Advanced Solid Mechanics Project

Topology Optimization in a Solid Mechanics Problem

Students: Andrea Gorgi, Gianmarco Boscolo
Professor: Gianluca Mazzucco
19^{th} September 2022

Università degli Studi di Padova

Outline

Project Description

Basic idea: Understand and improve the already existing project.
How:

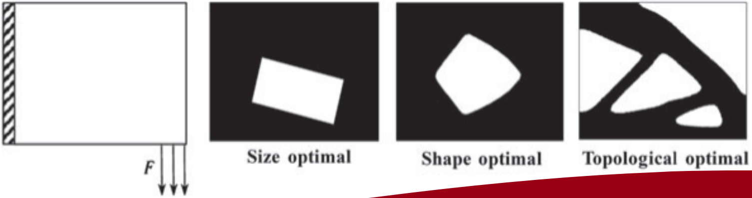
- Clearer implementation
- Generalization of the problem
- Analysis of main drawbacks and issues of the model
- New possible optimization methods

Motivations:

- Incredible tool for design projects
- Growing interest in last decades
- Used in various fields

Three mayor types:

- Size Optimization
- Shape Optimization
- Topology optimization



Starting Project

Author: Artem Mavliutov

SIMP Algorithm

(a) 20×60 V = 0.5 (b) 20×60 V = 0.5 (c) 40×120 V = 0.5

Figure 3: The original SIMP algorithm

(a) 20×60 V = 0.028 (b) 20×60 V = 0.022 (c) 40×120 V = 0.023

Figure 4: The modified SIMP algorithm

Code Issues:

- Fixed Loads and BC;
- Only 1 typology of Elements;
- Geometry related to the number of elements.

Frame Title

$$\frac{\partial c}{\partial t} = f + D\Delta c - v \cdot \nabla c$$

Code Generalization

- Object-Oriented Programming;
- Various Elements;
- Change Geometry;
- Generalized Boundary Conditions;

$\downarrow$

Loads and Constraints can be imposed in every region of the boundary

Loads and BC


```matlab
%% IMPOSE BC
% constraint = [fixX,fixY , xMin,xMax, yMin,yMax]
%----------------
constraints(1,:) = [1,1, 0,0, 0,Ly];
sizeConstraints = size(constraints);
for i = 1:sizeConstraints
    fix = constraints(i,1:2);
    xrange = constraints(i,3:4);
    yrange = constraints(i,5:6);
    Problem = Problem.constraint(fix, xrange, yrange);
end
%---------------------------------------------------------------------------
%% IMPOSE EXTERNAL LOADS
% concentrated load = [ [coordX,coordY], [xLoad, yLoad]]
% distributed load  = [ [xmin, xmax], [ymin, ymax], xLoadDensity,
%                                                    yLoadDensity ]
%-------------------------
Problem = Problem.addConcLoad([Lx,0], [0,1e4]);
%------
Problem = Problem.addDistrLoad([50,100],[100,100],[0,-500000/50]);
```

Stress-Strain Relations

Let's consider σ as the stresses' array, ε as the array of strains and $\mathbf{E}$ the constitutive matrix.
The Stress-Strain relations are stated as:

Stresses = Constitutive Matrix $\times$ Strain + Initial Stresses

$$\sigma = \mathbf{E}\varepsilon + \sigma_0 = \mathbf{E}(\varepsilon + \varepsilon_0). \tag{1}$$

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} + \begin{pmatrix} \sigma_{x_0} \\ \sigma_{y_0} \\ \tau_{xy_0} \end{pmatrix}.$$

The constitutive matrix $\mathbf{E}$ is a symmetric invertible matrix that can represent isotropic or anisotropic properties.

