Struct Multidisc Optim (2015) 51:565–572
DOI 10.1007/s00158-014-1157-0

EDUCATIONAL ARTICLE

Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework

Niels Aage · Erik Andreassen · Boyan Stefanov Lazarov

Received: 26 March 2014 / Revised: 7 July 2014 / Accepted: 29 July 2014 / Published online: 17 August 2014

Abstract This paper presents a flexible framework for parallel and easy-to-implement topology optimization using the Portable and Extendable Toolkit for Scientific Computing (PETSc). The presented framework is based on a standardized, and freely available library and in the published form it solves the minimum compliance problem on structured grids, using standard FEM and filtering techniques. For completeness a parallel implementation of the Method of Moving Asymptotes is included as well. The capabilities are exemplified by minimum compliance and homogenization problems. In both cases the unprecedented fine discretization reveals new design features, providing novel insight. The code can be downloaded from www.topopt.dtu.dk/PETSc.

Keywords Topology optimization · Parallel computing · PETSc · Homogenization · Large scale

1 Introduction

The educational aim of this paper is to demonstrate how large scale topology optimization allows for optimization of three-dimensional problems with yet unseen fine discretizations, and to show how this leads to the discovery of new effects in otherwise well-studied design problems. Furthermore, the paper presents a flexible framework for parallel topology optimization, made freely available in order to facilitate the transit to large scale topology optimization for the interested reader. However, the paper is not intended to be a detailed introduction to parallel programming, instead we focus on showing some interesting examples and presenting the framework.

The utilization of parallel processing in scientific computing is constantly increasing, and has also made an impact on the topology optimization community. Within the past decade several works have been published on the subject, see e.g. Borrvall and Petersson (2001), Kim et al. (2004), Vemaganti and Lawrence (2005), Mahdavi et al. (2006), Aage et al. (2008), Evgrafov et al. (2008), Wadbro and Berggren (2009), Schmidt and Schulz (2011), Challis et al. (2013) and Aage and Lazarov (2013). Though many of these works provide schematics on how the parallelization is realized and/or platform specific code, no one provides an easy to use and portable code. In this work we provide such a framework based on the freely available library for high performance and scientific computing PETSc (Balay et al. 2013). Using PETSc reduces the size of the actual implementation, and results in a compact code (compared to our existing in-house parallel optimization codes), which is easy to read, use and extend. Therefore, the presented framework is an ideal development platform for topology optimization problems.

PETSc is an acronym for the *Portable and Extendable Toolkit for Scientific Computing* and it forms the basis for the presented parallel topology optimization code. PETSc is a collection of parallelized (and sequential) libraries that contain most of the necessary building blocks needed for large scale topology optimization, i.e. sparse matrices, vectors, iterative linear solvers, non-linear solvers and time-stepping scheme. The package eliminates the need to write

N. Aage (✉) · E. Andreassen · B. S. Lazarov
Department of Mechanical Engineering, Solid Mechanics, Technical University of Denmark, Nils Koppels Alle, B.404, 2800 Kgs. Lyngby, Denmark
e-mail: naage@mek.dtu.dk

such low-level math libraries and thus speeds up development time by orders of magnitude. The main advantages of PETSc are as follows: a) The libraries are tested extensively and the code is well maintained by specialists. The implementation is parallel scalable to thousands of cores and portable to Linux, UNIX, Mac and Windows. b) The code is written in C in an object oriented manner, such that the base-code is easily extendable and hides the parallel complexity from the user. c) The library is extremely easy to install and use which, combined with the above highlights, makes PETSc well suited for both novices and experts within the field of parallel and scientific computing. The authors acknowledge that several other numerical libraries have similar functionality to PETSc, e.g. Trilinos Heroux et al. (2005). However, PETSc is (one of) the simplest frameworks that provide the basic sparse linear algebra routines, while e.g. Trilinos is a larger and more complex framework, which makes it more difficult to customize. This, and fact that PETSc can interface most other relevant libraries, have made PETSc the obvious choice for the framework.

The framework presented in this work is made freely available and can be downloaded from www.topopt.dtu.dk/PETSc. The package contains the building blocks for conducting large scale, parallel topology optimization of the standard minimum compliance problem on structured grids (Bendsøe and Sigmund 2004) based on the multigrid approach, similar to the one presented in Amir et al. (2014). The code solves a minimum compliance cantilever problem for which an example can be seen in Fig. 1.

In the following sections we present the PETSc based topology optimization framework along with guidelines for usage and extensions. Next we demonstrate the possibilities of the framework by solving two minimum compliance problems as well as two problems from homogenization (minimum Poisson ratio and maximum bulk modulus problems).

