# TopOpt in PETSc:
## Exercises in large scale topology optimization

Niels Aage
Department of Mechanical Engineering
Technical University of Denmark

Email: naage@mek.dtu.dk

Group homepages:
www.topopt.dtu.dk
www.camm.elektro.dtu.dk

March 2, 2016

# TopOpt in PETSc

This note describes how to get started with using the TopOpt in PETSc [1] framework on the DTU TopOpt cluster. A number of programming and numerical examination exercises are provided and the goal of the exercises is to get you started with the TopOpt in PETSc framework on distributed memory type clusters which allow you to conduct very large scale optimization.

The components required to use the TopOpt in PETSc framework are (1) PETSc itself [4] (see separate note on using PETSc on the DTU HPC system) and (2) the TopOpt in PETSc code [1]. In relation to the TopOpt Matlab codes (`www.topopt.dtu.dk`), you may think of PETSc as being a high performance version of Matlab and the TopOpt in PETSc code as the 99-line Matlab script. The main difference is that PETSc needs to be compiled and not installed like Matlab, and that the TopOpt in PETSc code should be re-compiled every time you make changes to its core components.

The building blocks, i.e. linear solvers, domain decomposition method and optimization algorithm, used in the framework are described in the following papers. The multigrid preconditioned Krylov solver is similar to that presented in [3], the parallel version of the Method of Moving Asymptotes (MMA) optimization algorithm is described in [2], the framework itself is presented in [1] and the numerical linear algebra backbone PETSc in [4].

On a side note we advise you to have your own installation of PETSc and TopOpt in PETSc framework on your local machine, e.g. your desktop or laptop, such that you can continue your work after the course. Compilation of PETSc [4] and the TopOpt framework is most easily done using a Linux machine, where you can directly follow the guidelines described on `http://www.mcs.anl.gov/petsc/` and `www.topopt.dtu.dk/PETSc`, respectively. Note however, that both Windows and Mac installations are possible but that we do not have experience with this and therefore can not provide you with support.

# PETSc exercises

The following exercises are intended to firstly give a new user an overview of the TopOpt in PETSc framework, i.e. which classes does what and how to modify, and secondly to give an idea of how to modify the code the solve different optimization problems.

Feel free to skip exercises for which you find the solution obvious and move on to the harder problems.

Important notes on cluster usage, PETSc and coding:

- Do not run too large examples. You only have a limited amount of time for the course - use it optimally.
- It is good practise on a cluster that you specify a realistic time limit for your jobs. This helps the scheduler to do its job optimally and hence yourself.
- Be carefull when using the 'standard' filter (both sensitivity and density filters). Both construct a neighborhood matrix, which for large filter radii become extremely memory consuming.
- Remember to make backup of your code for each of the problems posed below.

Problems 1-4 are warm up exercises and are intended to get you started with the TopOpt framework and to let you get familiar with the matrix vector operations in PETSc. Problems 5-8 are more involved and therefore also requires a bigger coding effort.

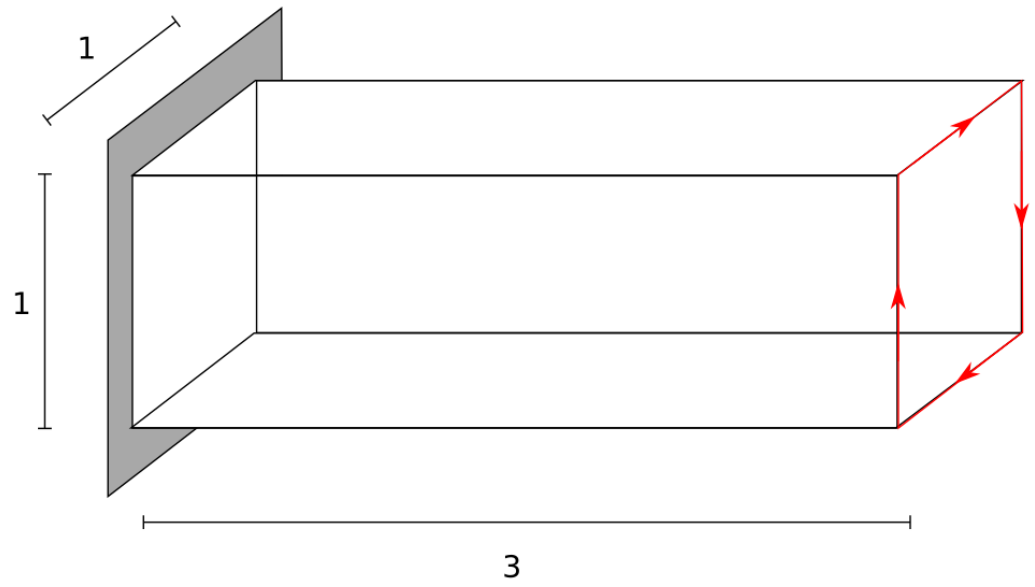
*IMPORTANT: For you to obtain the 2.5 ECTS credits you will have to complete problems 1, 2, 3 and 5. and report your findings as a poster. Upon hand-in you are to send the poster in pdf using the A0 format (corresponding to 16 A4 sheets) to the course responsible. Grades are pass/fail*. Finally, we hope that you will get to try out the more complex problems, since these are are intended to pave the way for using PETSc for design problem in e.g. fluids, thermal/electrical, etc.

