On multigrid-CG for efficient topology optimization

Oded Amir[1], Niels Aage[2] and Boyan S. Lazarov[2]

[1] Faculty of Civil and Environmental Engineering, Technion - Israel Institute of Technology

[2] Department of Mechanical Engineering, Technical University of Denmark

1 Abstract

This article presents a computational approach that facilitates the efficient solution of 3-D structural topology optimization problems on a standard PC. Computing time associated with solving the nested analysis problem is reduced significantly in comparison to other existing approaches. The cost reduction is obtained by exploiting specific characteristics of a multigrid preconditioned conjugate gradients (MGCG) solver. In particular, the number of MGCG iterations is reduced by relating it to the geometric parameters of the problem. At the same time, accurate outcome of the optimization process is ensured by linking the required accuracy of the design sensitivities to the progress of optimization. The applicability of the proposed procedure is demonstrated on several 2-D and 3-D examples involving up to hundreds of thousands of degrees of freedom. Implemented in MATLAB, the MGCG-based program solves 3-D topology optimization problems in a matter of minutes. This paves the way for efficient implementations in computational environments that do not enjoy the benefits of high performance computing, such as applications on mobile devices and plug-ins for modeling software.

Keywords: Topology optimization, preconditioned conjugate gradients, multigrid.

2 Introduction

In typical structural topology optimization procedures, the computational effort involved in repeated solution of the analysis equations dominates the computational cost of the whole process. This motivates the search for efficient approaches aimed at reducing the computational effort invested in the analysis. The current contribution aims primarily at deriving computational procedures for performing 3-D topology optimization on a standard PC within high-level programming environments such as MATLAB. To ensure that the approach can be easily ported to other programming environments, the suggested code is based almost entirely on the SuiteSparse library [18]. This general approach paves the way for integrating 3-D topology optimization in CAD software, e.g. within the Grasshopper-Rhino TopOpt component released recently; as well as in applications for mobile devices such as the TopOpt app [2]. Even though large-scale parallel procedures are not of primary concern in this work, the general form of the iterative solver presented here is indeed also applicable to large-scale topology optimization problems.

Reduction of computational effort in topology optimization has been approached from various standpoints over the past few years. One of the major research trajectories examines the application of multiple computational scales and resolutions, thus avoiding the inherent high cost of solving sequences of finite element analyses on a fine mesh. Kim and Yoon [25] suggested the concept of multi-resolution multi-scale topology optimization (MTOP) where optimization is performed while progressively refining the grid. They used a wavelet-based parametrization of the density distribution, as was also suggested by Poulsen [30] in the context of regularizing topological design. The utilization of wavelet bases was later extended by Kim et al. [23] who applied adaptive wavelet-Galerkin analysis to multi-scale topology optimization. Stainko [35] proposed a multilevel scheme where optimization is first performed on a coarse grid, which is then adaptively refined in the interface between solid and void. Ultimately, a fine-resolution design is achieved for a relatively low computational expense. Using this strategy, it is not clear whether fine features that did not exist in the coarse design can emerge in the finer levels. More recently, another multi-resolution

topology optimization scheme was suggested, where the density distribution is realized on a finer scale than the displacements [28, 29].

Other approaches for saving computational cost include reducing the number of design variables by using adaptive design variable fields [21]; generating sets of Pareto-optimal topologies within a single optimization process [37]; and adaptive reduction of the number of design variables based on the history of each variable during optimization [24]. However, the current study is mostly related to contributions focusing solely on reducing the cost involved in repeated solution of the nested analysis equations, either by recycling of Krylov subspaces [42]; integrating approximate reanalysis [4, 13, 44, 6]; and truncating iterative solutions [5, 3].

It was observed in some of the recent investigations that accurate results of the optimization process can be achieved even if the analysis equations are not solved to full accuracy [e.g. 42, 5, 3]. This observation is based primarily on numerical experimentation and providing comprehensive mathematical argumentation is still a challenge for future research. Results presented here demonstrate again that approximate solution of the analysis equations can yield accurate outcome of the optimization. Several advancements with respect to previous studies are presented, that result in significant reduction in computational effort. We continue to use the framework of Krylov subspace iterative solvers, particularly the Preconditioned Conjugate Gradients method (PCG), but with a multigrid V-cycle applied as a preconditioner. The resulting procedure, typically termed MGCG (MultiGrid Conjugate Gradients) is nowadays a well-established technique and was shown to provide mesh-independent convergence as well as good parallel scalability [39, 8].

This article focuses on the effective utilization of MGCG for the purpose of solving the nested analysis equations arising in topology optimization problems, a topic which has yet to be explored. It is shown that straightforward implementation of MGCG indeed leads to better performance compared to other preconditioning techniques. More importantly, further reduction in computational cost is possible by exploiting specific characteristics of MGCG and by linking the accuracy of the MGCG analysis to the progress of optimization. We show that MGCG is unique due to the relations between the required number of iterations and the geometric parameters of the problem - namely the number of grid levels and the filter size. Furthermore, we propose to relate the tolerance of the iterative solution of the analysis equations to the progress of optimization. This means that the analysis is solved to full accuracy only when the optimization problem approaches a solution, whereas approximations are sufficient in earlier stages as long as they do not hamper the progress towards an optimum. In other words, a 'semi-nested' approach is taken: The analysis is indeed solved separately from the optimization but the effort invested in it, and consequently the accuracy achieved, are both linked to the progress of optimization.

Up to date, we are not aware of any publications concerning the utilization of MGCG for solving the analysis equations arising in topology optimization problems. This might be related to the expected difficulty of multigrid methods to deal with high contrast in coefficients - an inherent property due to the desired solid-void (or 1-0) density distribution. Nevertheless, multigrid solvers were proposed for solving the complete Karush-Kuhn-Tucker (KKT) system of equations corresponding to topology optimization problems [27, 36]. Another application arises when following the phase field approach to topology optimization, where a multigrid method was utilized for solving the Cahn-Hilliard equations [43].

Another alternative for improving the solution timing is the employment of parallel computing for solving both the state problem as well as the optimization [1]. Such a parallelization approach leads to very high parallel efficiency, i.e., good utilization of a large-scale parallel computer. However, good scalability is irrelevant if the solution method is very slow. Our main interest is in decreasing the total time for a given set of computational resources. Therefore, multigrid solvers are excellent candidates as they are parallelizable and numerically scalable, i.e., the solution time is proportional to the problem size. Outline of the main challenges for multigrid parallelization can be found in [17] and recent reports on solving very large-scale problems on one of the fastest supercomputers can be found in [9, 10]. In the current study, the main focus is on demonstrating and discussing the applicability of MGCG as a solver of the linear analysis equations in topology optimization. Reporting the challenges of parallel implementation is left for future work.

The article is organized as follows. The optimization problem formulation and standard implementation of the multigrid-CG procedure are briefly discussed in Section 3. The heart of the article is in Section 4, where observations based on numerical experimentation and the suggested approximate approach are presented. This is followed by several examples involving 3-D optimized designs in Section 5. Finally, discussion and conclusions are given in Section 6.

3 Problem formulation and computational framework

For demonstrative purposes we address classical problems in structural topology optimization involving linear elasticity, namely minimum compliance and force inverter problems. These are hereby reviewed briefly for the sake of completeness. We follow the density-based approach [11] and apply the modified SIMP interpolation scheme relating density to elastic stiffness [34]. This does not imply any loss of generality - various topology optimization approaches share the same basic formulation and can equally benefit from the advancements discussed in this article. The considered optimization problem has the following generic form

$$\begin{aligned} \min_{\boldsymbol{\rho}} \phi(\boldsymbol{\rho}) \quad &= \quad \mathbf{l}^{\mathsf{T}}\mathbf{u} \\ \text{s.t.:} \quad & \sum_{e=1}^{N} v_e \rho_e \leq V \\ & 0 \leq \rho_e \leq 1 \qquad e = 1, ..., N \\ \text{with:} \quad & \mathbf{K}(\boldsymbol{\rho})\mathbf{u} = \mathbf{f} \end{aligned} \tag{1}$$

where $\boldsymbol{\rho}$ represents the density distribution; $\mathbf{K}(\boldsymbol{\rho})$ is the stiffness matrix; $\mathbf{f}$ is the load vector; $\mathbf{u}$ is the displacements vector; v_e is the element volume; and V is the total available volume. In each finite element the elastic stiffness is given by the modified SIMP interpolation rule

$$E(\rho) = E_{min} + (E_{max} - E_{min})\rho^p$$

where E_{max} is typically 1; E_{min} is a small number; and the penalty power p is typically set to 3. The ratio between E_{max} and E_{min} will be referred to later as the *contrast* of the layout because it controls the sharpness of the stiffness distribution for a given 0-1 topological layout.

Minimum compliance optimization is obtained by setting $\mathbf{l} \equiv \mathbf{f}$ while for the force inverter problem $\mathbf{l}$ is a vector with the value of 1 at the output degree of freedom and zeros otherwise. Design sensitivities are computed by the adjoint method. For both cases, the sensitivity of the objective with respect to a particular element density is given by

$$\frac{\partial \phi}{\partial \rho_e} = -\overline{\mathbf{u}}^{\mathsf{T}} \frac{\partial \mathbf{K}}{\partial \rho_e} \mathbf{u}$$

where $\mathbf{K}(\boldsymbol{\rho})\overline{\mathbf{u}} = \mathbf{l}$, thus for the compliance problem $\overline{\mathbf{u}} \equiv \mathbf{u}$. Using the computed gradients of the objective and the constraint, the problem can be solved by various means. We apply either an optimality criteria (OC) procedure or a first-order nonlinear programming method - MMA [38]. Mesh-independence and regularization are treated by either sensitivity filtering [32, 33] or density filtering [16, 14].