2 Code layout, usage and extensions

The main motivation for the development of the proposed topology optimization framework is to have an easy-to-use and easy-to-extend basis for conducting large scale topology optimization of various physical problems. However, to ensure that the publicly available distribution is easily accessible to the topology optimization community, the default problem is a minimum compliance problem.

Basing the code on PETSc means that structured grid generation, partitioning, parallel assemblies, matrix-vector operations, linear and non-linear solvers are readily available. Thus, the focus can be put on the physics and the optimization, and not on the development of high performance linear algebra libraries. The PETSc library can be downloaded from www.mcs.anl.gov/petsc and compiled following the simple instructions on the website.

To provide an overview of the topology optimization framework, a diagram of the code components can be seen in Fig. 2. The foundation of the framework is the PETSc library, and from the diagram one can see that the code we provide consist of five C++ classes and a short main program. The diagram also indicates which classes that need to be modified in order to solve a different optimization problem, i.e.: black boxes mean that modifications are likely to be required and red boxes that no modifications should be applied. The dashed red box indicates that modifications might be necessary, depending on the application. Finally, the code complexity is indicated by the large arrow to the left which points in the direction of less complex code.

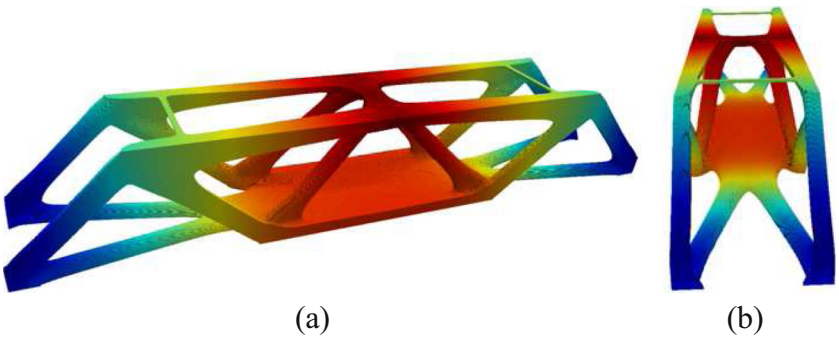

Fig. 1 The classical MMB beam in 3D on a $6 \times 1 \times 1$ domain. The beam is loaded on the line at the center top of domain and have roller supports at the bottom left and right edges. The volume fraction is set to 12 % and the filter radius for the PDE filter is 0.08. The problem is solved using symmetry on a mesh of $1008 \times 336 \times 336$ elements, i.e. a total of 113.8 million design elements and 343.8 million state dofs. The visualized design is thresholded at $\rho_{\text{Phys}} = 0.5$ and colored by the magnitude of the displacement field. The problem was terminated at a design change less than 0.01, which occured after 928 iterations. The problem was distributed on 1800 cores and took a total of 4 hours and 32 minutes to complete

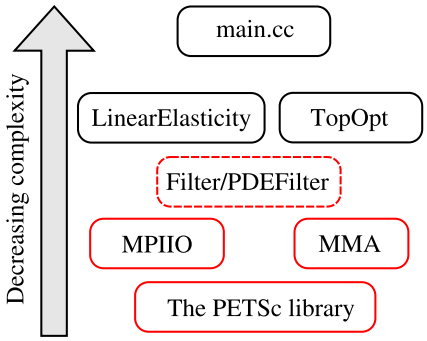

Fig. 2 Layout of the optimization framework. The names in the boxes refer to the main program and to the classes containing physics, optimization settings, filtering, output, MMA and the PETSc library. The box colors indicate whether or not a given class needs modification when changing the problem type, such that red indicates no need for modifications, dashed red that some minor modifications can be needed and black that modifications likely to be are needed. The arrow to the left indicates the complexity of the different components, i.e. the higher in the diagram the simpler the code

2.1 Class structure

Since the goal is to obtain a versatile framework, the classes are divided into clearly separated units. This means for example, that changing the physics only requires modifications of a single class, i.e. the physics class which in the default case solves a linear elasticity problem. A list of the classes with description is given below

- `TopOpt` Contains information on the optimization problem, gridsize, parameters and general settings.
- `LinearElasticity` The physics class which solves the linear elasticity problem on a structured 3D grid using 8-node linear brick elements, see e.g. Zienkiewicz and Taylor (2000). It also contains methods necessary for the minimum compliance problem, i.e. objective, constraint and sensitivity calculations (Bendsøe and Sigmund 2004). The default linear solver is a Galerkin projection multigrid preconditioned flexible GMRES with GMRES/SOR smoothing, which follows the implementation presented in Amir et al. (2014), except for the choice of Krylov method and smoother. An example on how to change the solver is given in the upcoming section.
- `Filter/PDEFilter` are filter classes which contains both sensitivity (Sigmund 1997), density (Bruns and Tortorelli 2001; Bourdin 2001) and PDE (Lazarov and Sigmund 2011) filters through a common interface.
- `MMA` Class containing a fully parallelized implementation of the Method of Moving Asymptotes (MMA) (Svanberg 1987) following the description given in Aage and Lazarov (2013).
- `MPIIO` A versatile output class capable of dumping arbitrary field data into a single binary file.