Element Stiffness Matrix

Since the **Strain-Displacement** relation is:

$$\varepsilon = \partial \mathbf{u}$$

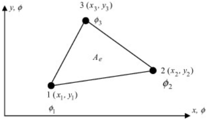
where ∂ is a derivative operator, it follows:

$$\varepsilon = \mathbf{B}\mathbf{d}, \qquad \text{with } \mathbf{B} = \partial \mathbf{N}.$$

The matrix **B** is called the *Strain-Displacement matrix*.
Applying the principle of virtual displacements substituting the new quantities we get the element stiffness matrix:

$$\boxed{\mathbf{k} = \int \mathbf{B}^T \mathbf{E} \mathbf{B} dV = \int_0^t \int_A \mathbf{B}^T \mathbf{E} \mathbf{B} dA d\tau = \int \mathbf{B}^T \mathbf{E} \mathbf{B} dA \cdot t \quad (2)}$$

Constant-strain Triangle (CST)



Let's consider a linear triangular element with nodal coordinates matrix:

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}.$$

Let's now define $x_{ij} := x_i - x_j$ and $y_{ij} := y_i - y_j$. Therefore the area of and element can be calculated as

$$A = \frac{x_{21}y_{31} - x_{31}y_{21}}{2}.$$

Constant-strain Triangle (CST)

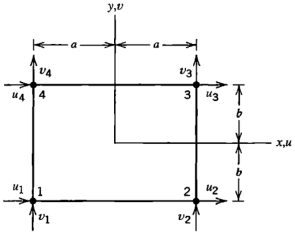
The developed procedure yields to the following strain-displacement matrix for each linear triangular element:

$$\mathbf{B} = \frac{1}{2A}\begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}$$

Now since both **B** and **E** are constant for each element the local stiffness matrix can be calculated as:

$$\text{k}= \int \mathbf{B}^T\mathbf{EB}dV = \mathbf{B}^T\mathbf{EB}\cdot A \cdot t.$$

Bilinear Rectangle (Q4)



Let's consider a linear rectangular element with nodal coords:

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}.$$

Bilinear Rectangle (Q4)

The developed procedure yields to the strain-displacement matrix

$$\mathrm{B}= \frac{1}{4ab}\begin{bmatrix} -(b-y) & 0 & (b-y) & 0 & (b+y) & 0 & -(b+y) & 0 \\ 0 & -(a-x) & 0 & -(a+x) & 0 & (a+x) & 0 & (a-x) \\ -(a-x) & -(b-y) & -(a+x) & (b-y) & (a+x) & (b+y) & (a-x) & -(b+y) \end{bmatrix}$$

where a is the horizontal semi-length and b is the vertical semi-length.

Since **B** is a function of the position to calculate the local stiffness matrix for each element must be performed a quadrature.

Gaussian Quadrature of B

In order to do that it has been chosen to perform a Gaussian quadrature with two points $\xi_{1,2} = \pm\frac{1}{\sqrt{3}}$ and weights $w_{1,2} = 1$, since it's exact for polynomials of degree less or equal to $2n - 1 = 3$.
Let $f(x)$ be a polynomial of degree ≤ 3:

$$\int_{\alpha}^{\beta} f(x)dx = \frac{\beta - \alpha}{2} \sum_{i=1,2} w_i f(\frac{\beta - \alpha}{2}\xi_i + \frac{\beta + \alpha}{2})$$

Now, since $f = f(x, y)$ we can perform a double Gaussian quadrature imposing $\beta = a = -\alpha$ for the *x-integration* and $\beta = b = -\alpha$ for the *y-integration*, obtaining:

$$\int_{-b}^{b}\int_{-a}^{a} f(x, y)dxdy = ab \sum_{i,j=1,2} f(a\xi_i, b\xi_j)$$

Problem:

$$
\begin{cases}
\min_{\gamma} J(\mathbf{u}(\gamma), \gamma) & \\
\text{subject to} & : V(\gamma) \leq V_0 V_{rmax} \text{ Maximum Volume constraint,} \\
 & : V(\gamma) \leq V_0 V_{rmin} \text{ Minimum Volume constraint,} \\
 & : K\mathbf{u} = \mathbf{f}, \qquad \text{Governing equations,} \\
 & : 0 \leq \gamma \leq 1, \qquad \text{Design variable bounds}
\end{cases}
$$

with:

$$0 \leq \gamma_{iel} \leq 1 \sim \text{discretized design variable: material density}$$

$$\mathbf{K}_{iel} = \gamma_{iel}^{p} \mathbf{K}_{iel}^{ideal},$$

Problem Formulation

Main Concern: Need gradient information for fast optimization rules

- Hessian info schemes: faster resolution but high memory demand for the storing
- Heuristic schemes: huge amount of functional evaluations

$$\frac{d}{d\gamma}[J(\mathbf{u}[\gamma],\gamma)] = \frac{\partial J}{\partial \gamma} + \frac{\partial J}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \gamma}.$$