For demonstrating the performance of MGCG-based procedures we utilize three test cases: 1) Minimum compliance topology optimization of a beam with a point load, typically referred to as the MBB-beam, with various mesh resolutions; 2) Minimum compliance topology optimization of a cantilever beam, with a single point load at the middle of the free face; and 3) Maximum output displacement of a force inverter, see Figure 1. For the cantilever beam, The FE mesh resolution is 160×80; the volume fraction is 0.4; and the modified SIMP penalty is 3.0. The same mesh and penalty are used also for the force inverter, but with a volume fraction of 0.3. Within the multigrid V-cycle, we apply a single damped Jacobi smoothening cycle with $\omega = 0.8$. We use up to four multigrid levels: According to the specified number of levels, a direct solve is performed on either 80×40, 40×20 or 20×10 grid. With 3-D applications in mind, it is clear that the coarsest level should be chosen such that a direct solve can be performed efficiently.

3.1 Preconditioned Conjugate Gradient method

The focus here is on the effective solution of the linear system of equations $\mathbf{Ku} = \mathbf{f}$ arising from a finite element discretization of the state problem. The stiffness matrix is large, sparse and positive definite. The solution $\mathbf{u}$ can be obtained using either direct methods [e.g. 18] or iterative methods [31]. For full matrices, the computational cost of direct Cholesky decomposition is of $O\left(n^3\right)$, while for sparse matrices direct methods such as nested dissection require $O\left(n^{3/2}\right)$ operations, where n is the size of the stiffness matrix [18]. For small problems, particularly in 2-D, direct solvers

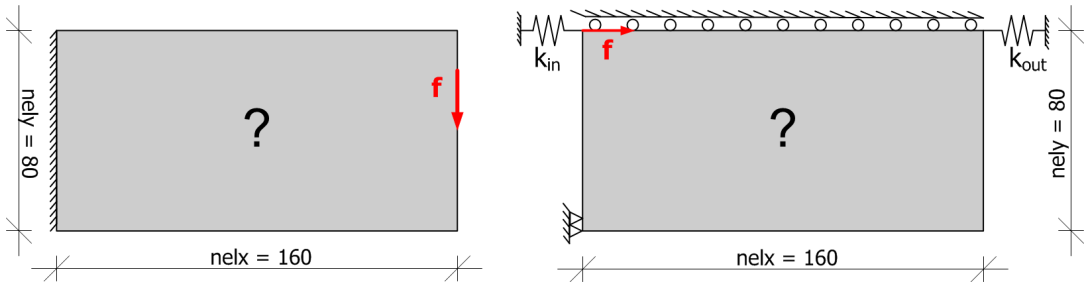

Figure 1: Demonstrative test cases. Left: Minimum compliance topology optimization of a 2-D cantilever beam. Right: Maximum output displacement of a 2-D force inverter.

are often preferable due to their robustness. However, solution time and memory requirements become prohibitive for 3-D problems as well as for large-scale 2-D. For these cases, the sparsity of the stiffness matrix results in an inexpensive matrix vector product operation which can be effectively utilized by iterative solution algorithms.

For positive definite system matrices the iterative method of choice is the classical Conjugate Gradient method [22]. It is a descent method which minimizes the functional $F(\mathbf{u}) = \|\mathbf{Ku}-\mathbf{f}\|_{\mathbf{K}^{-1}}$ and requires only a single matrix vector multiplication per iteration. Theoretically the method can find the solution in less than n iterations and the convergence rate is given by

$$\|\mathbf{u}-\mathbf{u}_k\|_{\mathbf{K}^{-1}} \leq \|\mathbf{u}-\mathbf{u}_0\|_{\mathbf{K}^{-1}} \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{2}$$

where κ is the condition number of the matrix $\mathbf{K}$ and k is the iteration number. For $\kappa >> 1$ the convergence is very slow. In practice convergence is ensured and accelerated by preconditioning the linear system with a matrix or an operator $\mathbf{M}$ for which $\kappa\left(\mathbf{M^{-1}K}\right) << \kappa(\mathbf{K})$. The construction of an effective preconditioner or the action of $\mathbf{M^{-1}}$, should cost as little as possible and the condition number $\kappa\left(\mathbf{M^{-1}K}\right)$ should be close to one and independent of n. Classical preconditioners such as incomplete Cholesky factorization, diagonal scaling or Factorized Sparse Approximate Inverses (FSAI) cannot provide a mesh independent convergence rate and often the preconditioned system is contrast-dependent. Therefore, their utilization should be limited in comparison to modern, numerically scalable multilevel/multigrid techniques [e.g. 41].

3.2 Multigrid method

Multigrid (MG) is a multilevel iterative method originally developed for solving discretized homogeneous elliptic problems. Nowadays it constitutes a family of methods that can be used to solve also inhomogeneous and non-elliptic partial differential equations. MG is considered numerically scalable as it can provide the solution of a linear system for a computational cost of $O(n)$. Such optimal performance is achieved by employing two complementary processes: Smoothing and coarse-grid correction. The smoothing process reduces the oscillatory error in the solution and is typically based on a stationary iterative method such as Gauss-Seidel or Jacobi. The smoother removes the high frequency components of the residual so that smoothed low order components can be effectively approximated on a coarser grid. The coarse grid correction procedure transfers information to a coarser grid through *restriction*, solves the coarsened system of equations and transfers back the solution to the fine mesh through *prolongation* (or *interpolation*). Coarse grid correction eliminates the slowly varying error components which cannot be eliminated by the smoothing operation.

The basic two-grid MG algorithm is shown in Algorithm 1. The prolongation $\mathbf{P} : \mathbb{R}^{\mathbf{n_c}} \rightarrow \mathbb{R}^{\mathbf{n}}$ maps the coarse grid solution to the fine grid and is represented by a $n \times n_c$ sparse matrix, where n_c is number of the degrees of freedom utilized for defining the solution on the coarse grid. The prolongation matrix is obtained by using standard finite element interpolation on the coarse mesh for determining the weight of the fine grid nodes. The restriction maps the fine grid solution to the coarse grid and is here selected to be the transpose of the prolongation, i.e. $\mathbf{P}^{\mathsf{T}}$. The coarse operator $\mathbf{K_c} = \mathbf{P}^{\mathsf{T}}\mathbf{KP}$ is obtained by Galerkin projections. The presented algorithm differs slightly from the so-called geometric multigrid formulation and more details can be found in the literature [e.g. 41, 40]. Defining a set of nested coarse discretizations $\mathcal{T}^{(1)} \subset \mathcal{T}^{(2)} \subset \cdots \subset \mathcal{T}^{(\mathcal{L})}$

and solving the coarse system recursively by re-applying the two-level algorithm on each coarse level $1, 2, \ldots, \mathcal{L}$ results in the so-called V-cycle MG algorithm. For smooth problems the algorithm can be utilized either as a stand alone solver or as a preconditioner in the PCG method where $\mathbf{M^{-1}}$ is replaced with the operator $\mathbf{r_k} \rightarrow \mathrm{MG}\left(\mathbf{0}, \mathbf{r_k}, \mathbf{K}\right)$ where $\mathbf{r_k} = \mathbf{f} - \mathbf{K}\mathbf{u_k}$ is the system residual at a certain PCG iteration k. When using a Jacobi smoother, the MG algorithm results in a symmetric preconditioner, which is a requirement for being applied in PCG [41]. The construction of the algorithm ingredients, namely the smoother, the interpolation operator and the restriction operator, does not require extensive computations. The MG preconditioner is spectrally equivalent to the system matrix $\mathbf{K}$ and the convergence rate of the resulting MGCG algorithm is independent of the mesh size. We note that the convergence rate does depend on the contrast in the system properties [15] and the development of contrast-independent multilevel preconditioners is an active research topic. Preliminary results in topology optimization were reported recently [26]. The application in multigrid settings will be reported in following articles.

Algorithm 1 Two-grid algorithm $\mathbf{u} = \mathrm{MG}\left(\mathbf{u}, \mathbf{f}, \mathbf{K}\right)$

Pre-smooth $\mathbf{u} = \mathbf{u} + \mathbf{S^{-1}}\left(\mathbf{f} - \mathbf{K}\mathbf{u}\right)$
Coarse grid correction - solve $\mathbf{K_c}\mathbf{u_c} = \mathbf{P}^{\mathsf{T}}\left(\mathbf{f} - \mathbf{K}\mathbf{u}\right)$
Interpolate $\mathbf{u} = \mathbf{u} + \mathbf{P}\mathbf{u_c}$
Post-smooth $\mathbf{u} = \mathbf{u} + \mathbf{S^{-1}}\left(\mathbf{f} - \mathbf{K}\mathbf{u}\right)$
return $\mathbf{u}$

3.3 MGCG implementation notes

We now turn to review the multigrid-CG (MGCG) implementation that is utilized in this study for solving the structural analysis equations. The developed code is based on the 88-line MATLAB code [7], which has been further extended to 3-D for some of the examples presented later in the article. For improving the transparency of the presentation, a 2-D MATLAB code for MGCG-based minimum compliance topology optimization is appended. The code includes all functions that are required for performing MGCG, according to a few user-defined parameters. The main change in comparison to the 88-line code is the replacement of the linear solver by MGCG.

