Università della Svizzera italiana

Institute of Computational Science ICS

High Performance Topology Optimization

Ezekiel Barnett, Sumeet Gyanchandani, Sameer Rawat, Dimosthenis Pasadakis, Olaf Schenk

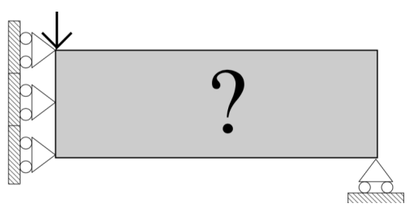
Topology Optimization Problem

Topology Optimization is a mathematical method which optimizes the layout of a material given a set of loads and volume constraints such that global compliance is minimized. The solution algorithm is formulated as an iterative method based around a finite element discretization of the design domain (MBB beam) and an optimization procedure to allocate material such that compliance is minimized.

Compliance Minimization

$$\min_{\boldsymbol{x}}: \quad c(\boldsymbol{x}) = \boldsymbol{u}^T \boldsymbol{K} \boldsymbol{u} = \sum_{e=1}^{N} (x_e)^p \boldsymbol{u}_e^T \boldsymbol{k}_e \boldsymbol{u}_e$$

$$\text{subject to:} \quad \frac{V(\boldsymbol{x})}{V_0} = V_f \ , \ \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \ ,$$
$$\mathbf{0} < \boldsymbol{x}_e \leq \mathbf{1}$$



where,

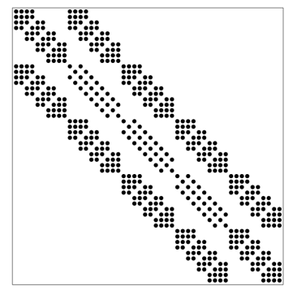
$c(\boldsymbol{x})$	= Compliance,
$\boldsymbol{u}$	= Global Displacement Vector,
$\boldsymbol{f}$	= Global Force Vector,
$\boldsymbol{K}$	= Global Stiffness Matrix,
$\boldsymbol{u_e}$	= Element Displacement Vector,
$\boldsymbol{k_e}$	= Element Stiffness Matrix,
$\boldsymbol{x_e}$	= Element Design Variable,
N	= Number of elements,
p	= Penalization power,
$V(\boldsymbol{x})$	= Material volume,
V_0	= Design domain volume,
V_f	= Volume Fraction.

Iterative Refinement and Computational Cost

The Topology Optimization algorithm functions by iteratively redistributing volume across the mesh elements such that compliance is minimized. See below for the 2nd, 5th, and 100th iteration of the algorithm on a domain of 200 x 50 elements.

The main computational cost of the Topology Optimization algorithm is solving the linear system in order to compute compliance. The stiffness matrix **K** grows linearly with the number of elements. Also observe the sparsity pattern formed by the stiffness tensor of the 2D quadrilateral elements for a 4×4 mesh.

nelx × *nely*	**K**
10×10	242×242
20×20	882×882
40×40	3362×3362
80×80	13122×13122
160×160	51842×51842
320×320	206082×206082



Topology Optimization Algorithm

Algorithm 1 Topology Optimization

Input: nelx, nely, V_f, p, $r_{\min}$

Output: $c(\boldsymbol{x})$, $\boldsymbol{x}$

function TopOpt(nelx, nely, V_f, p, $r_{\min}$)
  Initialize all elements in $\boldsymbol{x} \leftarrow V_f$
  while not converged **do**
    $\boldsymbol{u} \leftarrow$ **FE(nelx, nely,** $\boldsymbol{x}, \boldsymbol{f}$**)**
    $c(\boldsymbol{x}) \leftarrow 0$
    for every element $\boldsymbol{e}$ in $\boldsymbol{x}$ **do**
      $c(\boldsymbol{x}) \leftarrow c(\boldsymbol{x}) + (x_e)^p \boldsymbol{u_e^T k_e u_e}$
      $\delta c(\boldsymbol{x}) \leftarrow -p(x_e)^{p-1} \boldsymbol{u_e^T k_e u_e}$
    end for
    $\boldsymbol{\delta c(x)} \leftarrow$ **MIF(nelx, nely,** $r_{\min}, \boldsymbol{x}, \boldsymbol{\delta c(x)}$**)**
    $\boldsymbol{x} \leftarrow$ **OC(nelx, nely,** $\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{\delta c(x)}$**)**
  end while
  return $c(\boldsymbol{x})$, $\boldsymbol{x}$
end function