The distribution also includes Python scripts that conveniently converts the binary outputdata to the VTU format (Schroeder and Martin 2003) that can be visualized in e.g. ParaView version 4 or newer (Ahrens et al. 2005).

2.2 Compiling and running the code

Assuming that PETSc is already installed (see www.mcs.anl.gov/petsc) the user is only required to perform the following six steps to get started on a 64-bit Linux platform.[1] Note that visualization of the results requires Python for postprocessing and ParaView for the actual visualization.

1. Download and extract the code from www.topopt.dtu.dk/PETSc.
2. Modify the `makefile` such that `PETSC_DIR` and `PETSC_ARCH` points to the local PETSc installation on your system.
3. Type `make topopt` in a terminal to compile the code.
4. Type `mpiexec -np 2 ./topopt` to run the code on two processors using the default settings. The optimization and solver settings along with optimization history will be written to the terminal.
5. Prepare the outputdata for ParaView by typing `python bin2vtu #`, where # is the desired timestep. Note that the default settings saves data the first 10 iterations and then subsequently every 10th iteration in order to save space.
6. Visualize in ParaView by typing `paraview *.vtu`.

The default problem is the minimum compliance cantilever problem as described in Aage and Lazarov (2013) on a $2 \times 1 \times 1$ domain using sensitivity filter with radius 0.08 and a volume fraction of 0.12 percent. The contrast between solid and void is set to 10^9, the convergence criteria is $||x_k - x_{k-1}||_\infty < 0.01$ or a maximum of 400 design cycles. Results based on the default settings can be seen on the download page.

2.3 Run time options

The flexibility of the framework allows the user to change a number of settings and parameters at run time. The optimization specific options that can be changed at run time are written to screen before the optimization begins, whereas

[1] For other operating systems please follow the guidelines on www.mcs.anl.gov/petsc. After PETSc is installed, the compilation of the TopOpt application is done similar to that described in Section 2.2.

the PETSc specific options, e.g. linear solver, can be found in the PETSc manual. For example changing the filter to PDE filtering with a radius of 0.2, a volume fraction of 0.3 and a total of 200 iterations is done in the following command

```
mpiexec -np 2 ./topopt -filter 2 -rmin 0.2 \
        -volfrac 0.3 -maxItr 200
```

Due to the flexible construction of PETSc it is possible to change, monitor and test a vast variety of different solver combinations simply by changing the run command. For example, changing the linear solver from the default to a Jacobi preconditioned conjugate gradient method can be obtained as follows

```
mpiexec -np 2 ./topopt -ksp_type cg \
        -ksp_max_it 10000 -pc_type jacobi
```

However, using such a simple solver would result in increased CPU time, especially for large problems.

2.4 Extension

The list of possible extension that can be achieved, using the presented framework as a platform, covers everything from