The implementation can be separated logically into two main parts: A preparation phase and an optimization phase. During the preparation phase the prolongation matrix $\mathbf{P}^{\mathcal{L}}$ for each coarse level $\mathcal{L}$ is assembled. Even though the assembly cost for the prolongation operators can become relatively large, they are constructed only once and re-utilized in each optimization cycle. Therefore the total increase of the computational cost due to the preparation phase is negligible. In the MATLAB code, the prolongation operators are named `Pu` and are generated in lines 19-22, with the actual function named `Prepcoarse` in lines 165-192.

The optimization phase is entirely based on the 88-line topology optimization code with the linear solver replaced by the MGCG iterative solver. The MGCG solver is essentially a PCG procedure [e.g. 31] with a multigrid V-cycle as a preconditioner. The multigrid smoother $\mathbf{S^{-1}}$ (as in Algorithm 1) is based on the damped Jacobi method which can be stated as $\mathbf{S^{-1}} = \omega\mathbf{D^{-1}}$, where ω is a damping factor and $\mathbf{D}$ is the diagonal of the matrix $\mathbf{K}$. In the MATLAB code, the V-cycle is called within the MGCG iterations in line 114. The actual function `VCycle` appears in lines 135-149, followed by the damped Jacobi smoothing function.

For demonstrating the performance of the standard MGCG we present the following example, where we solve the FE analysis systems corresponding to the optimized layouts of MBB-beams in various mesh resolutions. For a fair comparison, we examine layouts generated with three different grids using an equal physical filter size: 1) 240×80 mesh, $r = 1.5$; 2) 480×160 mesh, $r = 3.0$; and 3) 960×320 mesh, $r = 6.0$. Furthermore, the coarsest grid level where MGCG performs a direct solve is the same for the three cases - namely a 30×10 grid. Convergence of MGCG is shown to be mesh-independent, meaning the same number of MGCG iterations is required regardless of the resolution of the finest grid. This is a significant advantage when compared to other preconditioning techniques. As expected, convergence of MGCG depends on the contrast of the stiffness distribution. Nevertheless, the procedure does indeed converge even for high-contrast layouts, and it appears to be insensitive to contrast once a certain value is exceeded.

The number of MGCG iterations required to achieve 10^{-10} accuracy (in terms of residual forces, $\frac{\|\mathbf{f}-\mathbf{K}\mathbf{u}_k\|}{\|\mathbf{f}\|}$) is plotted versus the problem size (in terms of number of DOF) in Figure 2.

For each mesh resolution, four FE systems are solved corresponding to various contrasts of the stiffness distribution E_{max}/E_{min}. The performance of MGCG is compared to that of PCG with an incomplete, zero fill-in Cholesky preconditioner (denoted IC(0) PCG). It can be seen that the number of MGCG iterations is significantly smaller than with IC(0) and it does not increase when refining the mesh. When increasing the contrast from 1 (corresponding to uniform material distribution) to 10^3 a significant increase in the number of MGCG iterations is observed; but further increase of the contrast has a minor effect. The IC(0) preconditioner exhibits a clear mesh-dependence: Refining the FE mesh results in an increase in the number of PCG iterations. With respect to contrast dependence, incomplete preconditioning appears to behave similar to MGCG.

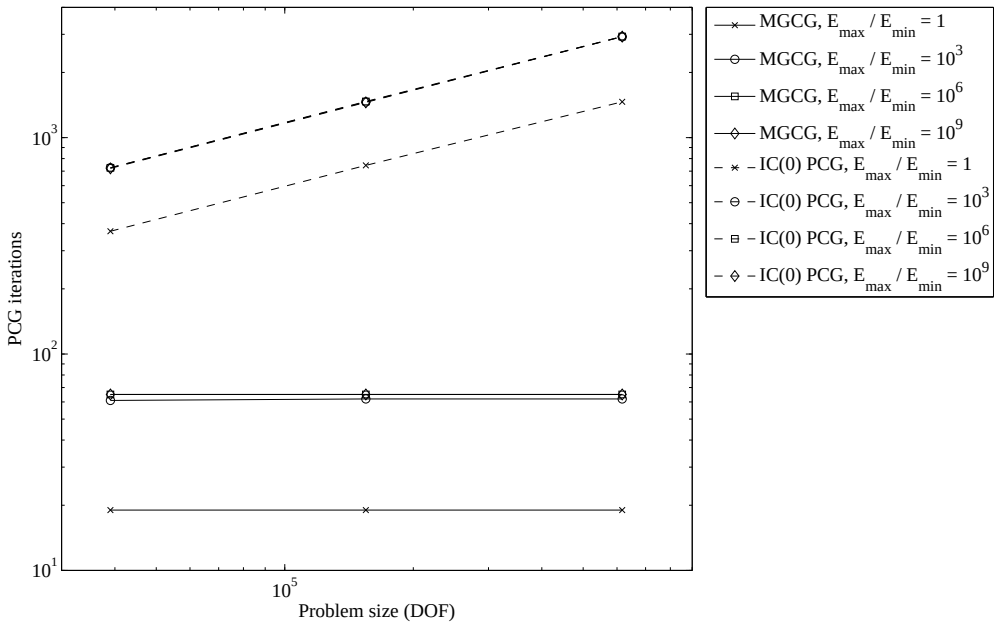

Figure 2: The number of PCG iterations required to achieve 10^{-10} accuracy in terms of residual forces for various mesh resolutions and contrasts E_{max}/E_{min}. Results correspond to the FE analysis of the optimized layouts of MBB-beams on 240×80, 480×160 and 960×320 grids. MGCG requires many fewer iterations and exhibits mesh-independent convergence while IC(0) does not.

4 Approximate approach based on MGCG solver

In addition to the improved efficiency resulting from utilizing the MGCG solver, we also suggest an approximate scheme that can further reduce computational burden. The approximate approach is derived primarily from extensive numerical experience with MGCG as the analysis solver for nested structural topology optimization problems. According to our experiments, it is evident that only a few MGCG iterations are required for achieving the "correct" optimization result - meaning the same layout and objective value that are obtained with an accurate solution of the analysis equations. This is not very surprising considering recent work with approximate procedures in topology optimization [4, 5, 3]. However, MGCG is unique due to the relations between the required number of iterations and the geometric parameters of the problem - namely the number of grid levels and the filter size. In the following, these relations will be discussed based on various test cases.

In the standard formulation (1) it is assumed that the structural analysis equilibrium equations are solved accurately, which is always the case if a direct solver is employed. When utilizing an iterative solver, one can choose to terminate the process after an arbitrary number of iterations and thus obtain an approximation of the solution

$$\mathbf{K}(\boldsymbol{\rho})\widetilde{\mathbf{u}} \approx \mathbf{f}.$$

The error associated with $\widetilde{\mathbf{u}}$ can be taken into account by means of consistent sensitivity analysis [4, 5, 6]. Another strategy is to compute the design sensitivities *as if* the analysis equations are

solved accurately, meaning that the error is transferred further to the design sensitivities

$$\frac{\partial \phi}{\partial \rho_e} \approx -\widetilde{\overline{\mathbf{u}}}^{\mathsf{T}} \frac{\partial \mathbf{K}}{\partial \rho_e} \widetilde{\mathbf{u}} \tag{3}$$

where $\widetilde{\overline{\mathbf{u}}}$ is an approximation of the adjoint vector $\overline{\mathbf{u}}$, obtained by an iterative solver in the same manner as $\widetilde{\mathbf{u}}$. According to recent experience, this strategy tends to yield better optimized performance than the consistent sensitivity approach [6].

With the goal of formulating an efficient yet reliable approximate procedure, two central requirements are considered: 1) A minimum number of MGCG iterations should be imposed, with relation to the geometric parameters of the problem - the number of grid levels and the filter size; and 2) The stopping criterion for MGCG should be related to the progress of optimization in order to avoid local minima. These two aspects are addressed in the following.

4.1 Imposing a minimum number of MGCG iterations

A minimum number of MGCG iterations is imposed, essentially in order to ensure reasonable error reduction throughout the fine mesh. In its basic (non-preconditioned) form, the error in CG propagates through a single finite element in every iteration. For this reason, "long" domains are less preferable than equilateral domains - computing an accurate solution will require more iterations the further the error needs to "travel". In the context of multigrid, one would require that the error propagates throughout the fine grid, while the solution (the preconditioning step) on the coarsest grid level is assumed to be accurate. The introduction of a filter improves this situation, because the filter operation has a smoothening effect on all fine grid points within the filtered domain. Therefore, a heuristic rule for a minimum number of MGCG iterations will ensure that the error is propagated by MGCG from the coarse grid points into the filtered domain.