Finite Element Analysis

Solve $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$

Mesh-Independency Filtering

$$\hat{H}_f = r_{\min} - \text{dist}(e, f)$$

$$\widehat{\frac{\partial c}{\partial x_e}} = \frac{1}{x_e \sum_{f=1}^{N} \hat{H}_f} \sum_{f=1}^{N} \hat{H}_f x_f \frac{\partial c}{\partial x_f}$$

Optimality Criteria Optimizer

$$x_e^{\text{new}} = \begin{cases} \max(x_{\min}, x_e - m) \\ \quad \text{if } x_e B_e^{\eta} \leq \max(x_{\min}, x_e - m), \\ x_e B_e^{\eta} \\ \quad \text{if } \max(x_{\min}, x_e - m) < x_e B_e^{\eta} \\ \qquad < \min(1, x_e + m), \\ \min(1, x_e + m) \\ \quad \text{if } \min(1, x_e + m) \leq x_e B_e^{\eta} \end{cases}$$

Topology Optimization consists of 4 main stages which are iterated until some change criterion is satisfied: a FE procedure to determine compliance, then computation of sensitivities, a filtering (MIF) to insure mesh-independence, and a design update through the element densities (OC).

The MIF incorporates a convolution operator $\hat{H}_f$ to create a filter imposing a design restriction to ensure the existence of a solution.

The OC optimizer utilizes a heuristic update based on the given criteria where m is a positive move limit, η is a numerical damping coefficient, B_e is found from the optimality condition and is computed after solving a Lagrange Multiplier problem via a bisection algorithm.

Linear Elasticity Analysis with FEniCS

We simulate forces and deformations on our optimized topology using the FEniCS library in a linear elasticity solver to compute displacements and von Mises stresses for various isotropic materials, with a domain of dimensions 320m x 160m.

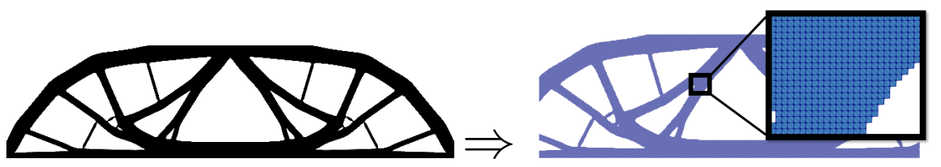
Linear Elasticity Formulation

$$-\nabla \cdot \sigma = \mathbf{f} \in \Omega \tag{1}$$

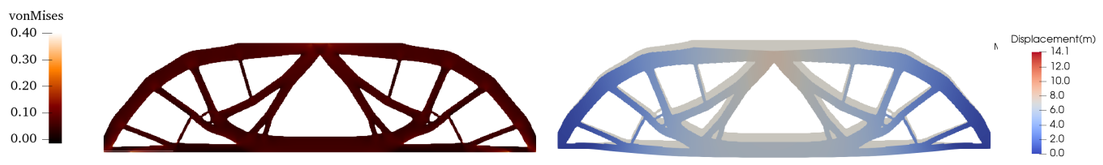
$$\sigma = \lambda(\nabla \cdot \mathbf{u})I + \mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \tag{2}$$

$$a(\mathbf{u}, \mathbf{v}) = \int_{\Omega} \sigma(\mathbf{u}) : \epsilon(\mathbf{v}) dx \tag{3}$$

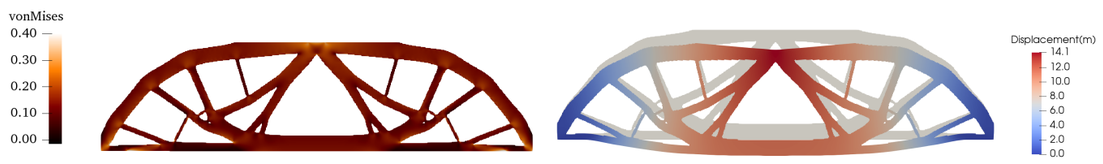
(1) Deformations on the domain $\boldsymbol{\Omega}$ are written as relation between the gradient of the stress tensor σ and a point load $\mathbf{f}$. The stress tensor σ (2) is computed from $\mathbf{u}$, the displacement with Lamé's elasticity parameters μ and λ. FEniCS takes the variational formulation (3) to construct our linear system, solve for displacements, and use these displacements to compute von Mises stresses.