Scheme formulation

I. Choose an initial value for γ (initial material configuration),
II. Solve Governing equations ($K\mathbf{u} = \mathbf{f}$) for the nodal displacements $\mathbf{u}$),
III. Compute derivative of objective function and constraints with respect to gamma:
    1. Evaluate the implicit derivative $\frac{\partial \mathbf{u}}{\partial \gamma}$ with the direct or adjoint method (next chapter)
    2. Compute the gradient of the objective function
IV. Filter the gradient to avoid the checkboard effect
V. Use gradient based- algorithm to update the value of γ to the value that minimizes J based on past iteration history and gradient informations,
VI. Check convergence, if no, go back to step (2), if yes end the process.

Gradient computation: Standard techniques

Compute implicit derivative $\frac{\partial \mathbf{u}}{\partial \gamma}$

- Direct method $\sim$ faster when number of variables less then number of decision functionals

$$\frac{\partial K}{\partial \gamma}\mathbf{u} + K\frac{\partial \mathbf{u}}{\partial \gamma} = \frac{\partial \mathbf{f}}{\partial \gamma}, \tag{3}$$

$$\frac{d}{d\gamma}[J(\mathbf{u}[\gamma], \gamma)] = \frac{\partial J}{\partial \gamma} + \frac{\partial J}{\partial \mathbf{u}} K^{-1}\left[\frac{\partial \mathbf{f}}{\partial \gamma} - \frac{\partial K}{\partial \gamma}\mathbf{u}\right]. \tag{4}$$

- Adjoint method $\sim$ faster when number of functionals less then number of decision variables

$$\frac{\partial J}{\partial \mathbf{u}} = \lambda^T K. \tag{5}$$

Gradient computation: compliance

UNIVERSITÀ DEGLI STUDI DI PADOVA

Maximize the **stiffness**, **minimize** the **work** done by external forces on the system

$$J(\mathbf{u}, \gamma) = \mathbf{u} \cdot \mathbf{f} = \mathbf{u} \cdot K\mathbf{u} = \sum_{iel=1}^{N} \gamma_{iel}^{p} \mathbf{u}_{iel} \cdot \mathbf{k}_{iel} \mathbf{u}_{iel},$$

$$\frac{dJ}{d\gamma_{iel}} = -p\gamma_{iel}^{p-1} \mathbf{u}_{iel} \cdot \mathbf{k}_{iel} \mathbf{u}_{iel}.$$

$$p \geq 3 \sim \textbf{Penalty factor} \tag{6}$$

Optimization Methods

Most used gradient based schemes:

- **Optimality Criteria:** easy implementation and oc met at each iteration, but extremely specific for the compliance problem
- **MMA:** general-purpose algorithm, but efficiency depends on asymptote and move limits $\sim$ parameters calibration
- **Sequential Linear Programming:** linearize functional and constraints with gradient informations. Easy implementation but problems at move limits corners

GOC: Extension of the OC scheme.

UNIVERSITÀ DEGLI STUDI DI PADOVA

$$
\begin{cases}
\min\limits_{\gamma} J(\gamma) = \sum\limits_{iel=1}^{N} \quad (x_e)^p \mathbf{u}_{iel} \cdot \mathbf{k}_{iel} \mathbf{u}_{iel} \\
\text{subject to} \qquad : V(\gamma) = V_0 V_r \quad \text{Volume constraint,} \\
\qquad : K\mathbf{u} = \mathbf{f}, \quad \text{Governing equations,} \\
\qquad : \gamma_{min} \leq \gamma \leq \gamma_{max}, \quad \text{Design variable bounds.}
\end{cases}
\tag{7}
$$

