algorithm appears to impact the topology optimization results even when parabolic quadrilateral elements are included, albeit to a lesser extent.



NX Nastran QUAD4, GCMMA, CHBC OFF, SIMP p=3	
Convergence	Hard Convergence at Iteration 871
Objective	1,192.9
Constraint	2.517E-03

NX Nastran QUAD8, GCMMA, CHBC OFF, SIMP p=3	
Convergence	Hard Convergence at Iteration 837
Objective	1,234.4
Constraint	2.487E-03

NX Nastran QUAD4, GCMMA, CHBC ON, SIMP p=3	
Convergence	Hard Convergence at Iteration 434
Objective	1,215.0
Constraint	2.355E-03

NX Nastran QUAD8, GCMMA, CHBC ON, SIMP p=3	
Convergence	Hard Convergence at Iteration 780
Objective	1,241.8
Constraint	2.620E-03

Output Set: Case 1 Design Cycle 871
Elemental Contour: Normalized Mass Density
Output Set: Case 1 Design Cycle 837
Elemental Contour: Normalized Mass Density
Output Set: MSC/NASTRAN Case 1
Elemental Contour: Normalized Mass Density
Output Set: Case 1 Design Cycle 780
Elemental Contour: Normalized Mass Density
1.
0.929
0.857
0.786
0.714
0.643
0.571
0.5
0.429
0.357
0.286
0.214
0.143
0.0714
0.
Y
X

Figure 5-24. Simcenter Nastran topology optimization results for the 2D MBB compliance using linear and parabolic quadrilateral elements, with and without the checkerboard control (CHBC) toggle enabled.

5.6. Commercial Software Results: 3D MBB

The goal of showing numeric examples of the 3D MBB is primarily to provide recommendations for end-user engineers who intend to use topology optimization in practice. Specifically, these numeric examples were meant to provide recommendations regarding element type and the efficacy of filtering algorithms. The 3D MBB problem was evaluated using Simcenter Nastran 2019.2 and TOSCA Structure 2019 using Abaqus 3DEXPERIENCE R2019X for FEA.

It is understood that many differences between the two solver types will exist due to details in the implementation of each optimization algorithm, as well as differences in finite element formulations, memory settings, etc. However, effort was taken to facilitate as accurate a comparison between the two solvers as reasonably possible. For each solver, the compliance minimization problem is solved subject to a volume constraint equal to 20% of the initial design domain volume. Because the design variables in Simcenter Nastran are initialized to values of one, the TOSCA Structure default initialization value was overridden to values of one (the default is to initialize the design variables so that the volume constraint is satisfied). For both solvers, the available solution CPU was set to 16 cores and the solution memory (RAM) was set to 58 GB. However, the ways in which Simcenter Nastran and Abaqus use memory differ and computational performance is expected to differ even though consistent CPU and memory settings were enforced. Finally, the convergence criteria for each solver were set to be nearly identical. In Simcenter Nastran, the default convergence occurs when the objective function changes by less than 0.0001 and the design variables change by less than 0.001. The TOSCA Structure default objective and design variable delta tolerances were overridden to match the Simcenter Nastran default values. For each solver, eight numeric test cases were evaluated using different finite element types, different checkerboard control toggle options, or different optimization algorithms.

For the Simcenter Nastran test cases, the 3D MBB was modeled with linear hexahedral, parabolic hexahedral, linear tetrahedral, and parabolic tetrahedral elements using Nastran CHEXA (8-noded), CHEXA (20-noded) CTETRA (4-noded), and CTETRA (10-noded) element types, respectively. The SPOT optimizer was selected for all test cases (this is the default for Simcenter Nastran provided the number of constraints is small). The Simcenter Nastran checkerboard control option was toggled from the default value (on) to off for half of the test cases.

For the TOSCA Structure test cases, linear hexahedral, parabolic hexahedral, linear tetrahedral, and parabolic tetrahedral elements were used to model the 3D MBB using Abaqus C3D8I, C3D20R, C3D4, and C3D10M element types, respectively. In TOSCA Structure, the checkerboard control algorithm is always on. Instead of toggling the checkerboard control, one half of the TOSCA Structure test cases use the “General Sensitivity” optimization algorithm while the other half uses the “Controller-based” algorithm.

A summary of the results of the 3D MBB compliance problem using Simcenter Nastran and TOSCA Structure is shown in Table 5-3; The results of this study may be used to derive several general conclusions for each solver, as described in the following subsections.

Table 5-3. Summary of topology optimization results for the 3D MBB compliance minimization problem using Simcenter Nastran and TOSCA Structure with various element formulations, checkerboard control options, and optimization algorithms.

