A Sequential Piecewise Linear Programming Algorithm for Topology Optimization

Francisco A. M. Gomes - chico@ime.unicamp.br

Thadeu A. Senne - senne@ime.unicamp.br

DMA - IMECC - UNICAMP

Campinas - SP, Brazil

3^{rd} International Conference on Engineering Optimization

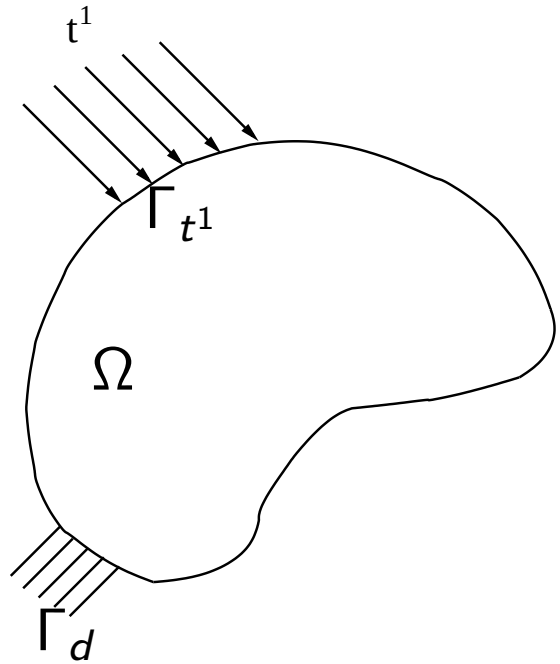
1-5 July, 2012

Rio de Janeiro - RJ, Brazil

*This work is supported by CAPES (mar/2010 - feb/2012) and CNPq (mar/2012 - feb/2013)

Standard Topology Optimization Problem

- Obtain an optimal distribution of material, on a domain Ω
- Maximizing the stiffness of the structure
- Subject to a volume constraint



The SIMP model - Bendsøe (1989)

- The domain Ω is discretized.
- To each element we associate a discrete variable χ that is set to 1 if the element belongs to the structure, or 0 if the element is void.
- Since it is difficult to solve a large nonlinear problem with discrete variables, χ is replaced by a continuous variable $\rho \in [0, 1]$, called the "element's density".
- In order to eliminate the intermediate values of ρ, Bendsøe (1989) introduced the SIMP method (*Solid Isotropic Material with Penalization*), which replaces ρ by the function ρ^p that controls the distribution of material.
- In general, $p = 3$ is sufficient to eliminate intermediate densities.

Small Displacements vs. Large Displacements

- The most part of papers suppose that the relation between strains and displacements is linear (small displacements).
- This assumption is not always valid for some kinds of structures (for example, compliant mechanisms).
- For these structures, it is necessary to consider a nonlinear relation between strains and displacements (large displacements).
- However, because of the difficulties related to the numerical implementation, a small number of papers deal with the large displacements assumption.
- In this work, the structures are under large displacements.

Topology Optimization of Structures
Bendsøe & Kikuchi (1988)

Small displacements

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \mathbf{f}^T \mathbf{u}(\boldsymbol{\rho}) \\ \text{s. t.} \quad & \mathbf{K}(\boldsymbol{\rho})\mathbf{u}(\boldsymbol{\rho}) = \mathbf{f} \\ & \sum_{i=1}^{n_{el}} v_i \, \rho_i \leq V^* \\ & 0 < \rho_{\min} \leq \boldsymbol{\rho} \leq 1 \end{aligned}$$

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \mathbf{u}(\boldsymbol{\rho})^T \mathbf{K}(\boldsymbol{\rho})\mathbf{u}(\boldsymbol{\rho}) \\ \text{s. t.} \quad & \sum_{i=1}^{n_{el}} v_i \, \rho_i \leq V^* \\ & 0 < \rho_{\min} \leq \boldsymbol{\rho} \leq 1 \end{aligned}$$

Large Displacements

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \mathbf{f}^T \mathbf{u}(\boldsymbol{\rho}) \\ \text{s. t.} \quad & \mathbf{K}(\mathbf{u}(\boldsymbol{\rho}), \boldsymbol{\rho})\mathbf{u}(\boldsymbol{\rho}) = \mathbf{f} \\ & \sum_{i=1}^{n_{el}} v_i \, \rho_i \leq V^* \\ & 0 < \rho_{\min} \leq \boldsymbol{\rho} \leq 1 \end{aligned}$$

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \mathbf{u}(\boldsymbol{\rho})^T \mathbf{K}(\mathbf{u}(\boldsymbol{\rho}), \boldsymbol{\rho})\mathbf{u}(\boldsymbol{\rho}) \\ \text{s. t.} \quad & \sum_{i=1}^{n_{el}} v_i \, \rho_i \leq V^* \\ & 0 < \rho_{\min} \leq \boldsymbol{\rho} \leq 1 \end{aligned}$$