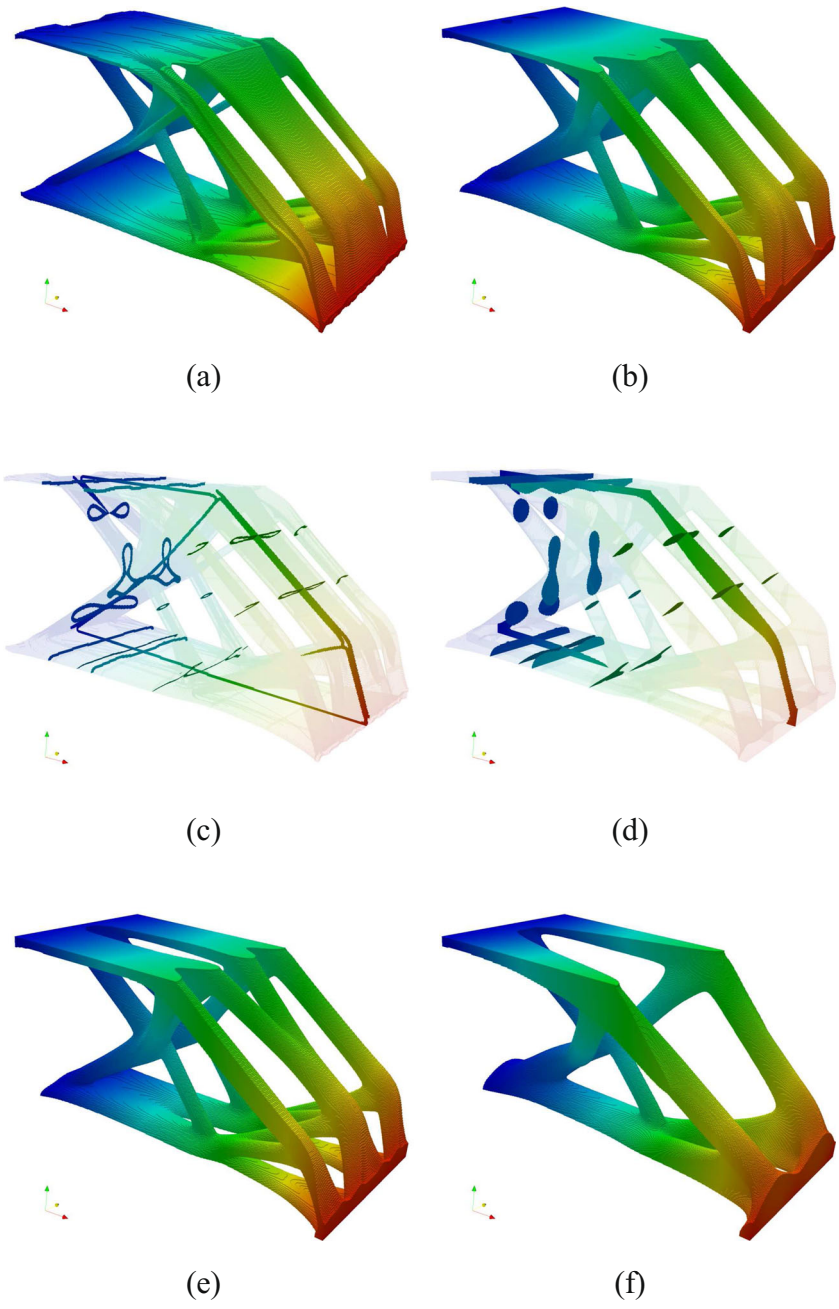

Fig. 3 Optimized cantilever beams on a $2 \times 1 \times 1$ domain allowing 12 % material discretized by 27.6 million elements. The design problems differ through the filter radius such that $r_{\min} = 0.01$ for (**a**, **c**), $r_{\min} = 0.03$ for (**b**, **d**), $r_{\min} = 0.04$ for (**e**) and $r_{\min} = 0.10$ for (**f**). Plot (**c**) and (**d**) shows cross sections of the designs with $r_{\min} = 0.01$ and 0.03, and illustrates the internal holes generated for the smaller filter radius. All the designs are thresholded at $\rho_{\text{Phys}} = 0.5$ and colored by the magnitude of the displacement field

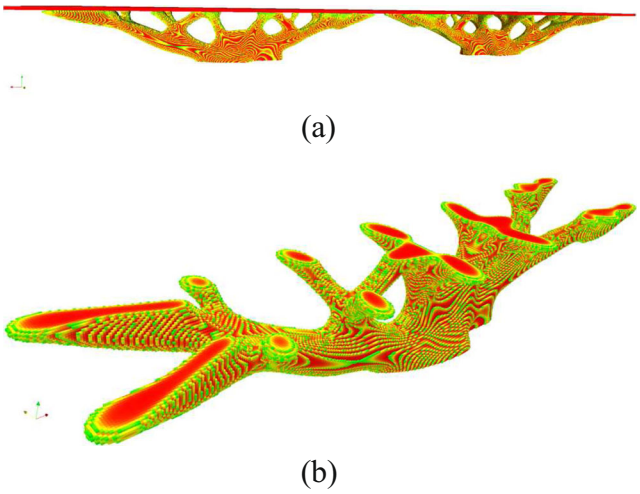

Fig. 4 Minimum compliance design of a roof support, c.f. the Qatar Convention center (Sasaki 2007). The plot in (**a**) shows the full optimized roof structure, while plot (**b**) shows a single of the two support structures. The design domain is due to symmetry reduced to one quarter of the domain seen in (**a**) or half of (**b**). The computational mesh consists of 3.1 million elements. Both plots are thresholded at $\rho_{\text{Phys}} = 0.5$ and colored by the density field

linear to non-linear mechanics, fluid mechanics, acoustics, electromagnetics and generalized multiphysics problems. Describing all aspects of such extensions is outside the scope of this paper, and we will therefore limit this section to discuss how to change the boundary conditions for the minimum compliance problem and give a few general guidelines for switching to a new physical setting.

