**Noname manuscript No.**
(will be inserted by the editor)

# Interactive topology optimization on hand-held devices

**Niels Aage** · **Morten N. Jørgensen** · **Casper S. Andreasen** · **Ole Sigmund**

Received: date / Accepted: date

**Abstract** This paper presents an interactive topology optimization application designed for hand-held devices running iOS or Android. The TopOpt app solves the minimum compliance problem with interactive control of load and support positions as well as volume fraction. Thus, it is possible to change the problem settings on the fly and watch the design evolve to a new optimum in real time. The use of an interactive app makes it extremely simple to learn and understand the influence of load-directions, support conditions and volume fraction. The topology optimization kernel is written C# and the graphical user interface is develop using the game engine Unity3D. The underlying code is inspired by the publicly available 88 and 99 line Matlab codes for topology optimization but does not utilize any low-level linear algebra routines.

The TopOpt App can be downloaded on iOS devices from the Apple App Store, At Google Play for the Android platform, and a web-version can be run from `www.topopt.dtu.dk`.

**Keywords** Interactiveness · Topology optimization · Smartphones · Tablets

The authors acknowledge the support from the Villum foundation through the NextTop project.

N. Aage · C.S. Andreasen · O. Sigmund
Department of Mechanical Engineering, Solid Mechanics,
Technical University of Denmark, Nils Koppels Alle, B.404,
DK-2800 Kgs. Lyngby, Denmark
E-mail: naa@mek.dtu.dk

M.N. Jørgensen
Department of Informatics and Mathematical Modelling,
Technical University of Denmark, Asmussens Alle, B.305,
DK-2800 Kgs. Lyngby, Denmark

## 1 Introduction

In Denmark more than 30% of the population owns a smartphone, and it is expected that the percentage is much higher amongst engineering, design and architecture students. Not only do smartphones provide the user with easy access to gadgets such as GPS, gyroscopes, accelerometers, etc., they also contain powerful processing units that can be used for advanced scientific computing and high-level educational tools as demonstrated in this paper.

The topology optimization method is a finite element based structural optimization tool which optimizes the material distribution in a specified design domain in order to maximize stiffness or other objectives, typically subject to a volume constraint. The TopOpt-group has since 1999 been hosting a web-based topology optimization Applet (`www.topopt.dtu.dk`, Applets and Software, Server side applets, Compliance Design) (Tcherniak and Sigmund, 2001). The Applet has been extensively used by engineering, architectural and industrial design students and practitioners as well as a general audience and has by now (March 2012) been run over 210,000 times by more than 13,000 unique users. This code is passive in the sense that the user selects design specifications (design domain, boundary conditions, volume fraction, etc.), presses a submit button and then waits for the optimization to finish (in usually 5-10 seconds) before he sees an animation of the design process on his screen. The code is run on a server and hence the response time depends on the number of active users at any given time and on the speed of the internet connection.

In this paper we present a fully interactive topology optimization application, the TopOpt App, which solves the minimum compliance problem with interactive control of loads and supports as well as volume fraction. The App is developed partially to provide a more interactive user interface than the existing web applet and partially to port the

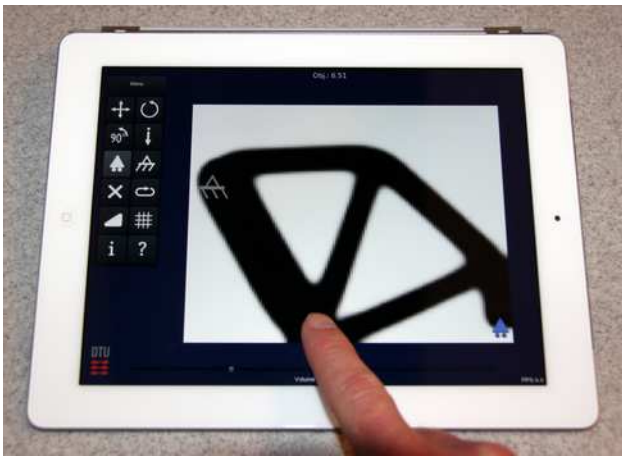
Fig. 1 Picture of the TopOpt App being used on an iPad.

web applet to mobile platforms. Based on an efficient code that is executed directly on the device, and not server-side, the users will have the impression that the structure becomes alive – constantly adapting to varying loads and boundary conditions. With this set-up it is extremely simple to learn and understand the influence of load-directions and support conditions and to develop a general understanding and intuition for structural design by the topology optimization method. A picture of the TopOpt App running on an iPad is shown in figure 1.