Small Displacements

- All the domain elements have the same local stiffness matrix $\mathbf{k}_0$ (it is symmetric), that is calculated only once during all the optimization process.
- Global stiffness matrix: $\mathbf{K}(\boldsymbol{\rho}) = \sum_{i=1}^{n_{el}} \rho_i^p \mathbf{P}_i \mathbf{k}_0 \mathbf{P}_i^T$
- $\mathbf{K}(\boldsymbol{\rho})$ is symmetric and, after imposing the boundary conditions, $\mathbf{K}(\boldsymbol{\rho})$ becomes positive-definite.
- It is necessary to solve the linear system $\mathbf{K}(\boldsymbol{\rho})\mathbf{u}(\boldsymbol{\rho}) = \mathbf{f}$ (static equilibrium conditions) to evaluate the objective function.
- Usually, this linear system is solved using the Cholesky factorization.
- $\mathbf{K}(\boldsymbol{\rho})$ is updated at every global iteration of the optimization algorithm adopted.

Large Displacements

- Issue 1: The local stiffness matrices are different for each domain element, and they depend on the nodal displacements: $\mathbf{k}(\mathbf{u}_i) = \mathbf{k}_0 + \mathbf{k}_L(\mathbf{u}_i)$.
- Global stiffness matrix: $\mathbf{K}(\mathbf{u}(\boldsymbol{\rho}), \boldsymbol{\rho}) = \sum_{i=1}^{n_{el}} \rho_i^p \mathbf{P}_i \mathbf{k}(\mathbf{u}_i) \mathbf{P}_i^T$
- Issue 2: It is necessary to solve the nonlinear system $\mathbf{K}(\mathbf{u}(\boldsymbol{\rho}), \boldsymbol{\rho})\mathbf{u}(\boldsymbol{\rho}) = \mathbf{f}$ (static equilibrium conditions) to evaluate the objective function. This nonlinear system can be solved using the Newton method.
- In the Newton method, fixing a vector of densities $\overline{\boldsymbol{\rho}}$ and giving an initial vector of nodal displacements $\mathbf{u}_0$, we solve the sequence of linear systems

$$\mathbf{K}_T(\mathbf{u}_k, \overline{\boldsymbol{\rho}})\Delta\mathbf{u}_k = \mathbf{f} - \mathbf{K}(\mathbf{u}_k, \overline{\boldsymbol{\rho}})\mathbf{u}_k$$

  and take

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta\mathbf{u}_k$$

  until the condition $\|\mathbf{f} - \mathbf{K}(\mathbf{u}_k, \overline{\boldsymbol{\rho}})\mathbf{u}_k\| < \varepsilon$ becomes true.
- $\mathbf{K}_T(\mathbf{u}_k, \overline{\boldsymbol{\rho}})$: global tangent stiffness matrix (symmetric)

Large Displacements

- Issue 3: The matrices $\mathbf{K}$ e $\mathbf{K}_T$ are updated at every iteration of the Newton method.
- Issue 4: Even imposing the boundary conditions, $\mathbf{K}_T$ cannot be positive-definite during the iterations of the Newton method.
- When it happens, we cannot use the Cholesky factorization to solve the linear systems. For example, we could adopt the LDL^T factorization.
- The Newton method can have a unstable behavior when $\mathbf{K}_T$ is not positive-definite.
- Then, in this case, we could remove the nodes surrounded by void - Buhl, Pedersen & Sigmund (2000) - or apply the arc-length method - Riks (1979), Crisfield (1981).
- But, in order to stabilize the Newton method, we adopted a different approach in this work (scalling the densities).

Scalling the densities

- ρ_i: original density of the i-th element.
- $\boxed{y_i = a\rho_i + b}$: scaled density of the i-th element, where a and b are chosen to map $[\rho_{min}, 1]$ into $[\bar{\rho}_{min}, 1]$ and $\bar{\rho}_{min}$ is the new minimum value for the densities.
- $a = \dfrac{1 - \bar{\rho}_{min}}{1 - \rho_{min}}$ e $b = 1 - a$.
- We use $\rho_{min} = 0.001$ and, for example, $\bar{\rho}_{min} = 0.25$.