## 2.1 Problem 1: Test runtime options

PETSc has a very simple input parsing interface which makes it possible to change most parameters at runtime, i.e. without a recompilation. In this exercise you are to test the influence of the filter type, mesh resolution, restart possibilities, volume fraction and penalization. You can see the flag for the different parameters written to the screen with the suffix −. I.e. changing the filter type from sensitvity to PDE based density filter you should append the run command with `-filter 2`

## 2.2 Problem 2: Changing the loads

Change the loads and solve a problem of your own choice. If you need inspiration you can use the torsion rod problem as sketched in figure 2.1. In the code you should open the `LinearElasticity` class and find the method called `SetUpLoadsAndBC()` in which the loads are set. Run it for different filter radii and resolutions and discuss the results.



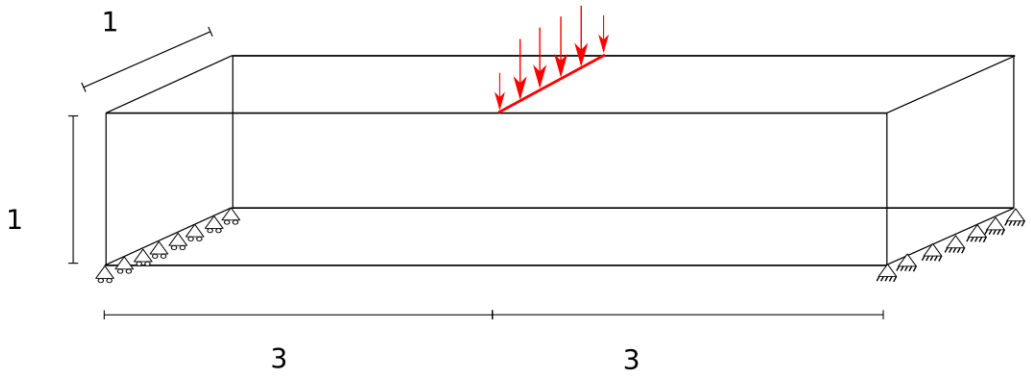

Figur 2.1: Design problem for a torsion rod.

## 2.3 Problem 3: Changing the supports

Solve a 3D MBB beam problem. Change the loads and supports according to figure 2.2. Again, you need to modify the `LinearElasticity` class in the

`SetUpLoadsAndBC()` method. Feel free to use symmetry conditions to reduce the computational domain. Discuss the results and compare the performance to what you observed for the reference problem distributed with the code.

*Note: In ParaView you can use the 'Reflect'-filter to visualize the complete structure when using symmetry.*




Figur 2.2: Design problem for a 3D MBB beam.

## 2.4 Problem 4: Test different solvers and preconditioners

The goal of this exercise is to test out different solution strategies for the linear elasticity problem solved for each iteration in the design process. Play around with solvers and especially different preconditioners and compare the results to the F-GMRES with MG preconditioning that is included in the TopOpt in PETSc framework.

## 2.5 Problem 5: Write an OC class

In this exercise you are to write an optimality criterion (OC) class that can perform design updates based on the constant strain energy optimality condition for stiffness with a weight constraint problems. You may use the 99-line or the 88-line Matlab code as inspiration, see the paper and Matlab code presented in [5]. To complete this exercise you should make sure to perform the following steps.

- Create two new files for the OC class (OC.h and OC.cc).

- Modify the makefile such that the new class is included in the compilation.
- Use Google search to find the PETSc methods required, i.e. minimum of a vector, sum of a vector, etc.

Remember to pass on any required objects that you need for making the OC update. How does the OC perform compared to the MMA implementation provided with the framework ?

## 2.6 Problem 6: Include passive domains

Include the possibility for passive elements (solid or void). For example you can design a roof support structure or a bridge, in which the roof or the deck of the bridge is treated as solid elements that should not be part of the optimization. Feel free to come up with a design problem with passive elements of your own.

There are overall two ways to implement passive elements in the framework.