		Simcenter Nastran 2019.2								TOSCA Structure 2019 with Abaqus 3DEXPERIENCE R2019X							
	Element Type	CHEXA	CHEXA	CHEXA	CHEXA	CTETRA	CTETRA	CTETRA	CTETRA	C3D8I	C3D8I	C3D20R	C3D20R	C3D4	C3D4	C3D10M	C3D10M
	Includes Mid-Nodes	NO	NO	YES	YES	NO	NO	YES	YES	NO	NO	YES	YES	NO	NO	YES	YES
	Checkerboard Control	NO	YES	NO	YES	NO	YES	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES
	Optimization Algorithm "GS"=General Sensitivity "CBO"=Controller-based	SPOT	SPOT	SPOT	SPOT	SPOT	SPOT	SPOT	SPOT	GS	CBO	GS	CBO	GS	CBO	GS	CBO
	Total Solution Time [hrs]	2.2	2.2	16.3	15.4	5.3	5.1	4.6	4.8	4.6	0.5	14.8	4.5	2.9	0.5	17.5	2.7
	Initial Objective	27.4	27.4	28.3	28.3	27.0	27.0	28.3	28.3	27.4	27.4	28.6	28.6	27.0	27.0	29.1	28.9
	Final Objective	105.2	104.7	107.0	107.1	87.1	93.3	110.5	112.3	89.8	86.8	93.0	93.8	97.8	85.6	103.3	91.9
	Number of DOF	6.25E+05	6.25E+05	2.45E+06	2.45E+06	3.35E+05	3.35E+05	2.49E+06	2.49E+06	6.25E+05	6.25E+05	2.45E+06	2.45E+06	3.35E+05	3.35E+05	2.49E+06	2.49E+06
	Final Constraint	2.5E-03	2.5E-03	2.5E-03	2.5E-03	2.5E-03	2.5E-03	2.5E-03	2.5E-03	-4.2E-04	5.9E-02	-4.1E-04	5.8E-02	-9.8E-04	5.8E-02	-1.1E-03	5.8E-02
	Problem Size (1k DOF)	6.25	6.25	24.50	24.50	3.35	3.35	24.89	24.89	6.25	6.25	24.50	24.50	3.35	3.35	24.89	24.89
	Solver CPU	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
	Available Solver Memory [GB]*	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58
	Iterations to reach Convergence	79	73	72	75	500	500	72	73	165	15	149	15	130	15	109	15
	Time per Iteration [min]	1.7	1.8	13.6	12.3	0.6	0.6	3.8	4.0	1.7	2.2	6.0	17.9	1.3	1.9	9.6	10.9
	Improvement Ratio (Initial / Final Objective)	0.261	0.262	0.264	0.264	0.310	0.290	0.256	0.252	0.305	0.316	0.308	0.305	0.277	0.316	0.282	0.314
Subjective Evalutations	**Presence of Checkerboarding 1=None, 3=Visible, 5=High**	5	2	1	1	5	3	1	1	1	1	1	1	1	1	1	1
	Presence of Intermediate Densities 1=None, 3=Visible, 5=High	1	2	2	2	2	2	1	1	3	1	3	1	3	1	3	1
	Overall Topology Clarity 1=Clear, 3=Poor, 5=Very Poor	1	1	1	1	5	5	4	3	1	1	1	1	1	1	1	1

*Simcenter Nastran pre-allocates the memory shown above. Abaqus does not pre-allocate memory, and uses a percentage of the available memory shown above as-needed.

Best Average Worst

5.6.1. Simcenter Nastran

The topology optimization results for Simcenter Nastran test case for the 3D MBB compliance problem are shown in Figure 5-26 and Figure 5-27. In each figure, elements with pseudo-density values greater than 0.1 are hidden for clarity. Together with Table 5-3, these results may be used to formulate several general conclusions and best practices using Simcenter Nastran topology optimization.

First, without the checkerboard control algorithm enabled, the checkerboarding problem is prevalent when either linear hexahedral or tetrahedral elements are used. In the case of linear hexahedral elements, the presence of checkerboarding appears to have greatly increased the number of iterations required to converge. When parabolic element formulations are used, the presence of checkerboarding is significantly lower. Nonetheless, the use of the checkerboard control algorithm impacts the topology optimization results even when parabolic elements are used—a finding is consistent with observations from the 2D MBB case. For all test cases, Simcenter Nastran produces topologies with very few, if any, elements containing intermediate density values.

Next, regardless of whether the checkerboard control algorithm is enabled, the use of linear tetrahedral elements produces topologies with no distinct topological features. Indistinct topologies are also present when parabolic tetrahedral elements are used without the checkerboard control algorithm enabled. However, when the checkerboard control algorithm is enabled, the final topology produced from the parabolic tetrahedral mesh is much cleaner and begins to resemble results produced using hexahedral elements. It is unknown why there is an apparent limitation to using linear tetrahedral elements to solve the compliance minimization problem in Simcenter Nastran, but it may be related to the checkerboard control algorithm.

Like the 2D MBB problem, the use of the checkerboard control algorithm results in “jumpy” constraint iteration behavior. This behavior may be related to the heuristic nature of the checkerboard control algorithm, and perhaps may be a result of enforcing a means to correct for checkerboarding on a localized scale. Figure 5-25 shows comparisons of the constraint iteration history for Simcenter Nastran for all test cases evaluated. Clearly, when the checkerboard control algorithm is disabled, the volume constraint remains constant and equal to the solver constraint tolerance value, as expected. However, when the checkerboard control is enabled, the “jumpy” constrain behavior may be seen. Simcenter Nastran aims only to satisfy the constraint within a given tolerance (the default value is +2.5E–3), which means that the final design violates the constraint slightly.

Finally, the use of parabolic hexahedral elements incurred a significant computational cost for the optimization without adding a significant benefit when compared to using linear hexahedral elements with the checkerboard control algorithm enabled. For these reasons, it is recommended that for Simcenter Nastran topology optimization of 3D problems, either linear hexahedral or parabolic tetrahedral elements be used in conjunction with the checkerboard control algorithm.