The paper is organized as follows. The topology optimization problem is presented in section 2 along with motivation for the chosen approach. Section 3 presents the framework used to develop the TopOpt App with the interactive graphical user-interface (GUI) and the C# optimization kernel. Section 4 presents snapshots of the applet as well as a discussion of the different issues that arise when allowing the optimization problem to be modified on the fly. Section 5 summarizes our findings and gives some directions for future extensions and applications.

## 2 Problem formulation

The minimum compliance problem is a classical topology optimization problem in which the goal is to maximize the stiffness of a structure subject to a constraint on the available material (Bendsøe and Sigmund, 2004). The requirements to the implementation presented in this paper differ slightly from standard implementations, including the ones found in (Sigmund, 2001) and (Andreassen et al., 2010), which otherwise form a basis for the optimizer presented here. The TopOpt App does not only require a fast solver to maintain a high frame rate. More importantly, the solver must be capable of starting from any given 0-1 design, and from this, evolve to a new optimum. Furthermore, the way in which the design evolves when changing loads and supports, must also look and feel right to ensure that the user experiences the structure as being alive and constantly adapting to changing conditions. Although the demands on the optimization solver are somewhat different, the minimum compliance problem remains unchanged and can be stated in a discrete form as

$$\begin{aligned} \min_{\rho \in \mathbb{R}^n} \quad & \phi(\boldsymbol{u}(\boldsymbol{\rho}),\boldsymbol{\rho}) = \boldsymbol{F}^T\boldsymbol{u} \\ \text{s.t.} \quad & \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{F} \\ & V(\boldsymbol{\rho})/V^* - 1 \le 0 \\ & 0 < \rho^{\min} \le \rho_i \le 1, \;\; i = 1,n \end{aligned} \tag{1}$$

where $\boldsymbol{\rho}$ is a vector of $n$ densities, $\phi = \boldsymbol{F}^T\boldsymbol{u}$ is the compliance, $\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{F}$ is the finite element form of Hooke's law, $V(\boldsymbol{\rho})/V^* - 1 \le 0$ is the volume constraint and $\rho^{\min}$ is a lower bound on the design variables. The elasticity equations are solved by the finite element method using four node bilinear elements (see e.g. Zienkiewicz and Taylor, 2000).

### 2.1 Design representation

The design representation for the TopOpt App has to provide the user with a smooth interactive experience, and the design representation should be as fine as possible. To accommodate the smoothness it was by numerous numerical experiments found that the SIMP scheme (Bendsøe, 1989; Zhou and Rozvany, 1991; Mlejnek, 1992) with a high lower bound on both density and stiffness gives the best results. The implemented stiffness interpolation is given as

$$E(\rho_i) = 0.01 + 0.99\rho_i^p \tag{2}$$

where $p = 3$ is the penalization factor. The lower bound on stiffness $E_{min} = 0.01$ is chosen relatively high, since this has shown to yield the best performance when the design has to evolve from one already determined optimum to the next. We have found that a higher value ensures better behavior when a load or a support suddenly is moved to a void region, however, a higher value also adds a risk of introducing artificial stiffness of void regions that may alter the design in undesired manners. The same argument applies to the choice of lower bound for the densities, which is set to $\rho^{\min} = 0.01$. During the development of the TopOpt App the RAMP interpolation scheme (Stolpe and Svanberg, 2001) was tested as an alternative to SIMP. However, at the end, the SIMP scheme was selected for its superior stability.

Though the CPUs of smartphones have evolved tremendously during the past few years, they are still lacking behind the performance of laptop, or desktop, CPUs. The speed ratio for the TopOpt App running on the IPhone 4S, Ipad 2 or a high level laptop is approximately 1:1.2:12. Therefore

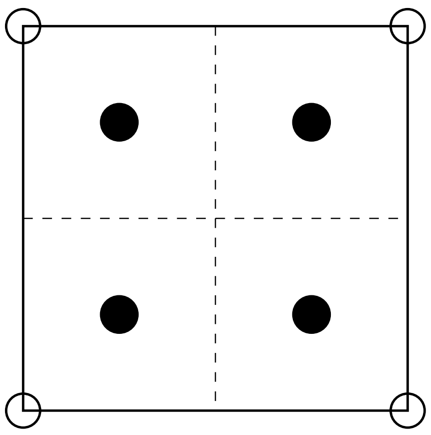

**Fig. 2** Illustration of the multiresolution (MTOP) approach used for the TopOpt App. The full lines represent the displacement element and the circles indicate displacement nodes. The dashes lines show the distribution of design variables within each physical element and the filled circles their location.