Three demonstrative situations with various numbers of grid levels and filter sizes are displayed in Figure 3. The purpose is to provide a graphical explanation for the tendencies observed in various numerical experiments, and then to derive a heuristic rule for the minimum number of MGCG iterations to be performed. With three grid levels (every coarse-grid element represents 4-by-4 fine grid elements) and a small filter size ($r = 1.5$), only 1 MGCG step is necessary in order to propagate into the filtered domain. With four grid levels (every coarse-grid element represents 8-by-8 fine grid elements) and a large filter size ($r = 3$), after two MGCG steps the coarse solution will propagate into the filtered domain; whereas for a small filter size ($r = 1.5$) three MGCG steps are required. Based on geometric relations the minimum number of MGCG iterations is defined as

$$k_{min} \geq 2^{nl-2} - \frac{1}{2} - \frac{r}{\sqrt{2}} \tag{4}$$

where nl is the number of grid levels. In three-dimensional cases, the same reasoning leads to the relation

$$k_{min} \geq 2^{nl-2} - \frac{1}{2} - \frac{r}{\sqrt{3}}. \tag{5}$$

We note that this choice of k_{min} is based on a rather simplistic analysis of the situation, while in practice several additional factors play a role in determining the quality of the solution obtained by MGCG. On the one hand, the coarse grid solution is gradually prolongated and smoothed within intermediate grid levels. This means, that it is possible to obtain sufficient accuracy even with fewer MGCG iterations, as observed in some examples described in the following. On the other hand, propagation of the error itself does not ensure convergence of MGCG - in PCG procedures this is typically determined according to the relative norm of the residual forces. Nevertheless, it will be shown that by requiring a minimum number of MGCG iterations and at the same time imposing a relaxed convergence criterion on MGCG, accurate outcomes of the optimization can be achieved efficiently.

The argument in favor of imposing a minimum number of MGCG iteration is supported by the following experiment where we solve the cantilever beam problem. The results with 2-4 MG levels, sensitivity filtering with $r = 1.5$, contrast $E_{max}/E_{min} = 10^9$ and OC optimization are presented in Table 1. In all experiments, we run a maximum number of 500 design iterations - the stopping criterion that requires a maximum change of 10^{-3} in all design variables is not achieved. The reference accurate solution of the equilibrium equations is reached by enforcing a tolerance of 10^{-10} on the relative force residual. In general, it can be seen that only very few

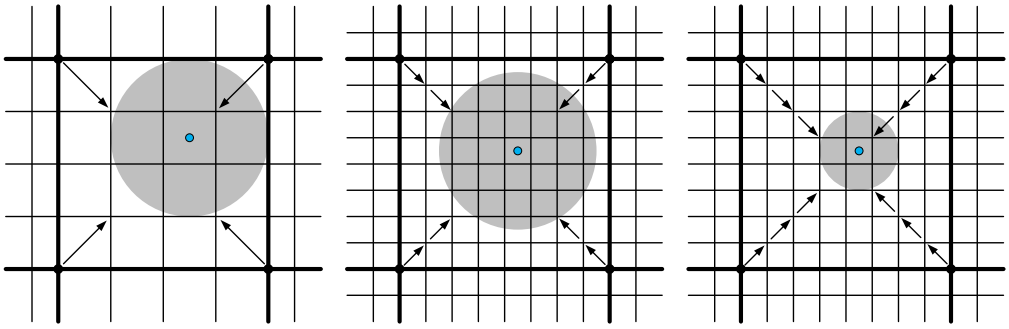

Figure 3: The effect of the number of grid levels and filter size on the number of MGCG iterations performed until the coarse-grid solution initially propagates into the filtered domain. Left: 3 grid levels, $r = 1.5$, 1 iteration; middle: 4 grid levels, $r = 3$, 2 iterations; right: 4 grid levels, $r = 1.5$, 3 iterations.

(up to 3) MGCG iterations are necessary for achieving an accurate optimization outcome. Quite surprisingly, even when 4 MG levels are used - meaning that small features defined by the filter size are not be captured on the coarse level - only small errors are exhibited with 1-5 MGCG iterations. As expected, fewer iterations are required if fewer MG levels are used, because the coarse-grid representation is more accurate. Judging by these results, roughly 10% of the MGCG iterations necessary for full convergence are actually necessary for accurate optimization. All the optimized layouts are practically identical, some examples are shown in Figure 4.

Table 1: Minimum compliance topology optimization of a 2-D cantilever, sensitivity filtering with $r = 1.5$. Results are shown for runs with 1-5 MGCG iterations and 2-4 MG levels. For reference, the average number of MGCG iterations per cycle and the objective value achieved with accurate analysis are presented in the bottom row.

4 MG levels, OC optimizer			3 MG levels, OC optimizer			2 MG levels, OC optimizer		
MGCG it. per cycle	Objective value	Relative diff. (%)	MGCG it. per cycle	Objective value	Relative diff. (%)	MGCG it. per cycle	Objective value	Relative diff. (%)
1	77.56	+0.13	1	77.43	-0.039	1	77.45	-0.013
2	77.55	+0.12	2	77.46	0.0	2	77.47	+0.013
3	77.45	-0.013	3	77.47	+0.013	3	77.46	0.0
4	77.45	-0.013	4	77.45	-0.013	4	77.46	0.0
5	77.47	+0.013	5	77.46	0.0	5	77.46	0.0
31.5	77.46	accurate	24.6	77.46	accurate	16.4	77.46	accurate

Results from experiments with a larger filter size, $r = 3$, are presented in Table 2. With both sensitivity as well as density filtering, using both OC and MMA, only one or two MGCG iterations are needed in order to reach an accurate result of the optimization. An intuitive explanation is that with a rather large filter, all structural features are captured relatively accurately on all grid levels. This means that the preconditioning operation is closer to solving on the fine grid, compared to the case of a small filter. This is also in line with the sketch in Figure 3: It is expected that fewer MGCG iterations will be required because of the proximity of the coarse grid points to the filtered domain. Again, we see potential for significant computational savings simply by reducing the number of MGCG iterations.

4.2 Relaxed MGCG convergence criterion

In order to maintain sufficient accuracy of the approximation generated by MGCG, it is suggested to impose a convergence criterion on the approximation of the design sensitivities. Because the nested approach is followed, the accuracy achieved in the analysis phase influences the solution of the optimization problem via the design sensitivities. This calls for direct monitoring of the accuracy of the design sensitivities, as it improves when performing further MGCG iterations. It is suggested to terminate MGCG once all design sensitivities satisfy the following criteria: (I)

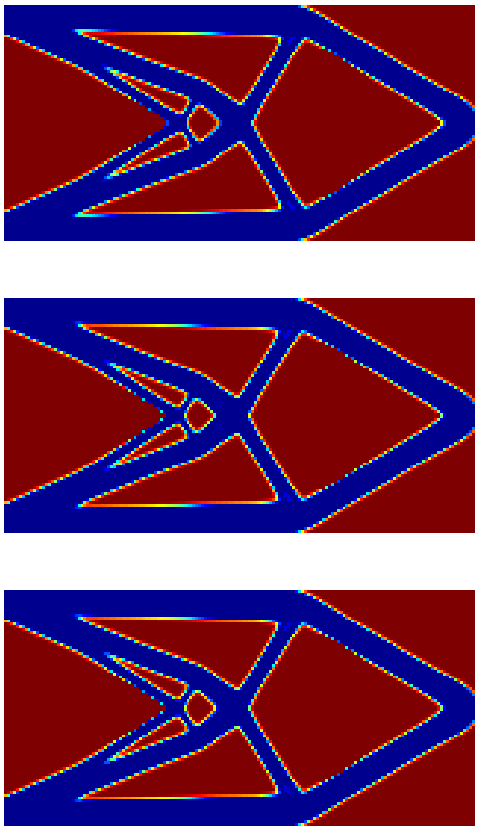

Figure 4: Optimized layouts corresponding to results from Table 1. From top: 4 MG levels and 1 MGCG iteration; 3 MG levels and 1 MGCG iteration; and accurate solve. Even though only a single MGCG iteration is performed, small design features that cannot be realized on the coarsest grid are captured on the fine grid. The optimized layouts are hardly distinguishable and the difference in compliance does not exceed 0.13%.

Table 2: Minimum compliance topology optimization of a 2-D cantilever, various optimizers and various filters with $r = 3$. Results are shown for runs with 1-3 MGCG iterations, four MG levels and two different optimizers. For reference, the average number of MGCG iterations per cycle and the objective value achieved with accurate analysis are presented in the bottom row.

4 MG levels, sensitivity filter, OC optimizer			4 MG levels, density filter, OC optimizer			4 MG levels, density filter, MMA optimizer		
MGCG it. per cycle	Objective value	Relative diff. (%)	MGCG it. per cycle	Objective value	Relative diff. (%)	MGCG it. per cycle	Objective value	Relative diff. (%)
1	80.41	-0.062	1	84.41	+0.059	1	83.83	-0.12
2	80.44	-0.025	2	84.36	0.0	2	83.85	-0.095
3	80.46	0.0	3	84.36	0.0	3	83.93	0.0
20.8	80.46	accurate	17.4	84.36	accurate	19.0	83.93	accurate

The relative change between consecutive MGCG cycles is smaller than a certain threshold; (II) The relative change between consecutive MGCG cycles is smaller than the current error in the corresponding term of the KKT conditions.

The rationale behind criterion (I) follows previous work on early termination of PCG, based on the accuracy of the objective function which can be seen as an aggregated measure for the accuracy of design sensitivities [5]. By "early termination of PCG", we refer to the situation in which the standard convergence criterion related to the relative norm of residual forces is not necessarily satisfied. It was shown that accurate optimization results can be obtained despite the utilization of such approximations in the analysis phase. In the current work, we take this idea one step further by directly monitoring all design sensitivities rather than an aggregate measure - thus imposing a tighter convergence requirement. Referring to a general objective/constraint functional of the form $\mathbf{l}^\mathsf{T}\mathbf{u}$ as in (1) and to the corresponding approximate design sensitivities (3), criterion (I) has the form

$$\frac{\left|\widetilde{\overline{\mathbf{u}}}_k^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e} \widetilde{\mathbf{u}}_k - \widetilde{\overline{\mathbf{u}}}_{k-1}^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e} \widetilde{\mathbf{u}}_{k-1}\right|}{\left|\widetilde{\overline{\mathbf{u}}}_k^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e} \widetilde{\mathbf{u}}_k\right|} < \eta \qquad \forall e \tag{6}$$

where k and $k-1$ represent two consecutive MGCG iterations.