Lagrangian $L(\gamma, \lambda) = J(\gamma) + \lambda(V(\gamma) - V_r V_0)$.
Karush-Kuhn-Tucker first-order optimality conditions

$$
\begin{cases}
\dfrac{\partial L}{\partial \gamma} = \dfrac{\partial J}{\partial \gamma} + \lambda \dfrac{\partial V(\gamma)}{\partial \gamma} = 0 \\
\dfrac{\partial L}{\partial \lambda} = V(\gamma) - V_r V_0 = 0.
\end{cases}
$$

OC: general scheme

Scale factor $D_{iel} = -\frac{\frac{\partial J}{\partial \gamma_{iel}}}{\lambda \frac{\partial V}{\partial \gamma_{iel}}}$ Coupled problem:

- **Inner loop: update** γ_{iel}

$$\gamma_{iel}^{new} = \gamma_{iel}^{old} \sqrt{D_{iel}}, \; \gamma_{iel}^{min} \leq \gamma_{iel}^{new} \leq \gamma_{iel}^{max}.$$

- **Outer loop: update** λ by bisection method based on volume constraint

Main Property: optimality conditions met at each iteration of the optimization algorithm

Method of Moving Asymptotes

P: minimize

$$f_0(\mathbf{x}) \quad (\mathbf{x} \in \mathbb{R}^N),$$

subject to

$$f_i(\mathbf{x}) \leq \hat{f}_i, \quad \mathit{for}\ i = 1, ..., M$$
$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, ..., N,$$

MMA (developed by Kristen Svamberg):

- based on special convex approximation
- solve general non linear programming even for high DOF
- mathematical background on its 'stability'

MMA: general scheme

I. Choose a starting point $\mathbf{x}^0$, and let the iteration index $k = 0$.
II. Given an iteration point $\mathbf{x}^{(k)}$, compute $f_i^{(x^{(k)})}$ and the gradients $\nabla f_i(\mathbf{x}^{(k)})$ for $i = 1, ..., M$.
III. Generate a subproblem $P^{(k)}$ by replacing, in P, the (usually implicit) functions f_i by approximating explicit functions $f_i^{(k)}$, based on the calculations from step 2.
IV. Solve $P^{(k)}$ and let the optimal solution of this subproblem be the next iteration point $x^{(k+1)}$. Let $k = k + 1$ and go to step 2.

$f_i^{(k)} \sim$ linearization in variables of type $\frac{1}{(x_j - L_j)}$ or $\frac{1}{(U_j - x_j)}$, U_j, L_j asymptotes.

Feature: Extends OC to inequality constraints with improved efficiency

$$\begin{cases} \min\limits_{\gamma} J(\gamma) \\ \text{subject to} \quad : g_i(\gamma) \leq 0, \;\; i = 1, ..., NC, \\ \qquad\qquad\quad\;\; : K\mathbf{u} = \mathbf{f}, \quad \text{Governing equations}, \\ \qquad\qquad\quad\;\; : \gamma_{min} \leq \gamma \leq \gamma_{max}, \quad \text{Design variable bounds}, \end{cases} \tag{8}$$

Lagrangian $L(\gamma, \lambda, s) = J(\gamma) + \sum_{i=1}^{NC} \lambda_i (g_u(\gamma) + s_i^2)$.
$s_i \sim$ constraint slack variables

GOC: General idea

Optimality conditions

$$
\begin{aligned}
&\frac{\partial J}{\partial \gamma} + \sum_{i=1}^{NC} \lambda_i \frac{\partial g_i}{\partial \gamma} = 0, \\
&g_i(\gamma) + s_i^2 = 0, \ i = 1, ..., NC \\
&\lambda_i s_i = 0, \ i = 1, ..., NC.
\end{aligned}
\tag{9}
$$

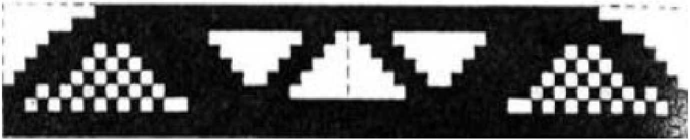
Problem: Coupled equations: need an inner loop for each constraint? **Main idea:** Equations strictly verified only at the end of optimization process

Checkboard effect

Common Topology optimization problems

- **checkerboard pattern**
- **mesh dependency**
- **local minima**

Checkerboard: periodic pattern of high and low values of Pseudo-densities, arranged in a fashion of checkerboards resulting from a numerical instability. Posses artificially high stiffness.