we have employed the multiresolution (MTOP) design representation from Nguyen et al. (2010), to obtain a finer design representation with little extra computational cost. The MTOP approach divides every finite element into several design elements. For the TopOpt App we use four design variables for every finite element as illustrated in figure 2. Thus, for each finite element the element matrix contribution can be given as

$$\boldsymbol{K}_e(\rho_e) = \sum_{i=1}^{4} \boldsymbol{K}_0^i \rho_e^i \tag{3}$$

where $\boldsymbol{K}_0^i$ is the reference stiffness matrix evaluated at each of the four design variable locations. Hence, the MTOP approach can be compared to a Gaussian integration with a piecewise constant density variable for each integration point. The major benefit of the MTOP approach is that although the assembly becomes four times more expensive, the size of the linear equation system to be solved remains the same. Since the mesh used is regular, the numerical integration of the four sub matrices can be done once and reused for the all finite elements throughout the iteration process. Although the MTOP approach yields a finer design representation, it is important to note that the checkerboard problem depends on the standard finite element discretization and hence any subsequent filtering must be performed with a radius comparable to the physical element size.

To alleviate checkerboarding we apply the sensitivity filter (Sigmund, 1997). The filter operator can be stated as seen below following the matrix approach from Andreassen et al. (2010)

$$\widehat{\frac{\partial \phi}{\partial \rho_i}} = \frac{1}{\rho_i \sum\limits_{j \in N_i} H_{ij}} \sum_{j \in N_i} H_{ij} \rho_j \frac{\partial \phi}{\partial \rho_j} \tag{4}$$

where $N_i$ is the set of design variables within a radius $r_{\min}$ from design variable $\rho_i$, and the weight factor $H_{ij}$ is given by the sparse matrix

$$H_{ij} = \max(0, r_{\min} - \operatorname{dist}(\rho_i, \rho_j)) \tag{5}$$

where $\operatorname{dist}(\rho_i, \rho_j)$ is the distance from design variable $i$ to variable $j$. The filter radius is set to $r_{\min} = 2.6$ times the design element size (i.e. 1.3 times the finite element size). Note that we use the sensitivity filter since, again by extensive numerical experiments, it tends to provide smoother convergence and is better to avoid getting stuck in local minima compared to the density filter (Bruns and Tortorelli, 2001; Bourdin, 2001).

The design update is performed using the optimality criteria approach as implemented in Andreassen et al. (2010).

Before the optimized design is displayed on screen it is post-processed in the following way. It is first projected onto a two times refined design mesh and subsequently filtered using the standard density filter using a filter radius equivalent to two element sizes on the refined mesh. Finally, to make the interface between void and material sharper, the refined and filtered design is projected using a Heaviside step function as presented in Wang et al. (2011), i.e.

$$\tilde{\rho}_i = \frac{\tanh(\beta\eta) + \tanh(\beta(\rho_i - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))} \tag{6}$$

with a sharpness control of $\beta = 6$ and cut-off point of $\eta = 0.5$. Note that the final steps of filtering and projection are performed outside the optimization framework, and thus simply act as an image processing technique to enhance the design resolution. Basically, the post-processing step corresponds to a smoothed thresholding of the density field.

## 3 Implementation

The TopOpt App has been developed in the Unity3D game engine (Unity Technologies, 2012), which was chosen due to its cross-platform portability and due to the presence of in-house expertise. The multi-platform support means that the same optimization kernel can be used for Android, iOS and web releases with only minor modifications to the layout of the GUI and user-interaction. However, the cross-platform support capability comes at the cost of generality which hinders the use of optimized linear algebra libraries such as BLAS and LAPACK which are only available for certain platforms. Therefore our optimization kernel is built from scratch including sparse libraries, etc.

### 3.1 Optimization kernel

The TopOpt kernel contains an optimization solver equivalent to that presented in the 99 line Matlab code paper (Sigmund, 2001), but is written entirely in C# since this is the language supported by Unity3D. Due to the object oriented nature of C# it is straight forward to implement sparse matrix classes which can be used for both filtering and linear solver. Using sparse matrices for filtering is described in detail in Andreassen et al. (2010), and the major advantage of this approach is that the neighborhood search only has to be done once while assembling the filter matrix. The filter can then be applied as a single sparse matrix-vector product followed by a scaling operation. The sparse matrix approach to filtering is generally faster than performing a quadruple for-loop for each filtering operation as done in the 99 line Matlab code paper (Sigmund, 2001). This is mainly because the for-loop approach requires the evaluation of eq. (4) and subsequently eq. (5) numerous times at each iteration. Especially eq. (5) is expensive since it both involves a max-statement and a squareroot. Note however, that if it was not for the benefit of the simple way to perform filtering using sparse matrices, a banded solver could just as well have been used since our tests have shown that a banded solver matches the performance of the implemented sparse solver.