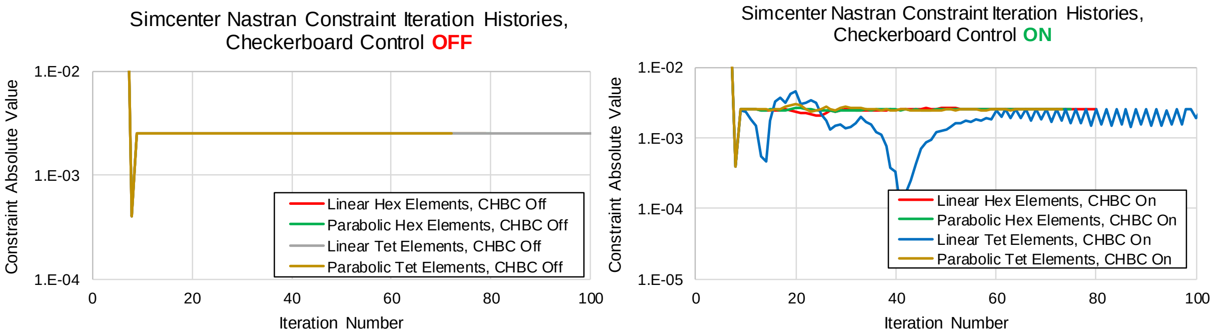
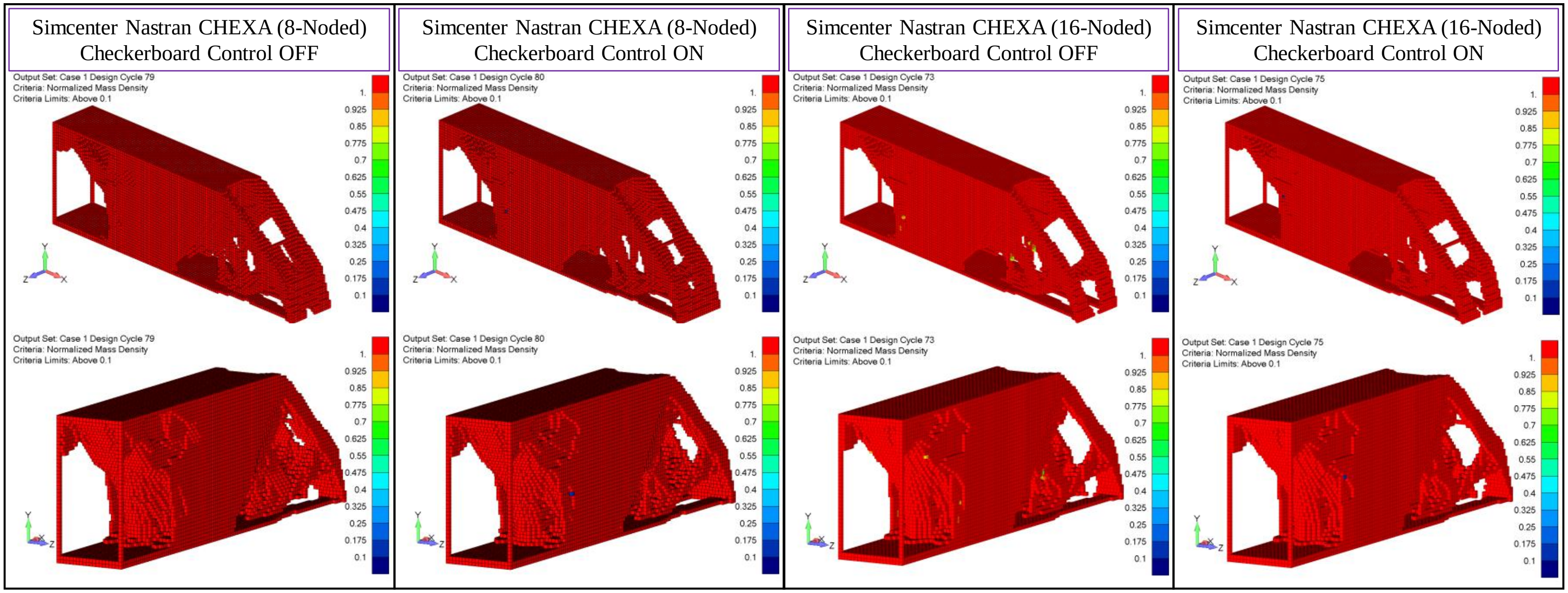


Figure 5-25. Comparisons of Simcenter Nastran constraint iteration histories with and without the checkerboard control algorithm enabled.