To solve the minimum compliance problem with different loading and support conditions, changes should only be made to a single method in the physics class, i.e. the `SetUpLoadandBC()` method in `LinearElasticity.h/cc`. This method simply sets the boundary conditions based on coordinates and modifications are therefore straightforward.

The modular composition of the optimization framework means that changing the physics to e.g. fluids, homogenization, etc, only requires that a single new class is written. The coding complexity is further minimized since much of the provided linear elasticity solver can be reused or at least provide inspiration for the new physical problem. To demonstrate the versatility of the framework we also show results from homogenization problems in the example section.

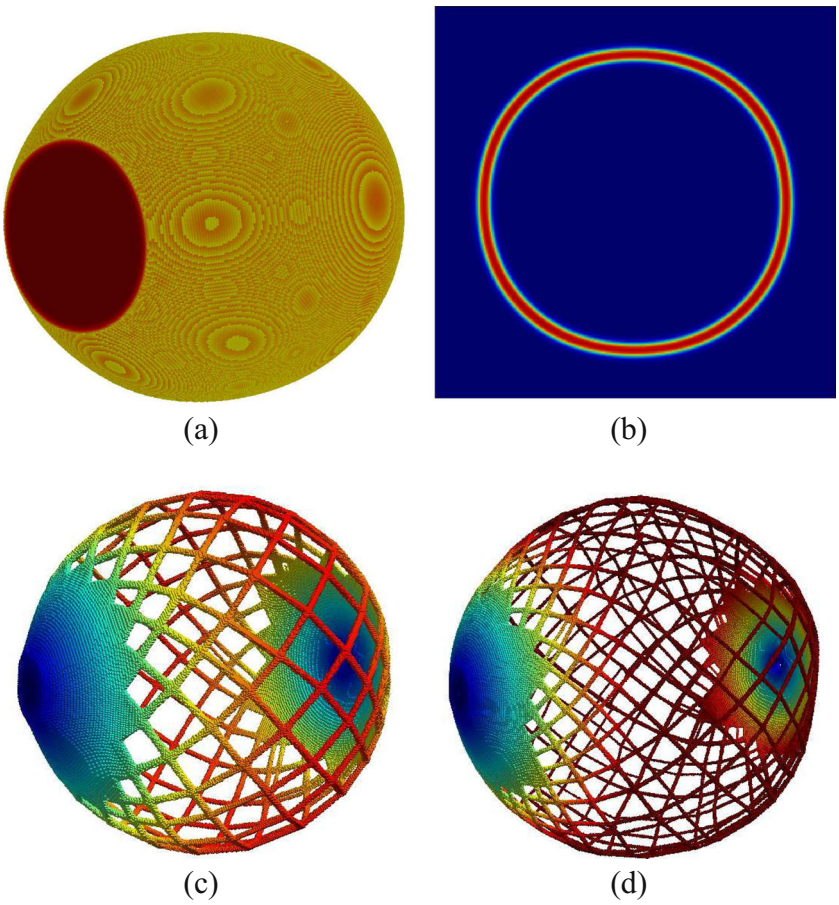

Fig. 5 Optimized designs for a $1 \times 1.2 \times 1.2$ box subjected to pure torsion. Figures (**a**) and (**b**) shows the design, and a slice thereof, for a volume fraction of 10 % and a filter radius of 0.025 when using an uniform initial guess. For the designs in (**c**) and (**d**) the initial guess is a hollow sphere, optimized for a volume fraction of 1 % and a filter radius of 0.003 and a volume fraction of 0.5% and a filter radius of 0.001, respectively. Designs (**a**–**c**) are obtained using a mesh of 26.1 million elements distributed on 1000 cores while the design in (**d**) was run on a mesh of 82.1 million elements on 2000 cores

3 Examples

In the following sections the capabilities of the framework is presented by solving different minimum compliance problems as well as material design problems for maximum bulk modulus and minimum Poison's ratio. Unless otherwise stated, all examples are run on a cluster with a total of 21 nodes, each equipped with two Intel Xeon 5650 6-core CPUs and 48GB memory connected by Infiniband. All problems are solved using the afore mentioned Galerkin projection geometric multigrid preconditioned flexible-GMRES. If not otherwise stated we use four multigrid levels, four GMRES/SOR smoothing steps per level and a relative convergence tolerance of 10^{-5}. The coarse level problem is solved with GMRES/SOR to 10^{-8} or a maximum of 30 iterations. Due to the effectiveness of the multigrid preconditioning strategy, the F-GMRES is never restarted, i.e. for the presented examples all linear solves are obtained with much less than the allowed 200 F-GMRES iterations.