The rationale behind criterion (II) is that when the optimization problem is far from a candidate solution, namely a point satisfying the KKT conditions, a strictly accurate solution of the analysis equations may not be necessary. This idea is elaborated in the following. The Lagrangian of the optimization problem (1) is

$$\mathcal{L} = \mathbf{l}^\mathsf{T}\mathbf{u} + \overline{\mathbf{u}}^\mathsf{T}(\mathbf{f} - \mathbf{K}(\boldsymbol{\rho})\mathbf{u}) + \Lambda(\sum_{e=1}^{N} v_e\rho_e - V) + \sum_{e=1}^{N} \lambda_e^+(\rho_e - 1) + \sum_{e=1}^{N} \lambda_e^-(-\rho_e).$$

The condition of optimality with respect to a certain element density ρ_e is

$$\frac{\partial \mathcal{L}}{\partial \rho_e} = -\overline{\mathbf{u}}^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e}\mathbf{u} + \Lambda v_e + \lambda_e^+ - \lambda_e^-.$$

For intermediate densities we then have the classical condition requiring constant mutual energy density

$$-\overline{\mathbf{u}}^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e}\mathbf{u} + \Lambda v_e = 0. \tag{7}$$

During the process of topology optimization, some design variables reach one of the bounds while the rest are determined by the condition (7) which forms the basis for typical optimality criteria approaches [12]. In case the residual corresponding to (7) is large, it makes sense to relax the requirement for highly accurate evaluation of the expression $-\overline{\mathbf{u}}^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e}\mathbf{u}$. Therefore criterion (II) has the form

$$\left|\widetilde{\overline{\mathbf{u}}}_k^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e} \widetilde{\mathbf{u}}_k - \widetilde{\overline{\mathbf{u}}}_{k-1}^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e} \widetilde{\mathbf{u}}_{k-1}\right| < \left|-\overline{\mathbf{u}}^\mathsf{T} \frac{\partial \mathbf{K}}{\partial \rho_e}\mathbf{u} + \Lambda v_e\right| \qquad \forall e \,|0 < \rho_e \ < 1 \tag{8}$$

where the right-hand-side term is based on information from the previous design cycle.

Direct control over the error in the design sensitivities is absolutely necessary in order to avoid local minimum solutions. This occurs for example when solving the cantilever test case with a

small density filter $r = 1.5$ and MMA as the optimizer. As can be seen in Table 3, procedures with up to 5 MGCG iterations fall into a local minimum, corresponding to the layout in the top of Figure 5. With 6 or more MGCG iterations these errors are not observed, but in practice we cannot predict how many iterations will be required - this can only be determined in an adaptive manner based on error control.

Table 3: Minimum compliance topology optimization of a 2-D cantilever, density filtering with $r = 1.5$. Results are shown for runs with 1-8 MGCG iterations, four MG levels and MMA optimizer. For reference, the average number of MGCG iterations per cycle and the objective value achieved with accurate analysis are presented in the bottom row.

4 MG levels, MMA optimizer		
MGCG it. per cycle	Objective value	Relative diff. (%)
1	79.90	+1.6
3	79.90	+1.6
5	79.92	+1.6
6	78.67	+0.013
8	78.65	-0.013
44.6	78.66	accurate

4.3 Accuracy-efficiency with MGCG solver

We now turn to demonstrate the accuracy and efficiency of the proposed approximate procedure, where we integrated both requirements: A minimum number of MGCG iterations based on geometric parameters; and a stopping criterion for MGCG based on the accuracy of the design sensitivities. As a first example we examine the problem described in Table 3, for which a local minimum was reached when fewer than 6 MGCG iterations were performed every design cycle. The purpose is to reach an accurate optimization result with relatively low cost, but without stating a constant number of MGCG iterations. Instead, we set a minimum number of MGCG iterations $k_{min} = 3$ and require that both criteria (6) and (8) are satisfied. The results are presented in Table 4 for various values of the threshold η.

The efficiency of the proposed approach is clearly demonstrated: Once a tight enough tolerance is enforced by the parameter η (10^{-2} and below), accurate outcome of the optimization is achieved with significant savings in computer time, reflected in the average number of MGCG iterations per design cycle. The key for such savings is in the adaptive nature of the procedure - more MGCG cycles are performed when it is necessary to compute relatively accurate design sensitivities. This adaptive characteristic is demonstrated in Figure 6 where the number of MGCG iterations is plotted versus design cycles, for various values of η.

The approximate approach can be integrated into topology optimization procedures based not only on rigorous nonlinear programming, but also on heuristic optimality criteria. For example, we

Table 4: Minimum compliance topology optimization of a 2-D cantilever, density filtering with $r = 1.5$, four MG levels and MMA optimizer. Accurate optimization outcome is achieved with up to 89% savings in the number of MGCG iterations.

4 MG levels, density filter, MMA optimizer			
η	MGCG it. per cycle	Objective value	Relative diff. (%)
$2 \cdot 10^{-1}$	5.12	78.67	+0.013
10^{-1}	7.42	78.65	-0.013
10^{-2}	10.42	78.64	-0.025
10^{-3}	16.76	78.66	0.0
Standard	44.6	78.66	accurate

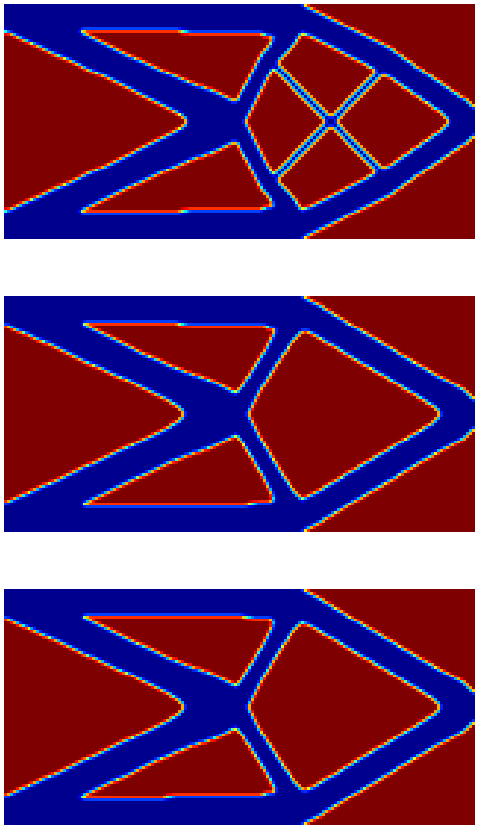

Figure 5: Optimized layouts corresponding to results from Table 3. From top: MMA with 5 MGCG iterations, 6 MGCG iterations and accurate solve. Notice the local minimum obtained when too few MGCG iterations are performed.

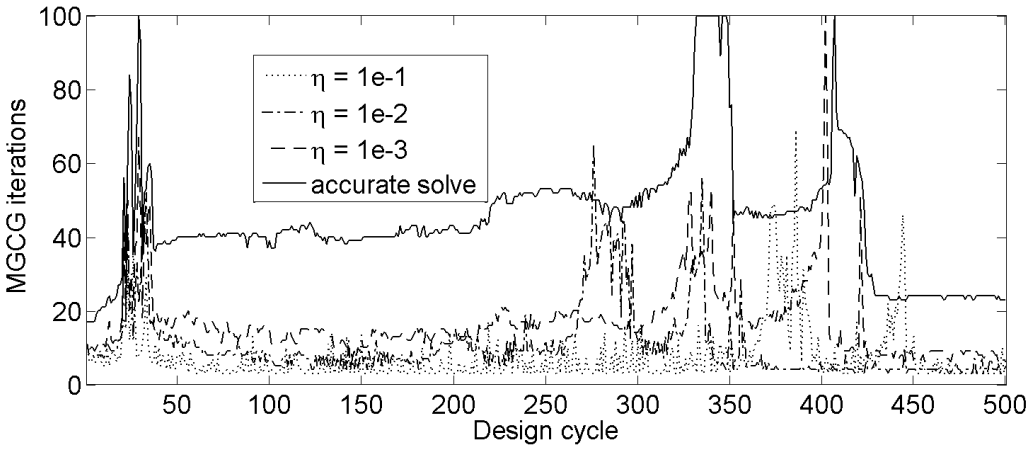

Figure 6: MGCG iterations performed throughout the optimization process of a 2-D cantilever beam, for various threshold values of η. Many MGCG iterations are required in the beginning and for overcoming the local minimum, otherwise the number is close or equal to the minimum.

Table 5: Minimum compliance topology optimization of a 2-D cantilever, sensitivity filtering with $r = 1.5$, four MG levels and OC optimizer. Accurate optimization outcome is achieved with up to 83% savings in the number of MGCG iterations.