Simcenter Nastran CHEXA (8-Noded) Checkerboard Control OFF

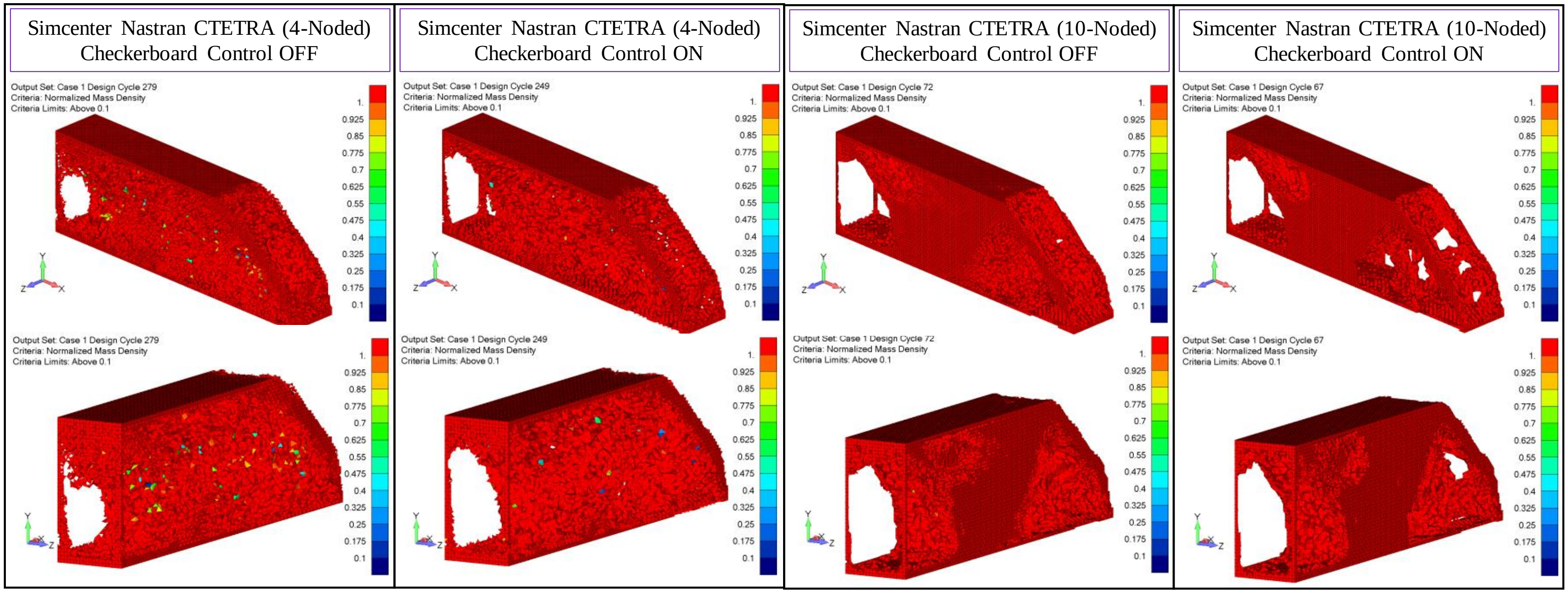
Simcenter Nastran CHEXA (8-Noded) Checkerboard Control ON

Simcenter Nastran CHEXA (16-Noded) Checkerboard Control OFF

Simcenter Nastran CHEXA (16-Noded) Checkerboard Control ON

Output Set: Case 1 Design Cycle 79
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
Output Set: Case 1 Design Cycle 80
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
Output Set: Case 1 Design Cycle 73
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
Output Set: Case 1 Design Cycle 75
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
1.
0.925
0.85
0.775
0.7
0.625
0.55
0.475
0.4
0.325
0.25
0.175
0.1
Y
Z
X

Figure 5-26. Topology optimization results for the 3D using Simcenter. (Left): Linear hexahedral (CHEXA, 8-noded)) elements with and without the checkerboard control algorithm enabled. (Right): Parabolic hexahedral (CHEXA, 20-noded) elements with and without the checkerboard control algorithm enabled. Note that by default, Simcenter Nastran results are output at the “best” design cycle, which is not necessarily the final design iteration.



Simcenter Nastran CTETRA (4-Noded) Checkerboard Control OFF

Simcenter Nastran CTETRA (4-Noded) Checkerboard Control ON

Simcenter Nastran CTETRA (10-Noded) Checkerboard Control OFF

Simcenter Nastran CTETRA (10-Noded) Checkerboard Control ON

Output Set: Case 1 Design Cycle 279
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
Output Set: Case 1 Design Cycle 249
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
Output Set: Case 1 Design Cycle 72
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
Output Set: Case 1 Design Cycle 67
Criteria: Normalized Mass Density
Criteria Limits: Above 0.1
1.
0.925
0.85
0.775
0.7
0.625
0.55
0.475
0.4
0.325
0.25
0.175
0.1
Y
Z
X

Figure 5-27. Topology optimization results for the 3D using Simcenter. (Left): Linear tetrahedral (CTETRA, 4-noded)) elements with and without the checkerboard control algorithm enabled. (Right): Parabolic tetrahedral (CTETRA, 10-noded) elements with and without the checkerboard control algorithm enabled. Note that by default, Simcenter Nastran results are output at the “best” design cycle, which is not necessarily the final design iteration.

5.6.2. TOSCA Structure

The topology optimization results for TOSCA Structure test cases for the 3D MBB compliance problem are shown in Figure 5-28 and Figure 5-29. In each figure, elements with pseudo-density values greater than 0.1 are hidden for clarity. Together with Table 5-3, these results may be used to formulate several general conclusions and best practices using TOSCA Structure for topology optimization.

First, TOSCA Structure examples that use the general sensitivity algorithm contain more elements with intermediate densities compared to the Simcenter Nastran solution. The presence of regions of intermediate densities is very likely related to the filtering algorithm implemented in TOSCA Structure. As expected, the use of a sensitivity or linear density filter introduces “gray” regions into the final topology. When using the controller-based optimizer, however, no intermediate densities are present throughout the design domain.

Next, all element types evaluated appear to be a valid choice for topology optimization. For the general sensitivity algorithm, the element choice appears to impact the final topologies and produce somewhat different final objective function values. The difference in behavior between element types may be a result of the optimizer converging to slightly different local optima because of differences in the finite element stiffness formulation. For all test cases, the general sensitivity algorithm satisfies the volume constraint. The fact that each element type exhibits overall favorable convergence behavior is desirable. Usually, the choice of finite element type is dictated by the problem geometry or computational resources. Therefore, the fact that TOSCA Structure is apparently not as limited by the finite element choice, as Simcenter Nastran, is positive. Interestingly, the topologies produced using tetrahedral elements appear to exhibit asymmetry. It seems likely this may be a result of the problem setup, and not an artifact of the TOSCA Structure topology optimization. However, in the event it is related to the element type, it is therefore recommended that a planar symmetry manufacturing constraint be included in the optimization process for topology problems that include tetrahedral design domains. The designs produced by the controller-based optimization tend to resemble results from the 2D MBB.