3.1 Minimum compliance

The first example is a re-run of the cantilever problem presented in Aage and Lazarov (2013). In this work we discretize the $2 \times 1 \times 1$ cantilever by $480 \times 240 \times 240$ elements, i.e. 27.6 million design elements and 83.8 million state dofs. The problem is then solved for various filter radii, i.e. from $r_{\min} = 0.01$ to 0.1, with a solid/void contrast of 10^9. The design problems have been run for 1000 design iterations on 24 CPUs (144 cores) yielding an average iteration time varying from 60s to 30s for the smallest and largest filter radius, respectively. The relationship between lengthscale and solution time (i.e. number of iterations for GMRES) is expected for the chosen multigrid preconditioner, since smaller lengthscales are harder to represent on the coarse

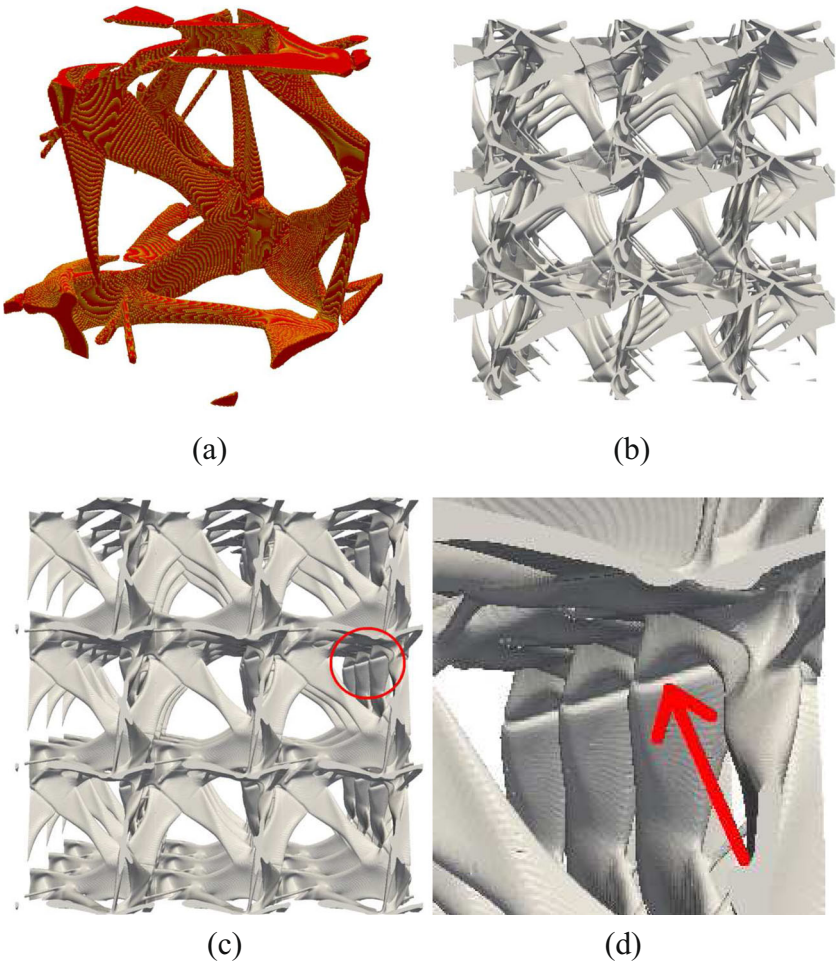

Fig. 6 Optimized 3D isotropic microstructure with a Poisson's ratio of -0.80. The unit cell is discretized by $200^3 = 8$ million elements and is shown in (**a**). The plots in (**b-c**) show $3 \times 3 \times 3$ unit cells from two different angles. The clearest occurence of a "line hinge" is marked by a red circle in (**c**) and a close up is shown in (**d**)

grids. The optimized designs can be seen in Fig. 3. It is especially interesting to note that the high design resolution and small filter radius, leads to a design where the beams contain internal holes.

The second minimum compliance example is the design of a roof support for the Qatar Convention center as described in Sasaki (2007). The design domain consists of a rectangular domain of size $125 \times 15 \times 20$ which, using symmetry conditions, corresponds to half of the design seen in Fig. 4b. The domain is discretized using $512 \times 64 \times 86 = 3.1$ million elements and the problem is solved using 12 CPUs (72 cores). Finally, a volume fraction of 12 % and a filter radius of $r_{\min} = 3.0$ is used to obtain the design shown in Fig. 4. The design of the roof support requires two extensions to the default code: First a passive solid domain of size $125 \times 15 \times 1$, i.e. the roof, is introduced and secondly a Heaviside projection continuation method (Guest et al. 2004) is necessary due to the large filter radius used for the PDE filter.