4 MG levels, sensitivity filter, OC optimizer			
η	MGCG it. per cycle	Objective value	Relative diff. (%)
$2 \cdot 10^{-1}$	5.22	77.46	0.0
10^{-1}	5.29	77.46	0.0
10^{-2}	5.72	77.46	0.0
10^{-3}	10.10	77.46	0.0
Standard	31.5	77.46	accurate

Table 6: Maximum output of a 2-D force inverter mechanism, density filtering with $r = 3.0$, four MG levels and MMA optimizer. Accurate optimization outcome is achieved with up to 73% savings in the number of MGCG iterations.

4 MG levels, density filter, MMA optimizer			
η	MGCG it. per cycle	Objective value	Relative diff. (%)
$2 \cdot 10^{-1}$	7.34	-2.268	0.0
10^{-1}	6.34	-2.267	+0.044
10^{-2}	10.35	-2.268	0.0
10^{-3}	13.97	-2.272	-0.18
Standard	23.56	-2.268	accurate

attempt to reproduce the result from the first column in Table 1 which was generated by an OC procedure. Again we choose $k_{min} = 3$ and require that both criteria (6) and (8) are satisfied. As can be seen in Table 5, all approximate procedures reached the same objective value as obtained when performing fully accurate analyses, with significant reductions in the number of MGCG iterations. Furthermore, similar computational effort was required for three different threshold values of η: $2 \cdot 10^{-1}$, 10^{-1} and 10^{-2}. This indicates that MGCG was typically terminated according to (8), thus supporting our argument that this is a sensible stopping criterion.

The approximate approach is applicable also to other classes of optimization problems which are not self-adjoint. A typical example is the design of a force inverting mechanism, see Figure 1 for the problem setup. We use 4 MG levels, contrast $E_{max}/E_{min} = 10^6$ and 500 MMA optimization cycles. The density filter radius is $r = 3.0$ therefore the minimum number of MGCG iterations is set to 2. The objective value and the number of MGCG iterations per cycle, for different threshold values η, are given in Table 6. While an accurate analysis requires over 23 MGCG iterations per design cycle in average, the same outcome can be achieved with roughly 8 MGCG iterations per design cycle - just above 1/3 of the computational effort. The number of MGCG iterations performed throughout the optimization is plotted in Figure 7. Again the adaptive characteristics of the procedure are revealed - the number of MGCG iterations varies according to the progress of optimization, with a clear peak in the first 40 cycles where a local minimum solution is successfully avoided.

5 Examples

In order to further establish the validity of the MGCG-based approach to topology optimization, we hereby discuss the solution of three example problems: Fine-resolution 2-D minimum compliance; 3-D minimum compliance; and 3-D force inverting mechanism. All examples were solved using a sequential, single-processor MATLAB code running on a standard laptop PC.

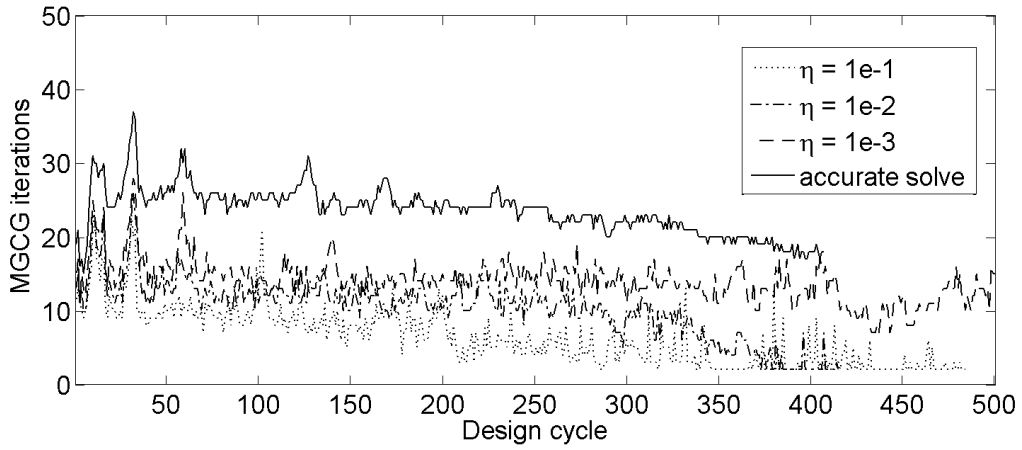

Figure 7: MGCG iterations performed throughout the optimization process of a 2-D force inverter, for various threshold values of η. Note that convergence of the optimization was achieved in less than 500 design cycles with the accurate solve, as well as with $\eta = 10^{-1}$ and $\eta = 10^{-2}$.

5.1 Example 1: 2-D MBB beam

In this example we demonstrate the performance of the MGCG-based procedure in solving a minimum compliance optimization of an MBB-beam. Even though 2-D applications are not of primary concern due to the efficiency of direct sparse solvers [7], the purpose is to show that high-resolution designs can indeed be obtained with multigrid-based procedures. Exploiting symmetry, half of the beam is optimized on a 480×160 mesh; the volume fraction is 0.5; and the modified SIMP penalty is gradually increased from 1.0 to 3.0. We use density filtering with a radius of $r = 1.5$. Within the multigrid V-cycle, we apply a single damped Jacobi smoothening cycle with $\omega = 0.8$. Five multigrid levels are utilized, meaning a direct solve is performed on a 30×10 grid. According to (4), the minimum number of MGCG iterations is set to 6. Optimization is performed using optimality criteria and is terminated after 500 design cycles - the stopping criterion that requires a maximum change of 10^{-3} in all design variables is not achieved.

A reference result that corresponds to a fully accurate analysis is obtained by setting $\eta = 10^{-6}$ in (6). Consequently, the relative norm of residual forces is in the order of 10^{-8} or smaller throughout the optimization process. The objective value achieved is 193.4, and the average number of MGCG iterations per cycle is 60.05. With $\eta = 1$, meaning criterion I (6) is essentially disregarded and only criterion II (8) is applied to control the error, we achieve the exact same objective value of 193.4, but with only 12.79 MGCG iterations per cycle. Moreover, the optimized layout obtained with the approximate procedure is identical to that from the fully accurate procedure, see Figure 8. In the MATLAB implementation, the savings in MGCG iterations are not directly translated into savings in computing time due to the effort invested in matrix assembly. Nevertheless, CPU time measured in MATLAB is reduced by 62% - from 2278 seconds to 868 seconds. The same example was solved with the 88-line code [7] that uses a direct sparse solver in 899 seconds and the objective was practically the same: 193.7 (though a very thin bar is not eliminated). With a standard zero fill-in incomplete Cholesky preconditioner, an average number of 1,092.4 PCG iterations per design cycle were required to obtain 10^{-8} accuracy in terms of residual forces and computing time exceeded 4 hours. The achieved objective was slightly better - 192.0, but the optimized layout consists of many thin 'gray' bars and does not constitute a rational design. This further demonstrates the attractiveness of the MGCG-based approach to topology optimization, even for medium-sized two-dimensional problems.

5.2 Example 2: 3-D cantilever

In this example we investigate the performance of the MGCG-based procedure in the optimization of a three-dimensional cantilever beam with a sine-shaped load at the bottom of the free edge, see Figure 9 for the problem setup. For comparing accurate versus approximate approaches, we

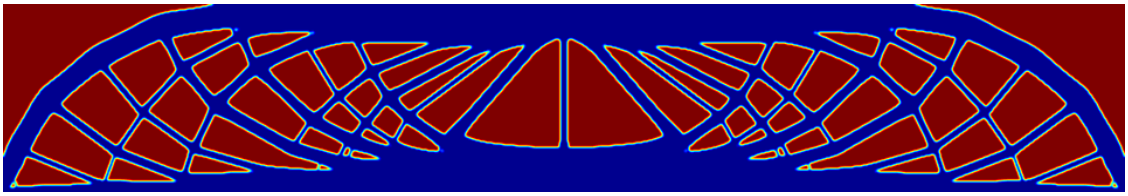

Figure 8: Optimized layout of a 960×160 simply-supported beam with a point load, generated using an MGCG iterative solver. Objective value and CPU time are competitive in comparison with the results of the 88-line code.

first solve the problem on a 48×24×24 mesh; the volume fraction is 0.12; and the modified SIMP penalty is 3.0. We use sensitivity filtering with a radius of $r = \sqrt{3}$. Within the multigrid V-cycle, we apply a single damped Jacobi smoothening cycle with $\omega = 0.6$. Four multigrid levels are utilized, meaning a direct solve is performed on a 6×3×3 grid. According to (5), the minimum number of MGCG iterations is set to 3. Optimization is performed using optimality criteria and is terminated after 50 design cycles - the stopping criterion that requires a maximum change of 10^{-2} in all design variables is not achieved.

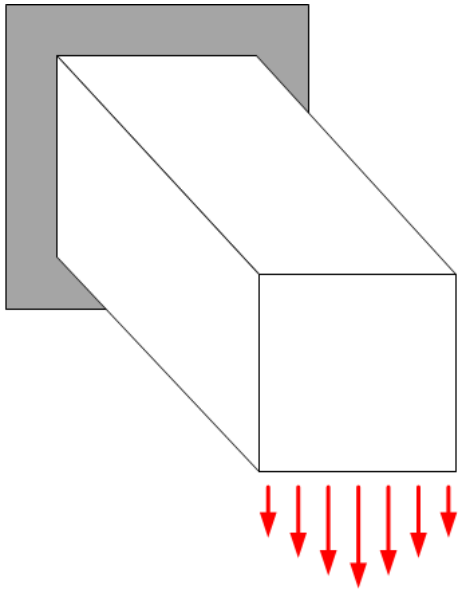

Figure 9: Problem setup for the minimum compliance of a 3-D cantilever beam.