Compared to Simcenter Nastran, overall the number of iterations required for the TOSCA Structure general sensitivity optimization algorithm to converge is lower. Both the general-sensitivity and controller-based optimization algorithms require a similar amount of times per iteration. The TOSCA Structure time per iteration is also like Simcenter Nastran for all element formulations except for tetrahedral elements with mid-side nodes, for which the Abaqus FEA seems to take longer. However, the fact that the controller-based optimization converges in 15 iterations means that the overall computational cost for solving the 3D compliance minimization problem is drastically reduced when using the controller-based optimization.

Finally, regardless of the choice of optimizer or finite element type, TOSCA Structure tended to find a lower objective than Simcenter Nastran. For all these reasons, it is clear the that TOSCA Structure controller-based algorithm provides the most robust and effective optimizer choice for compliance minimization problems. The benefits of producing topology results with no checkerboarding, no intermediate densities, and with such low computational overhead cannot be overstated. However, the controller-based optimizer is limited to compliance minimization problems. Furthermore, because the controller-based optimizer calculates nodal stresses for each design cycle, the per iteration computation time can be larger than the general sensitivity algorithm for higher-order elements, such as C3D20R.

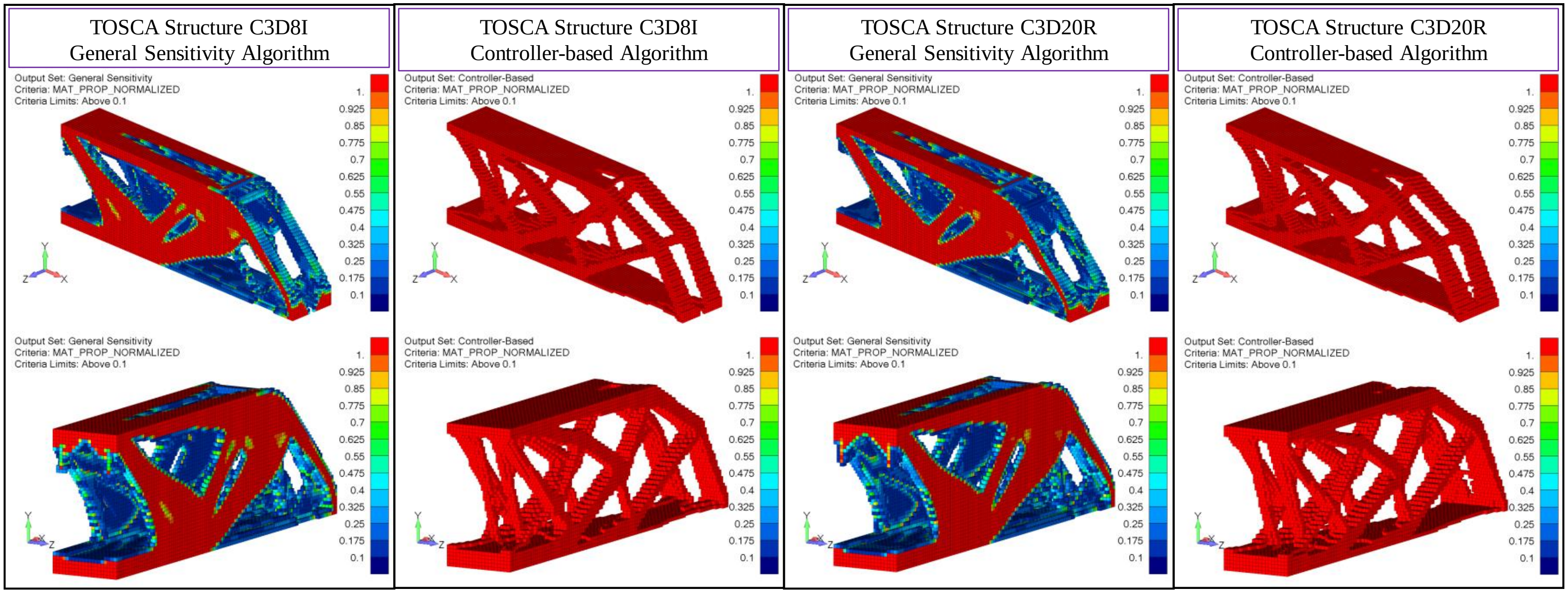
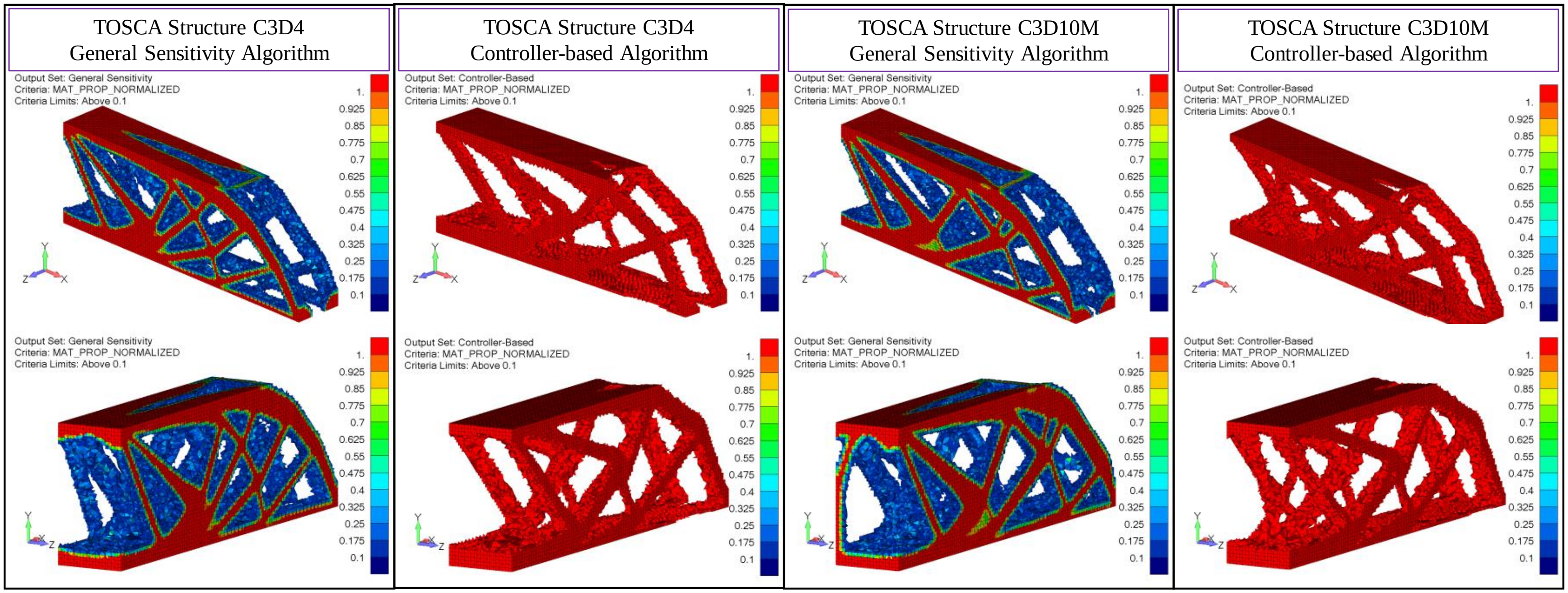