The third and final minimum compliance example concerns pure torsion of a $1 \times 1.2 \times 1.2$ box. The design problem is inspired by the work of Lewinski (2004) and thus by classical Michell truss layout (Michell 1904). The model is loaded by a moment, i.e. four point forces on one side and clamped on an equivalent area on the opposite surface. Optimized designs for varying filter radii and volume fractions can be seen in Fig. 5. Note that these examples are computed on a bigger cluster with 320 nodes, each with two ten Intel Xeon E5-2650 10-core CPUs. The design in Fig. 5d was obtained using a mesh of 82.1 million elements distributed on 2000 cores, and the wall-clock time for 2000 design cycles was approximately 4.5 hours. The optimized designs show that for a sufficiently small volume fraction the optimal structure is no longer a closed sphere but a truss-like structure, similar to the result presented in Lewinski (2004).

These examples demonstrate how the presented framework can be used to gain new insight into ultra-lightweight structures - even for the otherwise thoroughly studied minimum compliance problem. An indepth examination of these possibilities are left for a subsequent paper.

3.2 Material design

To demonstrate the versatility of the presented framework, the physics is changed from static linear elasticity to elastic material design. With respect to code complexity, this implies that the `LinearElasticity` class is modified to handle periodic boundary conditions, while only minor modifications are needed for `TopOpt` and `Filter` classes. The implementation of the homogenization procedure follows the approach given in Andreassen et al. (2014).

The first design problem is to optimize for isotropic minimum Poisson's ratio materials. The optimized design can be seen in Fig. 6. The Poisson's ratio of the optimized structure is -0.80 computed using a volume fraction of 0.08 and on a mesh of $200^3 = 8$ million elements. A lower bound on the bulk stiffness is assured by requiring the bulk modulus of material structure to be at least 1/4000 of the bulk modulus of the base material. This assures that no single node hinges appear in the design. For filtering the PDE version of the sensitivity filter has been used with an initial radius of 10 %, decreased to a final radius of 2.5 %. It is interesting to note that the high resolution leads to "line hinges", which is something that has not been clearly observed in other 3D topology optimization results because the resolution has been too coarse. A close up of a "line hinge" is shown in Fig. 6d.

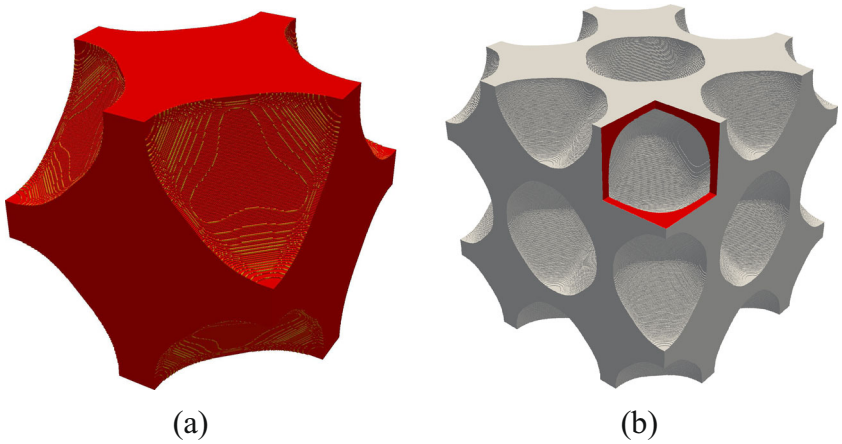

Fig. 7 Optimized maximum bulk modulus unit cell discretized with $288^3 = 23.8$ million elements (**a**) and $2 \times 2 \times 2$ unit cells (**b**). The isotropic and optimized unit cell has a bulk modulus within 99 % of the Hashin-Shtrikman bounds for a volume fraction of 30 %. The red surface in (**b**) indicate that a unit cell has been cut to illustrate that it is indeed hollow

The second material design problem is the maximization of the bulk modulus. The optimized design can be seen in Fig. 7. The volume fraction is 30 %. The resulting bulk modulus is within 99 % of the optimal modulus for porous materials, i.e. the Hashin-Shtrikman bounds. The example is run with a discretization of $288^3 = 23.8$ million elements for which one iteration, i.e. solving six load cases, filtering and design update, takes approximately 60s on 40 CPUs (240 cores).

It should be emphasized that both design problems include an isotropy constraint (see e.g. Andreassen 2014), and that the negative Poisson's ratio structure would be difficult to manufacture due to the small features.

4 Concluding remarks

A framework for large scale topology optimization on structured grids is presented and made publicly available at www.topopt.dtu.dk/PETSc. The framework is based on PETSc and forms an easy-to-use, fully parallelized and open source base for conducting large scale topology optimization. Included in the framework is a number of key building blocks for structural optimization such as filters, a simple, parallelized MMA and input/output classes.