Filtering

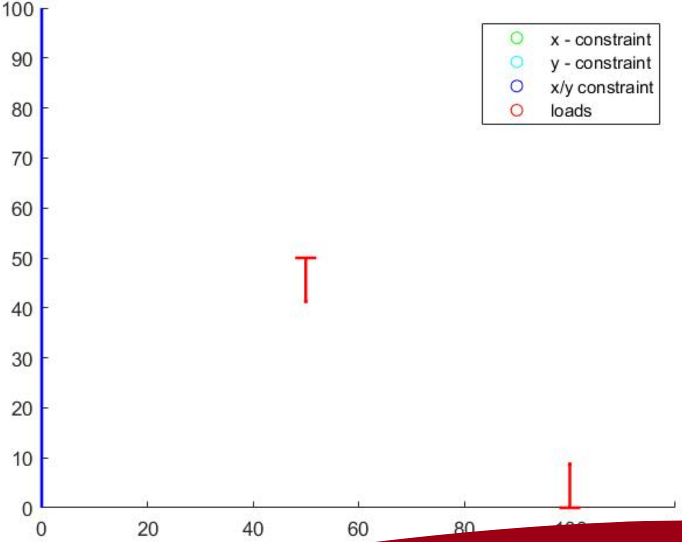
Make elemental material densities neighbour - dependent.

Filtering $\sim$ modify density sensitivity of specific element with weighted average of the element sensitivities in a fixed neighborhood

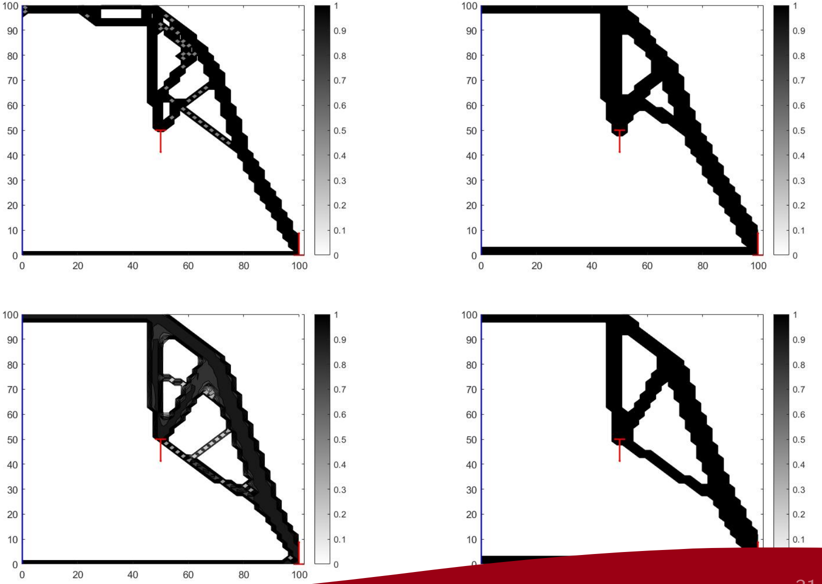
$$\frac{dJ}{d\gamma_{el}} = \frac{1}{\sum_{i=1}^{N} \gamma_i W_i} \sum_{i=1}^{N} W_i \gamma_i \frac{dJ}{d\gamma_i},$$

$W_i = 1/4 * N_{common\ vertices}$

Filtering: Example



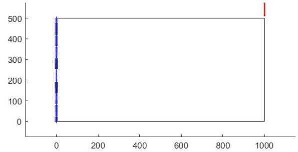
Filtering: Example

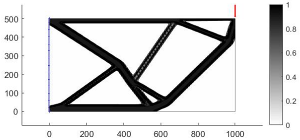


OCM, **MMA** and **GOCM** comparison.