TOSCA Structure C3D8I
General Sensitivity Algorithm
TOSCA Structure C3D8I
Controller-based Algorithm
TOSCA Structure C3D20R
General Sensitivity Algorithm
TOSCA Structure C3D20R
Controller-based Algorithm
Output Set: General Sensitivity
Output Set: Controller-Based
Criteria: MAT_PROP_NORMALIZED
Criteria Limits: Above 0.1
1.
0.925
0.85
0.775
0.7
0.625
0.55
0.475
0.4
0.325
0.25
0.175
0.1
Y
X
Z

Figure 5-28. Topology optimization results for the 3D using TOSCA Structure. (Left): Linear hexahedral (C3D8I) elements with the general sensitivity and controller-based optimization algorithms. (Right): Parabolic hexahedral (C3D20R) elements with the general sensitivity and controller-based optimization algorithms.



TOSCA Structure C3D4
General Sensitivity Algorithm

TOSCA Structure C3D4
Controller-based Algorithm

TOSCA Structure C3D10M
General Sensitivity Algorithm

TOSCA Structure C3D10M
Controller-based Algorithm

Output Set: General Sensitivity
Criteria: MAT_PROP_NORMALIZED
Criteria Limits: Above 0.1
Output Set: Controller-Based
Criteria: MAT_PROP_NORMALIZED
Criteria Limits: Above 0.1
1.
0.925
0.85
0.775
0.7
0.625
0.55
0.475
0.4
0.325
0.25
0.175
0.1
Y
Z
X

Figure 5-29. Topology optimization results for the 3D using TOSCA Structure. (Left): Linear tetrahedral (C3D4) elements with the general sensitivity and controller-based optimization algorithms. (Right): Parabolic tetrahedral (C3D10M) elements with the general sensitivity and controller-based optimization algorithms.

6. Conclusions and Recommendations

Conclusions and recommendations regarding filtering settings, as well as general recommendations regarding topology optimization in the four commercial software discussed in this paper are presented in this section.

6.1. Filter Settings Recommendations

All filter parameters assessed in this paper are effective at eliminating the checkerboarding problem. In general, the use of conic weighting matrices results in lower objective values and fewer intermediate density elements at little to no extra computational expense compared to a constant weighting scheme. Therefore, it is recommended to use a weighting matrix that accounts for element distances, such as the conic weighting matrix discussed in this paper.

Any filter radius that is large enough to encompass the elements immediately adjacent to a given element will eliminate the checkerboarding problem. Larger filter radii may be used to introduce a mesh-independent length scale into the optimization problem and serve to enforce minimum member sizes. However, excessively large filter radii result in an increasingly large number of intermediate design variable elements with a progressively less complex final design. Eventually, the filter radius may become so large as to effectively average out the entire structure. For these reasons, in most cases it is recommended to use the minimal possible filtering radius. Other means to control minimum member sizes via manufacturing constraints are implemented in commercial software such as Simcenter Nastran, TOSCA Structure, and ANSYS, and are recommended if the intent is to enforce a minimal structural member size. Minimum member size manufacturing constraints were not evaluated in this paper.