The validity and performance of the framework is demonstrated through a number of numerical examples covering both compliance and material design. It is shown that the framework can be used to solve optimization problems with more than 100 million design variables, and thus that the problem size is only bounded by the amount of computational resources available. The versatility of the framework is exemplified by extensions such as projection methods, passive domains and homogenization problems.

Acknowledgments The authors acknowledge the support from the Villum foundation through the NextTop project, the Danish Research Agency through the innovation consortium F•MAT and the LaScISO project (Grant No. 285782). Fruitful discussions with members of the DTU TopOpt-group are also gratefully acknowledged.

References

Aage N, Lazarov B (2013) Parallel framework for topology optimization using the method of moving asymptotes. Struct Multidiscip Optim 47(4):493–505. doi:10.1007/s00158-012-0869-2

Aage N, Poulsen T, Gersborg-Hansen A, Sigmund O (2008) Topology optimization of large scale stokes flow problems. Struct Multidiscip Optim 35(2):175–180

Ahrens J, Geveci B, Law C (2005) ParaView: an end-user tool for large data visualization. Elsevier

Amir O, Aage N, Lazarov B (2014) On multigrid-CG for efficient topology optimization. Struct Multidiscip Optim 49:815-829. doi:10.1007/s00158-013-1015-5

Andreassen E, Lazarov BS, Sigmund O (2014) Design of manufacturable 3d extremal elastic microstructure. Mech Mater 69(1):1–10. doi:10.1016/j.mechmat.2013.09.018

Balay S, Brown J, Buschelman K, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Smith BF, Zhang H (2013) PETSc users manual. Tech. Rep. ANL-95/11 - Revision 3.4, Argonne National Laboratory

Bendsøe M, Sigmund O (2004) Topology Optimization; Theory, methods and applications, 2nd edn. Springer, Berlin

Borrvall T, Petersson J (2001) Large-scale topology optimization in 3d using parallel computing. Comput Methods Appl Mech Eng 190(46–47):6201–6229

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

Bruns TE, Tortorelli DA (2001) Topology optimization of non-linear elastic structures and compliant mechanisms. Comput Methods Appl Mech Eng 190(26–27):3443–3459

Challis V, Roberts A, Grotowski J (2013) High resolution topology optimization using graphics processing units (GPUs). Struct Multidiscip Optim 49:315–325. doi:10.1007/s00158-013-0980-z

Evgrafov A, Rupp CJ, Maute K, Dunn ML (2008) Large-scale parallel topology optimization using a dual-primal substructuring solver. Struct Multidiscip Optim 36(4):329–345

Guest JK, Prevost JH, Belytschko T (2004) Achieving minimum length scale in topology optimization using nodal design variables and projection functions. Int J Numer Methods Eng 61(2):238–254

Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, Salinger AG, Thornquist HK, Tuminaro RS, Willenbring JM, Williams A, Stanley KS (2005) An overview of the trilinos project. ACM Trans Math Softw 31(3):397–423. doi:10.1145/1089014.1089021

Kim TS, Kim JE, Kim YY (2004) Parallelized structural topology optimization for eigenvalue problems. Int J Solids Struct 41(9–10):2623–2641

Lazarov BS, Sigmund O (2011) Filters in topology optimization based on helmholtz-type differential equations. Int J Numer Methods Eng 86:765–781

Lewinski T (2004) Michell structures formed on surfaces of revolution. Struct Multidiscip Optim 28(1):20–30. doi:10.1007/s00158-004-0419-7

Mahdavi A, Balaji R, Frecker M, Mockensturm EM (2006) Topology optimization of 2D continua for minimum compliance using parallel computing. Struct Multidiscip Optim 32(2):121–132

Michell AGM (1904) The limits of economy of materials in frame structures

Sasaki M (2007) Morphogenesis of flux structure. Architectural Association Publications London

Schmidt S, Schulz V (2011) A 2589 line topology optimization code written for the graphics card. Comput Vis Sci 14(6):249–256. doi:10.1007/s00791-012-0180-1

Schroeder W, Martin K (2003) The Visualization Toolkit. 3rd edn. Kitware Inc

Sigmund O (1997) On the design of compliant mechanisms using topology optimization. Mech Struct Mach 25(4):493–525

Svanberg K (1987) The method of moving asymptotes - a new method for structural optimization. Int J Numer Methods Eng 25

Vemaganti K, Lawrence WE (2005) Parallel methods for optimality criteria-based topology optimization. Comput Methods Appl Mech Eng 194(34–35):3637–3667

Wadbro E, Berggren M (2009) Megapixel topology optimization on a graphics processing unit. SIAM Rev 51(4):707–721

Zienkiewicz OC, Taylor RL (2000) Finite element method: volume 1, 5th edn. Butterworth-Heinemann