During the development phase we have also experimented with iterative solvers and the multigrid preconditioned conjugate gradient method in particular. Although this solver can outperform the direct solver when utilizing a smart stopping criteria as described in Arioli (2004) and tricks from reanalysis (Amir and Sigmund, 2011), it yields un-desirable iterates when changing the optimization settings. That is, due to an inexact FE solution the sensitivities in parts of the domain may be wrong, which results in material appearing and disappearing at seemingly random locations in brief glimpses. Since the App is intended to provide a natural transition between optima, the slower, yet more stable, direct solver is used.

Due to the interactive nature of the App, it is very easy to pose a problem which is infeasible or numerically ill-posed. This could for example be a result of missing loads, inadequate supports (singular system), possible free modes (close to singular system) or any combinations of such. If any of the above problems are detected, the optimization solver is frozen and an error message is displayed until the user resolves the problem.

As a final remark our experience have shown that for very coarse discretizations superior stability is achieved for a larger filter radius, e.g. $r_{\min} = 1.4$.

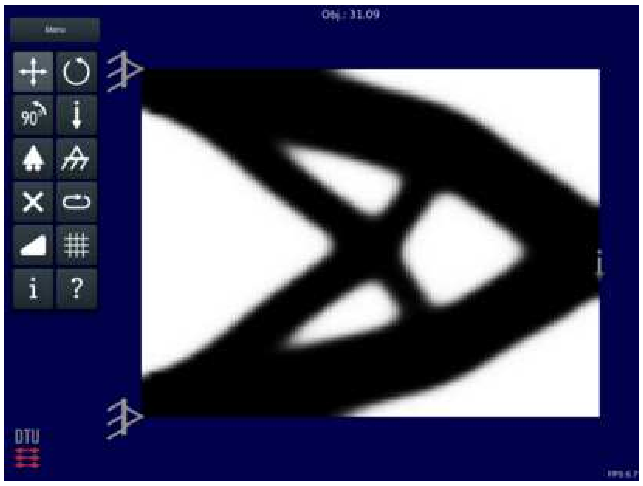

**Fig. 3** Screenshot of the interactive TopOpt App. The menu can be seen to the left and the optimization domain to the right. Centered at the top the current value of compliance is shown. In case of un-physical settings an appropriate error message is shown instead. At the bottom right corner the frame-rate is shown.

### 3.2 GUI

The GUI can be seen in figure 3 and consists of the following items – all conveniently implemented using the Unity3D game engine; The move symbol allows the user to freely move both loads and supports between nodes in the underlying finite element mesh. The two rotation symbols provide the possibility to either rotate forces freely or by 90°, respectively. Supports can also be rotated, although it only has physical meaning for the simple supports. The downward arrow denotes a nodal force with unit magnitude, independent of load orientation. The two supports represent simple and fixed nodal supports, respectively. The tilted cross is used to delete items from the optimization domain, and the white triangle is used to change the volume fraction. All the above mentioned GUI objects are fully interactive, meaning that any modification will change the optimization problem from the next frame (iteration) and on.

The remaining symbols are comprised of the following functionality: A restart feature denoted by an elongated circle, is used to restart the optimization solver with a uniform material distribution equal to that of the current volume fraction. This is needed since the optimizer can, and does, reach local optima, which can be alliviated by a restart. The grid button allows the user to change between a number of fixed mesh resolutions and will restart the entire optimization code when a new grid size is selected. The final two items in the menu yield a short summary of the underlying optimization problem and solution approach and a help menu, denoted with **i** and **?**, respectively.