With accurate analyses, the optimized compliance after 50 iterations is 3,330, the average number of MGCG iterations per cycle is 60.32 and the measured CPU time is 364 seconds. The exact same objective is obtained by an approximate procedure with $\eta = 1$, again meaning only criterion II (8) is controlling the error. Computational effort is reduced significantly, by performing only 15.04 MGCG iterations per cycle within a total CPU time of 183 seconds.

The main purpose of the article is to present a computational approach that facilitates the solution of 3-D structural topology optimization problems on a standard PC in reasonable time. This is clearly achieved: The cantilever beam problem modeled using a 80×40×40 mesh, with over 400,000 degrees of freedom, is solved by our MATLAB program in less than 15 minutes. The total number of MGCG iterations was 921, i.e. 18.42 per design cycle in average. We believe that the suggested procedure indeed paves the way for efficient implementations in computational environments that do not enjoy the benefits of high performance computing and parallel architectures, such as applications on mobile devices and plug-ins for architectural modeling software.

The MGCG-based procedure appears to be very effective also in comparison to other Krylov subspace solvers applied previously in the context of topology optimization. For example, Wang et al. [42] suggest a MINRES procedure with subspace recycling, scaling and a relaxed convergence tolerance. The number of iterations per design cycle is indeed relatively low - approximately 100 in average throughout the optimization process of a 3-D beam with 110,925 DOF. However, each

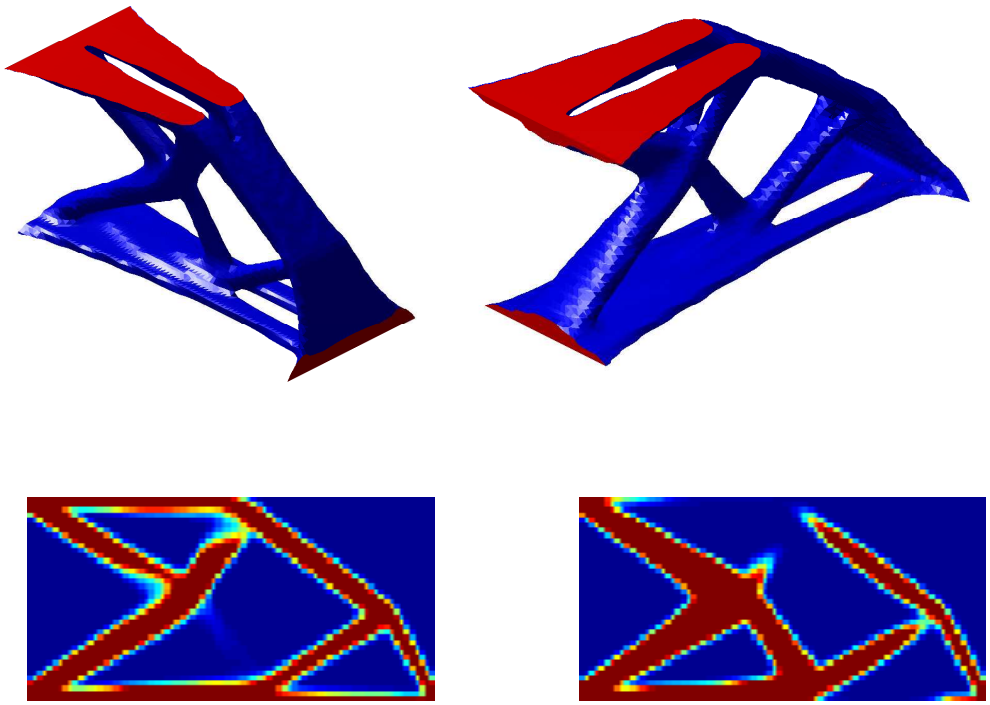

Figure 10: Optimized layout of a 3-D cantilever beam using a 80×40×40 mesh. 50 design iterations required less than 15 minutes on a single processor running MATLAB. The density value of the isosurface displayed is 0.8. The bottom figures show longitudinal slices at the 18th and 20th elements in the y-direction.

linear solve took 50.5 seconds in average on a single processor with similar properties to the one used in the current study. For comparison, the current MGCG-based procedure requires 3.66 seconds per design cycle of a 3-D beam with 91,875 DOF. Even if we assume that advances in computer hardware are responsible for part of the gap, still the current results imply significant improvements. Another reference result is the utilization of FETI-DP [20] in parallel topology optimization as reported by Evgrafov et al. [19]. When optimizing a 3-D beam with roughly 1,000,000 DOF, 300 CG iterations were required in each design cycle, taking approximately 85 seconds on 96 processors. Again, it appears that MGCG can be much more efficient: In the example above with 400,000 DOF, only 18.42 MGCG iterations were performed every design cycle and the time required was 17.86 seconds on a single processor. Assuming linear increase in time with respect to problem size, as expected in multigrid procedures, implies significant reduction in computing time also in comparison to the FETI-DP approach.

5.3 Example 3: 3-D force inverter

For demonstrating the capability to tackle also non-self-adjoint problems, we solve a 3-D force inverter problem, see Figure 11 for the problem setup. Exploiting double symmetry, only one quarter of the structure is optimized on a $64\times32\times32$ mesh. The spring stiffnesses are set to 0.1 for the input spring and 0.0001 for the output spring. The allowed volume fraction is 0.1 and the penalization factor used in the modified SIMP interpolation is 3.0. Within the multigrid V-cycle, we apply a single damped Jacobi smoothening cycle with $\omega = 0.6$. A density filter is utilized with the radius of $r = 3$ and the number of multigrid levels is four. Consequently the minimum number of MGCG iterations is set to 2, but in practice the number of MGCG iterations performed is greater or equal to 6 throughout all 100 design cycles. Again we set $\eta = 1$ in (6), leaving the error to be controlled by satisfying (8). Updates of the design are performed by MMA [38].

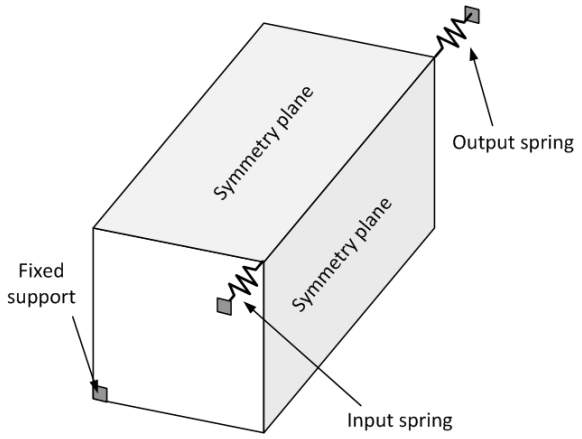

Figure 11: Problem setup for the maximum output displacement of a 3-D force inverter.

The optimized force inverter provides an output displacement of 23.76, see Figure 12 for the layout. This result is achieved after 100 design cycles, taking nearly 24 minutes on a standard PC (i.e. 14.3 seconds per design cycle). The average number of MGCG iterations per design cycle is 12.17, ranging between 6 to 33 iterations according to the accuracy requirements dictated by the convergence criterion. The MGCG-based procedure successfully overcomes a well-known strong local minimum at the objective value of zero and produces a result that physically resembles previous solutions of the 3-D force inverter problem [19, 5]. At the same time, computational efficiency is significantly improved compared to previous reports. Evgrafov et al. [19] solved the 3-D inverter problem on a much finer $150\times75\times75$ grid - meaning roughly 12 times the number of DOF used in the current study. They performed 360 CG iterations taking 830 seconds per design cycle on 8 processors. Amir et al. [5] used a coarser $60\times30\times30$ grid, IC(0) for preconditioning and a relaxed convergence criterion. They reported performing 58.71 PCG iterations in average throughout 200 design cycles, but no timing data is provided. Even with the differences in mesh resolutions taken into account, the MGCG-based procedure clearly offers the most promising computational efficiency.

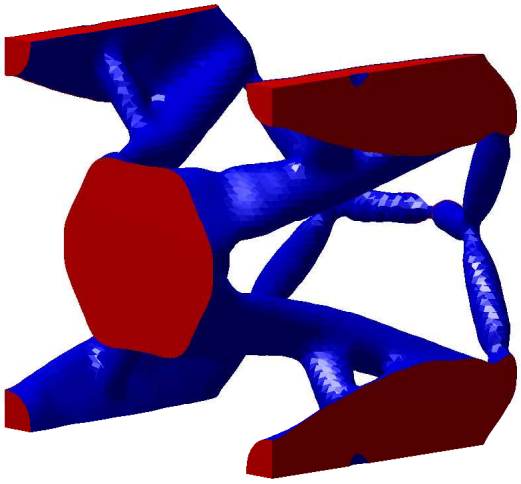

Figure 12: Optimized layout of a 3-D force inverter using a 64×32×32 mesh for one twice-symmetric quarter. 100 design iterations required less than 24 minutes on a single processor running MATLAB. The density value of the isosurface displayed is 0.3.

6 Discussion

An efficient computational approach to topology optimization was presented, based on utilizing a multigrid preconditioned conjugate gradient (MGCG) solver in the nested analysis problem. This facilitates the solution of three-dimensional design problems in reasonable time on a standard PC. The positive results reported here encourage the further development of the MGCG-based procedure also for high performance parallel computing environments where good scalability is expected based on recent experience with parallelization of MGCG.