1 Simple and crued approach: Zero out the relevant design sensitivities before calling MMA and make sure that the corresponding design variables are set to 0/1 before and after the MMA call.

2 Involved but correct approach: Create an extra set of vectors (design variables and sensitivities and possibly the restart vectors for MMA) with a modified index set. This index set should be generated such that only the active design variables are included. You will need to do several modifications to the `TopOpt` class for this approach to work.

Discuss the results and comment on the difference if both methods are implemented.

## 2.7 Problem 7: Multiple loads

Solve a problem with multiple load cases of your own choice. You may solve the multiple loadcase problem as a weighted sum of the compliances or by using a min-max formulation .

Hints:

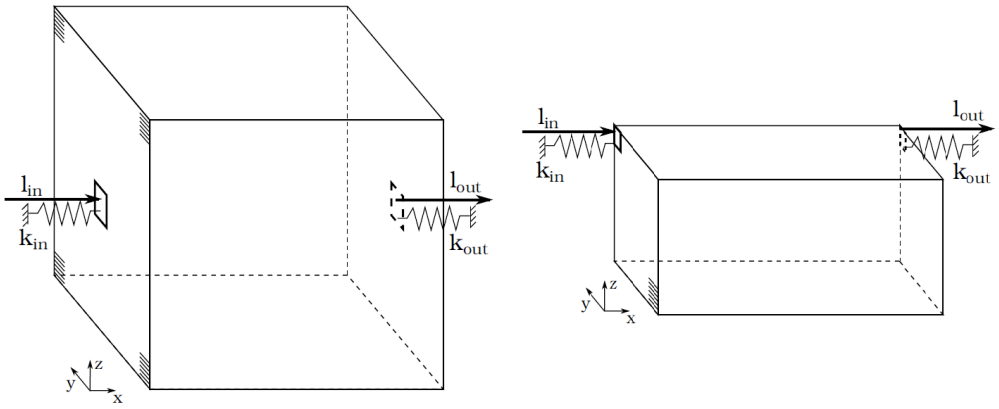
- Use PETSc's multivectors (see e.g. the multivector `dgdx` in the TopOpt class) for storing the loads and displacements. This requires some modifications to the `LinearElasticity` class.
- Use the $z$ variable in MMA to implement the min-max formulation.

## 2.8 Problem 8: Force inverter

Solve the force inverter problem as seen in figure 2.3. You may solve this problem both with and without symmetry conditions. It is recommended to solve the problem with added springs and other dimensions and parameters as given here: full domain $1 \times 1 \times 1$, $E_{\max} = 1$, $E_{\min} = 10^{-9}$, $F_{in}^{\text{total}} = 0.1$, $k_{in} = 1.5$ and $k_{out} = 10^{-5}$. The optimization problem is then stated as (similar to the Matlab exercise sheet).

$$
\begin{aligned}
\min_{\rho \in \mathbb{R}^n} \quad & \phi(\boldsymbol{u}, \boldsymbol{\rho}) = \boldsymbol{L}_{out}^T u \\
\text{s.t.} \quad & \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{F}_{in} \\
& \boldsymbol{L}_{out}^T \boldsymbol{u} + 0.001 \leq 0 \\
& V(\boldsymbol{\rho})/V^* - 1 \leq 0 \\
& 0 < \rho^{\min} \leq \rho_i \leq 1 \ , i = 1, \ldots, n
\end{aligned}
$$

Hints:




Figur 2.3: Design problem for a 3D force inverter. Left: without symmetry. Right: problem with symmetry. (Drawings from Daniel Vestergaard Nielsen's master thesis 2015.)

- Use the MMA method `SetAsymptotes()` to restrict the movement of the asymptotes. For example use (0.2,0.65,1.05) as input to the method.
- Use distributed loads, springs and output area to obtain a better numerical model.

# Bibliography


[1] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. *Struct Multidisc Optim*, 51(51), 2015.

[2] Niels Aage and Boyan S. Lazarov. Parallel framework for topology optimization using the method of moving asymptotes. *Structural and Multidisciplinary Optimization*, 47(4):493–505, 2013.

[3] Oded Amir, Niels Aage, and Boyan S. Lazarov. On multigrid-CG for efficient topology optimization. *Structural and Multidisciplinary Optimization*, pages 1–15, 2013.

[4] Satish Balay, Shrirang Abhyankar, Mark˜F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Victor Eijkhout, William˜D. Gropp, Dinesh Kaushik, Matthew˜G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry˜F. Smith, and Hong Zhang. PETSc Users Manual. Technical Report ANL-95/11 - Revision 3.5, Argonne National Laboratory, 2014.

[5] O. Sigmund. A 99 line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21(1999):120–127, 2001.