Optimized Topology for MBB problem [left]. Mesh generation using quadrilateral elements [right].



Case	PointLoad(MN)	YoungsMod(GPa)	PoisRatio	MaxDisplace(m)	MinV.M.(GPa)	MaxV.M.(GPa)
SteelASTM-A36	5	200	.31	7.98	.00001	.1938

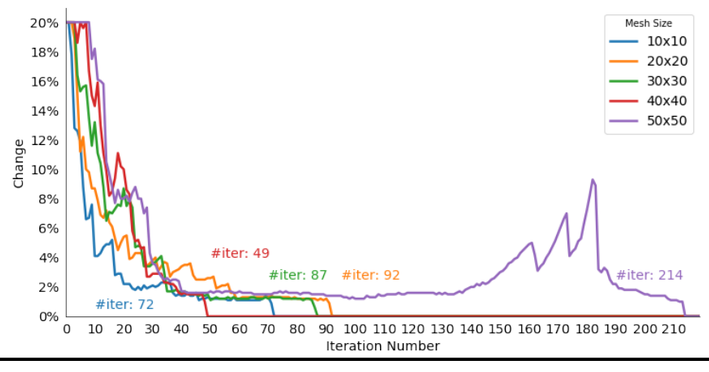


Case	PointLoad(MN)	YoungsMod(GPa)	PoisRatio	MaxDisplace(m)	MinV.M.(GPa)	MaxV.M.(GPa)
TitaniumTi-6Al	5	113.3	0.37	14.45	.0001	.28672

Convergence and Performance Analysis

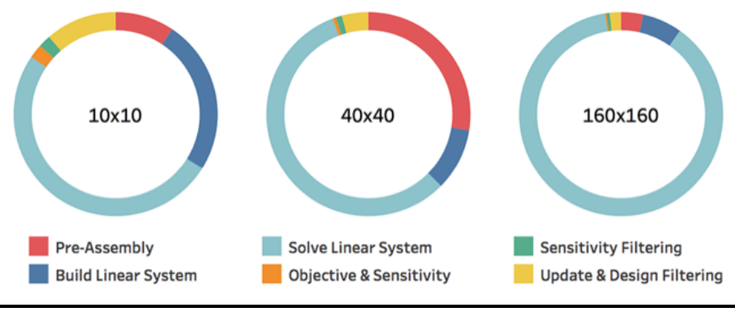
Convergence Analysis

The objective function approaches the minimum rapidly, however the design update process lengthens the convergence significantly resulting in numerous fine-grained changes.



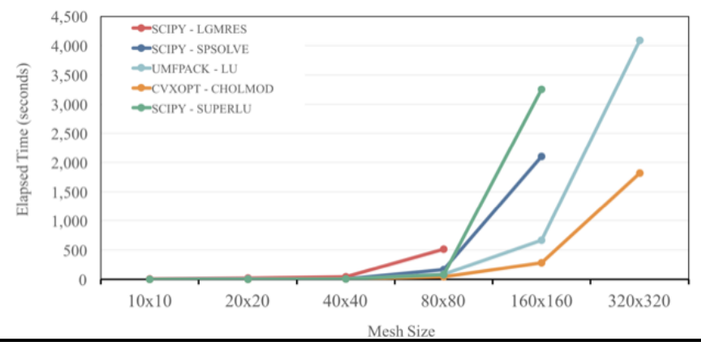
Motivation

The entire procedure consists of six discrete segments. The chart below shows the proportion of each, in comparison to the entire compute time. The computational bottleneck resides in solving the linear system ($\mathbf{Ku} = \mathbf{f}$), and is proportional to an increasing mesh size.



Sparse Linear Solver Choice

The choice of the sparse linear solver to compute compliance has a major impact on the algorithm's overall compute time. We performed experiments on a node of the ICS cluster, which has 2 × Intel E5-2630 v3, 16 (2×8) cores with 128GB DDR4 RAM. Results displayed below.



O. Sigmund, "A 99 line topology optimization code written in Matlab", in *Structural and Multidisciplinary Optimization, 2001*
M. S. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: A Python package for convex optimization, version 1.1.5", *Available at abel.ee.ucla.edu/cvxopt, 2012*
M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes and G. N. Wells, "The FEniCS Project Version 1.5", *Archive of Numerical Software, vol. 3, 2015*