The article exhibits three main findings regarding the integration of MGCG in topology optimization procedures: 1) In its basic form, MGCG converges also for high-contrast parameter distributions, thus it is suitable as a linear solver for the sequence of analysis problems arising in topology optimization; 2) The number of MGCG iterations required for obtaining an accurate optimization outcome is related to the geometric characteristics of the problem - the number of grid levels and the filter size - thus a minimum number of MGCG can be defined based on these parameters; and 3) The total number of MGCG iterations can be reduced by relating the required accuracy of the design sensitivities to the progress of optimization - thus implying an adaptive scheme where computational effort is invested only when it is necessary for achieving an accurate outcome of optimization.

Implemented in MATLAB, the MGCG-based approach solves 3-D topology optimization problems in reasonable time on a standard PC. For example, a cantilever with 400,000 degrees of freedom was optimized in less than 15 minutes. Furthermore, the approach appears to be competitive also for medium-sized 2-D problems where run time was slightly shorter than of the 88-line code. This clearly demonstrates the suitability of MGCG-based procedures for applications running in basic computational environments, e.g. the TopOpt app on hand-held devices and the Grasshopper-Rhino plug-in on standard PC's. Positive experience with parallel implementation of MGCG implies that the presented approach is very promising also in the context of large-scale topology optimization on high performance computers.

7 Acknowledgments

The authors wish to thank Ole Sigmund for fruitful discussions and helpful comments on the manuscript. The anonymous reviewers' valuable remarks are gratefully acknowledged. The authors acknowledge the financial support received from the European Commission Research Executive Agency, grant agreement PCIG12-GA-2012-333647; and from the NextTop project, sponsored by the Villum foundation. The authors also thank Krister Svanberg for the MATLAB MMA code.

References

[1] N. Aage and B. Lazarov. Parallel framework for topology optimization using the method of moving asymptotes. *Structural and Multidisciplinary Optimization*, 47(4):493–505, 2013. doi: 10.1007/s00158-012-0869-2.

[2] N. Aage, M. Nobel-Jørgensen, C. S. Andreasen, and O. Sigmund. Interactive topology optimization on hand-held devices. *Structural and Multidisciplinary Optimization*, 47(1):1–6, 2013.

[3] O. Amir and O. Sigmund. On reducing computational effort in topology optimization: how far can we go? *Structural and Multidisciplinary Optimization*, 44:25–29, 2011.

[4] O. Amir, M. P. Bendsøe, and O. Sigmund. Approximate reanalysis in topology optimization. *International Journal for Numerical Methods in Engineering*, 78:1474–1491, 2009.

[5] O. Amir, M. Stolpe, and O. Sigmund. Efficient use of iterative solvers in nested topology optimization. *Structural and Multidisciplinary Optimization*, 42:55–72, 2010.

[6] O. Amir, O. Sigmund, M. Schevenels, and B. Lazarov. Efficient reanalysis techniques for robust topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 245-246:217–231, 2012.

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

[8] S. F. Ashby and R. D. Falgout. A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nuclear Science and Engineering*, 124:145–159, 1996.

[9] A. Baker, R. Falgout, T. Gamblin, T. Kolev, M. Schulz, and U. Yang. Scaling algebraic multigrid solvers: On the road to exascale. In C. Bischof, H.-G. Hegering, W. E. Nagel, and G. Wittum, editors, *Competence in High Performance Computing 2010*, pages 215–226. Springer Berlin Heidelberg, 2012.

[10] A. Baker, R. Falgout, T. Kolev, and U. Yang. Scaling hypre's multigrid solvers to 100,000 cores. In M. W. Berry, K. A. Gallivan, E. Gallopoulos, A. Grama, B. Philippe, Y. Saad, and F. Saied, editors, *High-Performance Scientific Computing*, pages 261–279. Springer London, 2012. doi: 10.1007/978-1-4471-2437-5_13.

[11] M. P. Bendsøe. Optimal shape design as a material distribution problem. *Structural Optimization*, 1:193–202, 1989.

[12] M. P. Bendsøe and O. Sigmund. *Topology Optimization - Theory, Methods and Applications*. Springer, Berlin, 2003.

[13] M. Bogomolny. Topology optimization for free vibrations using combined approximations. *International Journal for Numerical Methods in Engineering*, 82(5):617–636, 2010. doi: 10.1002/nme.2778.

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

[15] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Mathematics of Computation*, 57:23–45, 1991.

[16] T. E. Bruns and D. A. Tortorelli. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, 190:3443–3459, 2001.

[17] E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang. A survey of parallelization techniques for multigrid solvers. In M. A. Heroux, P. Raghavan, and H. D. Simon, editors, *Parallel Processing for Scientific Computing*, chapter 10, pages 179–201. SIAM, 2006.

[18] T. A. Davis. *Direct Methods for Sparse Linear System*. SIAM, 2006.

[19] A. Evgrafov, C. J. Rupp, K. Maute, and M. L. Dunn. Large-scale parallel topology optimization using a dual-primal substructuring solver. *Structural and Multidisciplinary Optimization*, 36:329–345, 2008.

[20] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen. FETI-DP: a dual–primal unified FETI method–part I: A faster alternative to the two-level FETI method. *International journal for numerical methods in engineering*, 50(7):1523–1544, 2001.

[21] J. K. Guest and L. C. Smith Genut. Reducing dimensionality in topology optimization using adaptive design variable fields. *International Journal for Numerical Methods in Engineering*, 81(8):1019–1045, 2010. doi: 10.1002/nme.2724.

[22] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[23] J. E. Kim, G.-W. Jang, and Y. Y. Kim. Adaptive multiscale wavelet-galerkin analysis for plane elasticity problems and its applications to multiscale topology design optimization. *International journal of solids and structures*, 40(23):6473–6496, 2003.

[24] S. Y. Kim, I. Y. Kim, and C. K. Mechefske. A new efficient convergence criterion for reducing computational expense in topology optimization: reducible design variable method. *International Journal for Numerical Methods in Engineering*, 90(6):752–783, 2012. doi: 10.1002/nme.3343.

[25] Y. Y. Kim and G. H. Yoon. Multi-resolution multi-scale topology optimization-a new paradigm. *International Journal of Solids and Structures*, 37(39):5529–5559, 2000.

[26] B. S. Lazarov. Topology optimization using multiscale finite element method for high-contrast media. 2013. In review.

[27] B. Maar and V. Schulz. Interior point multigrid methods for topology optimization. *Structural and Multidisciplinary Optimization*, 19:214–224, 2000.

[28] T. H. Nguyen, G. H. Paulino, J. Song, and C. H. Le. A computational paradigm for multiresolution topology optimization (mtop). *Structural and Multidisciplinary Optimization*, 41: 525–539, 2010. doi: 10.1007/s00158-009-0443-8.

[29] T. H. Nguyen, G. H. Paulino, J. Song, and C. H. Le. Improving multiresolution topology optimization via multiple discretizations. *International Journal for Numerical Methods in Engineering*, 92(6):507–530, 2012. doi: 10.1002/nme.4344.

[30] T. A. Poulsen. Topology optimization in wavelet space. *International Journal for Numerical Methods in Engineering*, 53(3):567–582, 2002.

[31] Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM, 2003.

[32] O. Sigmund. On the design of compliant mechanisms using topology optimization. *Mechanics Based Design of Structures and Machines*, 25:493–524, 1997.

[33] O. Sigmund and K. Maute. Sensitivity filtering from a continuum mechanics perspective. *Structural and Multidisciplinary Optimization*, 46:471–475, 2012. doi: 10.1007/s00158-012-0814-4.

[34] O. Sigmund and S. Torquato. Design of materials with extreme thermal expansion using a three-phase topology optimization method. *Journal of the Mechanics and Physics of Solids*, 45(6):1037–1067, 1997.

[35] R. Stainko. An adaptive multilevel approach to the minimal compliance problem in topology optimization. *Communications in Numerical Methods in Engineering*, 22(2):109–118, 2006. doi: 10.1002/cnm.800.

[36] R. Stainko. *Advanced Multilevel Techniques to Topology Optimization*. PhD thesis, Johannes Kepler Universität Linz, 2006.

[37] K. Suresh. Efficient generation of large-scale pareto-optimal topologies. *Structural and Multidisciplinary Optimization*, 47:49–61, 2013. doi: 10.1007/s00158-012-0807-3.

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

[39] O. Tatebe and Y. Oyanagi. Efficient implementation of the multigrid preconditioned conjugate gradient method on distributed memory machines. In *Supercomputing'94. Proceedings*, pages 194–203. IEEE, 1994.

[40] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001.

[41] P. S. Vassilevski. *Multilevel Block Factorization Preconditioners: Matrix-based Analysis and Algorithms for Solving Finite Element Equations*. Springer, New York, 2008.

[42] S. Wang, E. de Sturler, and G. H. Paulino. Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. *International Journal for Numerical Methods in Engineering*, 69:2441–2468, 2007.

[43] S. Zhou and M. Y. Wang. Multimaterial structural topology optimization with a generalized cahn-hilliard model of multiphase transition. *Structural and Multidisciplinary Optimization*, 33:89–111, 2007. doi: 10.1007/s00158-006-0035-9.

[44] W. Zuo, T. Xu, H. Zhang, and T. Xu. Fast structural optimization with frequency constraints by genetic algorithm using adaptive eigenvalue reanalysis methods. *Structural and Multidisciplinary Optimization*, 43:799–810, 2011. doi: 10.1007/s00158-010-0610-y.