Consider S275 steel ($E = 210000MPa$, $\nu = 0.3$) with full thickness $t = 60mm$, and $Vr_{min} = 0.1$, $Vr_{max} = 0.2$ at the beginning

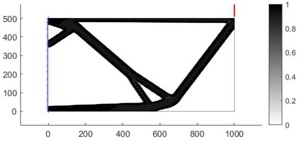
Example 1: Simple supported beam with concentrated load

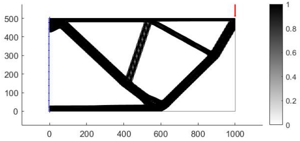




geometry (up), OC (down)

MMA (up), GOC(down)

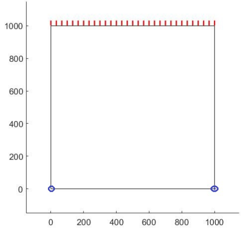


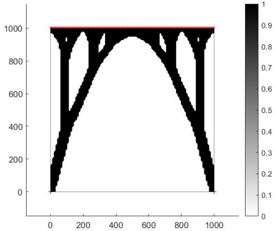


Example 1: Results

	OC	GOC	MMA
$obj[N \cdot mm]:$	$6.9e-3$	$7.5e-3$	$9.8e-3$
$it:$	40	40	40
$wall\ time[sec]:$	76.3	66.96	80.4
$SF:$	0.69	0.5	0.51
$Vr.$	0.2	0.195	0.2

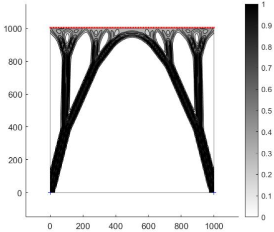
Example 2

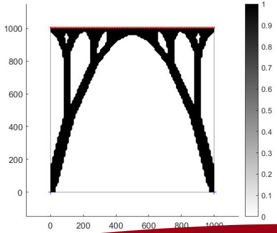




geometry (up), OC (down)

MMA (up), GOC(down)



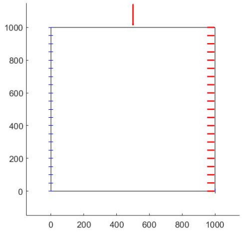


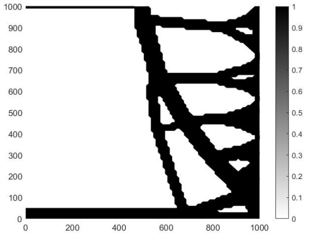
Example 2: Results

Università degli Studi di Padova

	OC	GOC	MMA
$obj[N \cdot mm]$:	$4.7e-2$	$3.6e-2$	$4.3e-2$
it :	60	60	60
$wall\ time[sec]$:	42.1	38.6	50.7
SF :	0.34	0.55	0.65
$Vr.$	0.13	0.18	0.2

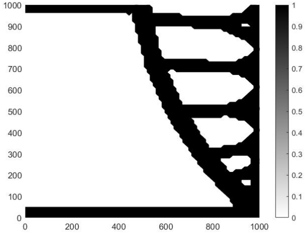
Example 3

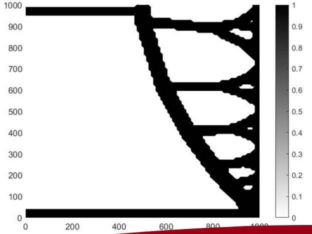




geometry (up), OC (down)

MMA (up), GOC(down)





Example 2: Results

	OC	GOC	MMA
$obj[N \cdot mm]$:	$3.7e-1$	$2.1e-1$	$8.0e-2$
it :	60	60	60
$wall\ time[sec]$:	57.8	42.8	72.7
SF :	0.39	0.74	0.5
$Vr.$	0.25	0.234	0.257

Conclusions

The above results shows that no optimization method is the absolute better.

GOC is the overall faster method, but not surely the most reliable.

The use of different techniques allows a good validation of the obtained result. More control over local minima.

Possible Improvements

- Import Mesh;
- Other types of Elements;
- Other functionals/constraints;
- Comparison with existing optimization solvers