Finally, both sensitivity filters and linear density filters have been proven to be effective at eliminating the checkerboarding problem and producing realizable designs with overall similar topologies and final objective values. The principal drawbacks of sensitivity filtering are that the heuristically modified sensitivities may become problematic for math programming-based optimizers and result in inaccurate estimates for the Lagrange multipliers. Therefore, convergence for math programming-based optimizers that use a sensitivity filter will always be based on the relatively weak criteria of changes to the objective and design variables.

Convergence based on the first-order gradient condition was observed when OC was applied in conjunction with a sensitivity filter.

Linear density filters, on the other hand, are not heuristic and will generally provide more accurate estimates of the gradient of the Lagrangian, particularly when used in conjunction with a math programming optimizer. Linear density filters therefore can be used to achieve more robust convergence based on the first-order gradient condition of the Lagrangian function. However, the linear density filter typically requires more iterations to achieve convergence compared to the sensitivity filter.

The modifications to the OC criteria to account for the linear density filter presented in this paper provide a robust alternative to sensitivity filtering. However, further work is required to investigate the method’s inability to achieve convergence based on the first-order Lagrangian gradient condition.

6.2. Commercial Software Recommendations

The commercial software programs evaluated in this paper make use of an optimizer based either on MMA/GCMMA, interior-point, or on OC methods. Each piece of software includes a method to control the checkerboarding effect, but the ability to toggle or control these methods varies. The default settings provide an appropriate method for controlling the checkerboarding effect for all software. Based on comparisons to the MATLAB GCMMA implementation and the availability of convergence criteria within each commercial software, it was postulated that most commercial software makes use of sensitivity filtering in their topology optimization algorithms. Simcenter Nastran appears to be the only outlier with regards to its filtering implementation. Its checkerboard control option often results in “jumpy” behavior of the constraint and/or design variable values, which may be attributed to a localized nature of its filtering method, but the exact cause is unknown. Final topologies produced by Simcenter Nastran, on average, also tend to be outliers in comparison to topologies produced by the other commercial software evaluated.

In general, inclusion of a method to prevent checkerboarding is a necessity for every commercial software, regardless of the element formulation employed. While using parabolic element formulations that include midside-nodes partially alleviates the need to implement a checkerboard control method, both 2D and 3D parabolic element meshes still tend to benefit from filtering algorithms.

The choice of a penalty parameter value will often be problem specific. In most cases, a relatively low penalty factor (e.g., SIMP $1 > p > 3$) will often converge to a better overall objective, but at the cost of a high number of required iterations and a large fraction of intermediate density design variables. Relatively high penalty factors (e.g., SIMP $p > 3.5$) will increase convergence speed and decrease the fraction of elements with intermediate densities, but oftentimes will converge to local optima with overall worse objective values. If possible, it is recommended that multiple penalty parameters be assessed for the same problem. In software such as TOSCA Structure, it may also be prudent to use a continuation approach for the penalty factor. That is, perform on topology optimization using a relatively low penalty value, then use the results from that optimization as the initial densities for a new topology optimization using a higher penalty value. The choice of RAMP over SIMP material penalization is generally recommended for problems that include dynamics or self-weight.

The choice of element formulation may be dictated by the specific problem geometry and available software for meshing. However, when possible, it is recommended that linear quadrilateral or linear hexahedral elements with a regular mesh density be used for solving topology optimization problems. Parabolic quadrilateral or hexahedral elements are another option and may be used in the absence of a filtering algorithm but will exhibit some degree of checkerboarding in most cases, and typically incur a large computational cost. Triangular or tetrahedral elements may also be used in conjunction with a filtering algorithm to produce useable topology results. However, it is recommended that, if tetrahedral elements are required, the overall mesh density ought to be increased to compensate for the fact that the element boundaries are often jagged and difficult to distinguish. It was determined that linear triangular or tetrahedral elements should be avoided in Simcenter Nastran topology optimization, even if a filtering algorithm is employed.

All four commercial software are effective at solving the minimum compliance problem evaluated in this paper. However, some software allows more control for the user or better overall results compared others. A summary of the author’s opinions regarding the use of each software follows.

ANSYS provides a robust topology optimization program that is easy to use and quite robust. Its principal drawbacks are an overall lack of control in terms of optimization parameters, filtering type, and output control. For example, the author was unable to determine how the elemental densities might be plotted as “raw” results, instead of the grey-to-red interpolated values output by ANSYS workbench. Furthermore, ANSYS requires CAD definition of the design domain, which may be problematic in some situations.

TOSCA Structure was determined to be the most robust and reliable commercial software for topology optimization that was evaluated. The ability of TOSCA Structure to couple with multiple FEA programs is inherently valuable. The filtering algorithm implemented in TOSCA Structure also appears to be very robust and reliable. Furthermore, the author found that the “controller-based” optimizer worked particularly well for the compliance minimization example problems using a single static load case and is excellent at scaling up with problem size. For compliance minimization problems with many elements, it is strongly recommended that TOSCA Structure’s controller-based optimization algorithm be used, as it has both excellent scaling properties and produces topological results with zero intermediate densities.

MSC Nastran is valuable in that includes both robust and well-documented control for the filtering algorithm and optimization algorithms. Nevertheless, the inability to override the “sufficiently black and white” logic of MSC Nastran is a major hindrance to effectively using the software. Furthermore, while MSC Nastran includes a means to enforce minimum member size via a filtering algorithm, it lacks the explicit minimum member size manufacturing constraint present in each other software evaluated.