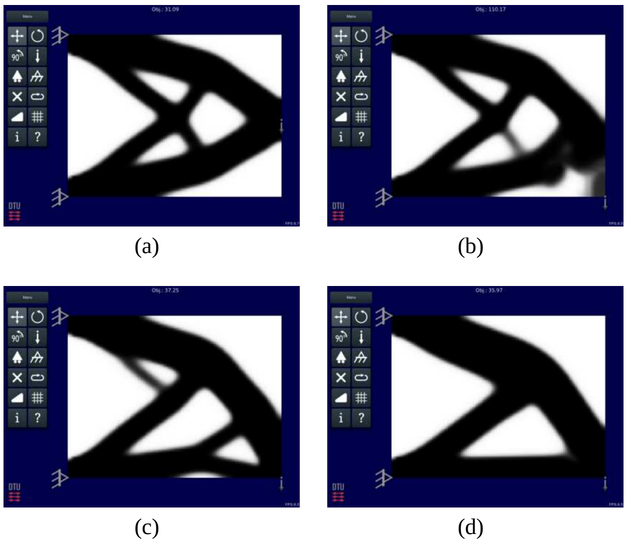

**Fig. 4** The screenshots in (a) through (d) shows how the TopOpt app evolves from one optimum to another.

## 4 Discussion

The TopOpt App is mainly intended for educational purposes, and not as a commercial optimization tool. It can, however, readily be used as inspiration in the early stages of a design process in e.g. architecture, industrial design and engineering. Figure 4 shows a series of screenshots from running the App, demonstrating how the design evolves from one optimum to another. As for most topology optimization problems, the major changes in topology take place within the first 30 design cycles. This means that although the TopOpt App should solve the optimization problem as fast as possible, more than 30 frames per second will hinder the user in following the design evolution. In order to get a reasonable speed we therefore adapt the discretization to the device. For example on an iPhone 4S the standard density mesh is chosen as $88 \times 64$ (i.e. $22 \times 16$ 4 node finite elements), which yields a satisfactory eight frames per second. For slower or faster devices, the user may select coarser or finer discretizations.

## 5 Conclusions

This paper presents a new interactive topology optimization applet, the TopOpt App, for minimum compliance problems. The App can be run on smartphones with iOS and Android, and as a web application. The TopOpt App both demonstrates the capabilities of smartphones in terms of CPU power, but also a new way to perform topology optimization by real-time interaction. The objective of the App is to provide engineering, design and architect students and practitioners with a fast and simple way to use topology optimization. This may lead to a better understanding of optimal material distributions with respect to changes in load conditions, support conditions and the amount of available material. Apart from the educative possibilities, the App may also be interesting to play with for seasoned topology optimization practitioners. For example, it is mind-boggling to see how much a design can change by simply moving a point load or support from the domain corner and one element length into the domain.

The TopOpt App is the first mobile App to offer topology optimization. Future versions will include multiple loads and passive domains as offered by its passive predecessor still found at the `www.topopt.dtu.dk` web-site.

With the publication of this App, we hope to inspire the topology optimization community as well as the mechanics community in general to provide educative Apps that can be used to help students in understanding complex mechanical topics. One may just think of the smartphones' built-in accelerometers and gyroscopes which, with the right App, could provide entirely new and inventive ways of teaching engineering dynamics.

**Acknowledgements** The authors would like to extend their gratitude to the members of the TopOpt and NextTop groups at DTU for their invaluable input on the design and testing of the TopOpt app.

## References

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

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., and Sigmund, O. (2010). Efficient topology optimization in matlab using 88 lines of code. *Structural And Multidisciplinary Optimization*, 43(1):1–16.

Arioli, M. (2004). A stopping criterion for the conjugate gradient algorithm in a finite element method framework. *Numerische Mathematik*, 97:1–24.

Bendsøe, M. (1989). Optimal shape design as a material distribution problem. *Structural Optimization*, 1:193–202.

Bendsøe, M. and Sigmund, O. (2004). *Topology Optimization; Theory, Methods and Applications*. Springer Verlag Berlin Heidelberg New York, 2nd edition.

Bourdin, B. (2001). Filters in topology optimization. *Int. J. Numer. Meth. Engng.*, 50(9):2143–2158.

Bruns, T. E. and 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.

Mlejnek, H. P. (1992). Some aspects of the genesis of structures. *Structural Optimization*, 5:64–69.

Nguyen, T. H., Paulino, G. H., Song, J., and Le, C. H. (2010). A computational paradigm for multiresolution topology optimization (mtop). *Struct Multidisc Optim*, 41:525–539.

Sigmund, O. (1997). On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines*, 25(4):493–525.

Sigmund, O. (2001). A 99 line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21(2):120–127.

Stolpe, M. and Svanberg, K. (2001). An alternative interpolation scheme for minimum compliance topology optimization. *Structural and Multidisciplinary Optimization*, 22(2):116–124.

Tcherniak, D. and Sigmund, O. (2001). A web-based topology optimization program. *Structural and Multidisciplinary Optimization*, 22(3):179–187.

Unity Technologies (2012). Unity3d. `www.unity3d.com`.

Wang, F., Lazarov, B., and Sigmund, O. (2011). On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6):767–784.

Zhou, M. and Rozvany, G. I. N. (1991). The COC algorithm, part II: Topological, geometry and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1-3):309–336.

Zienkiewicz, O. C. and Taylor, R. L. (2000). *Finite Element Method: (parts 1-3)*. Butterworth-Heinemann, fifth edition.