Simcenter Nastran, though relatively new in terms of development, was shown to have robust topology optimization capabilities. However, Simcenter Nastran topology is prone to several drawbacks that were not observed in other commercial software. The current checkerboard control algorithm exhibits some strange behavior, but it is nonetheless effective at addressing the checkerboard problem. Finally, Simcenter Nastran topology optimization tended to lag programs like TOSCA Structure’s general sensitivity algorithm in terms of the number of iterations required for convergence, although it is unknown why.

References

[1] Sigmund, O., & Petersson, J. (1998). Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. Structural Optimization, 16(1), 68–75. doi: 10.1007/bf01214002

[2] Bendsøe, M. P., & Sigmund, O. (2003). Topology optimization: theory, methods and applications. Berlin: Springer.

[3] Bendsøe, M. P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. Computer Methods in Applied Mechanics and Engineering, 71(2), 197–224. doi: 10.1016/0045-7825(88)90086-2

[4] Sigmund, O. (2001). A 99 line topology optimization code written in MATLAB. Structural and Multidisciplinary Optimization, 21(2), 120–127. doi: 10.1007/s001580050176

[5] Stolpe, M., & Svanberg, K. (2001). An alternative interpolation scheme for minimum compliance topology optimization. Structural and Multidisciplinary Optimization, 22(2), 116–124. doi: 10.1007/s001580100129

[6] Deaton, J. D., & Grandhi, R. V. (2013). A survey of structural and multidisciplinary continuum topology optimization: post 2000. Structural and Multidisciplinary Optimization, 49(1), 1–38. doi: 10.1007/s00158-013-0956-z

[7] Simcenter Nastran 2019.2 Quick Reference Guide. (2019). Siemens Product Lifecycle Management Software, Inc.

[8] Siemens NX 12.0 Documentation. (2019) Siemens Product Lifecycle Management Software, Inc.

[9] Haftka, R. T., & Gürdal Zafer. (1992). Elements of structural optimization. Dordrecht: Kluwer.

[10] Díaz, A., & Sigmund, O. (1995). Checkerboard patterns in layout optimization. Structural Optimization, 10(1), 40–45. doi: 10.1007/bf01743693

[11] Jog, C. S., & Haber, R. B. (1996). Stability of finite element models for distributed-parameter optimization and topology design. Computer Methods in Applied Mechanics and Engineering, 130(3-4), 203–226. doi: 10.1016/0045-7825(95)00928-0

[12] Petersson, J., & Sigmund, O. (1998). Slope constrained topology optimization. International Journal for Numerical Methods in Engineering, 41(8), 1417–1434. doi: 10.1002/(sici)1097-0207(19980430)41:8<1417::aid-nme344>3.0.co;2-n

[13] Sigmund, O. (2007). Morphology-based black and white filters for topology optimization. Structural and Multidisciplinary Optimization, 33(4-5), 401–424. doi: 10.1007/s00158-006-0087-x

[14] Svanberg, K., & Svärd, H. (2013). Density filters for topology optimization based on the Pythagorean means. Structural and Multidisciplinary Optimization, 48(5), 859–875. doi: 10.1007/s00158-013-0938-1

[15] Ambrosio, L., & Buttazzo, G. (1993). An optimal design problem with perimeter penalization. Calculus of Variations and Partial Differential Equations, 1(1), 55–69. doi: 10.1007/bf02163264

[16] Borrvall, T. (2001). Topology optimization of elastic continua using restriction. Archives of Computational Methods in Engineering, 8(4), 351–385. doi: 10.1007/bf02743737

[17] Bruns, T. E., & Tortorelli, D. A. (2001). Topology optimization of non-linear elastic structures and compliant mechanisms. Computer Methods in Applied Mechanics and Engineering, 190(26-27), 3443–3459. doi: 10.1016/s0045-7825(00)00278-4

[18] Wang & Wang (2005) (see filter papers)

[19] Sigmund, O. (2001). Design of multiphysics actuators using topology optimization – Part II: Two-material structures. Computer Methods in Applied Mechanics and Engineering, 190(49-50), 6605–6627. doi: 10.1016/s0045-7825(01)00252-3

[20] Svanberg, K. (2002). A Class of Globally Convergent Optimization Methods Based on Conservative Convex Separable Approximations. SIAM Journal on Optimization, 12(2), 555–573. doi: 10.1137/s1052623499362822

[21] Venkayya, V. (1971). Design of optimum structures. Computers & Structures, 1(1-2), 265–309. doi: 10.1016/0045-7949(71)90013-7

[22] Levy, R. (1991). Fixed Point Theory and Structural Optimization. Engineering Optimization, 17(4), 251–261. doi: 10.1080/03052159108941074Feury, C., & Geradin, M. (1978). Optimality criteria and mathematical programming in structural weight optimization. Computers & Structures, 8(1), 7–17. doi: 10.1016/0045-7949(78)90155-4

[23] Olhoff, N., Bendsøe, M. P., & Rasmussen, J. (1991). On CAD-integrated structural topology and design optimization. Computer Methods in Applied Mechanics and Engineering, 89(1-3), 259–279. doi: 10.1016/0045-7825(91)90044-7

[24] Canfield, Robert (2013). MATLAB code for Finite Element Analysis.

[25] Lazarov, B. S., Wang, F., & Sigmund, O. (2016). Length scale and manufacturability in density-based topology optimization. Archive of Applied Mechanics, 86(1-2), 189–218. doi: 10.1007/s00419-015-1106-4

[26] TOSCA Structure Documentation. (2019) Dassault Systèmes, Providence, RI, USA.