Original Problem

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \mathbf{u}(\boldsymbol{\rho})^T \mathbf{K}(\mathbf{u}(\boldsymbol{\rho}), \boldsymbol{\rho}) \mathbf{u}(\boldsymbol{\rho}) \\ \text{s. t.} \quad & \sum_{i=1}^{n_{el}} v_i \, \rho_i \leq V^* \\ & 0 < \rho_{\min} \leq \boldsymbol{\rho} \leq 1 \end{aligned}$$

Scaled Problem

$$\begin{aligned} \min_{\mathbf{y}} \quad & \mathbf{u}(\mathbf{y})^T \mathbf{K}(\mathbf{u}(\mathbf{y}), \mathbf{y}) \mathbf{u}(\mathbf{y}) \\ \text{s. t.} \quad & \sum_{i=1}^{n_{el}} v_i \, y_i \leq a V^* + (1 - a) \bar{V} \\ & 0 < \bar{\rho}_{\min} \leq \mathbf{y} \leq 1 \end{aligned}$$

Sequential Piecewise Linear Programming

Complicated problem (for example, Topology Optimization)

$\Downarrow$

Sequential Quadratic Programming (SQP)

$\Downarrow$

Sequential Quadratic Programming with diagonal Hessian

$\Downarrow$

Sequential Piecewise Linear Programming (SPLP)
Advantage: the subproblems are converted into a LP
Little disadvantage: the number of variables increases

Sequential Piecewise Linear Programming

- Subproblem solved in the SQP method:

$$\begin{aligned} \min_{\mathbf{s}} \quad & \mathbf{w}^T\mathbf{s} + \widehat{\Gamma}(\mathbf{s}) \\ \text{s. t} \quad & \mathbf{A}\,\mathbf{s} = \mathbf{c} \\ & \mathbf{s}_l \leq \mathbf{s} \leq \mathbf{s}_u \end{aligned}$$

- $\widehat{\Gamma}(\mathbf{s}) = \frac{1}{2}\mathbf{s}^T\mathbf{B}\mathbf{s}$, where $\mathbf{B} \in \mathbb{R}^{n\times n}$ is symmetric and semipositive-definite.

Sequential Piecewise Linear Programming

- Choosing **B** as a diagonal matrix, we obtain a separable quadratic

$$\widehat{\Gamma}(\mathbf{s}) = \sum_{i=1}^{n} \gamma_i(s_i)\,, \qquad \text{where} \qquad \gamma_i(s_i) \equiv \frac{1}{2} b_i s_i^2\,.$$

- Each term $\gamma_i(s_i)$ is approximated by a piecewise linear function

$$\Gamma_i(s_i) \;=\; \max_{j \in \{0,\,\ldots,\,2r\}} \left\{ (b_i\, t_i^{(j)})\, s_i - \frac{1}{2} b_i (t_i^{(j)})^2 \right\}$$

that interpolates γ_i e γ_i' at the points $t_i^{(j)}$, such that

$$\Gamma(\mathbf{s}) \equiv \sum_{i=1}^{n} \Gamma_i(\mathbf{s}_i) \qquad \text{and} \qquad \Gamma(\mathbf{s}) \approx \widehat{\Gamma}(\mathbf{s})\,,$$

with $\Gamma(\mathbf{s})$ convex and nonnegative.

- We take the interpolation points $t_i^{(j)} \in [L_i,\, U_i]$ such that

$$\mathbf{s}_{l_i} \;\leq\; L_i \;\leq\; U_i \;\leq\; \mathbf{s}_{u_i}\,.$$

Sequential Piecewise Linear Programming

- Piecewise linear problem:

$$\begin{aligned} \min \quad & \mathbf{w}^T\mathbf{s} + \Gamma(\mathbf{s}) \\ \text{s. t} \quad & \mathbf{A}\,\mathbf{s} = \mathbf{c} \\ & \mathbf{s}_l \leq \mathbf{s} \leq \mathbf{s}_u \end{aligned}$$

- Byrd *et al.* (2011): to ensure that $\mathbf{w}^T\mathbf{s} + \Gamma(\mathbf{s})$ is bounded below, we adjust the bounds L_i and U_i such that $[L_i, U_i]$ also contains the minimizer of the quadratic

$$q_i(s_i) = \frac{1}{2} b_i s_i^2 + w_i s_i \,.$$

- Noting that

$$\mathbf{w}^T\mathbf{s} + \Gamma(\mathbf{s}) = \sum_{i=1}^{n} [w_i s_i + \Gamma_i(s_i)]$$

and remembering that we use $2r+1$ interpolation points, we observe that each term $w_i s_i + \Gamma_i(s_i)$ is composed by $2r+1$ line segments.

Sequential Piecewise Linear Programming

- Change of variables: $\mathbf{s} = \sum_{j=0}^{2r} \boldsymbol{\delta}_j$, where each new variable $\delta_{i,j}$ is associated to the j-th line segment of the graph of $\Gamma_i(s_i)$.

- Then, the piecewise linear problem is converted into the LP

$$
\begin{aligned}
\min \quad & \sum_{j=0}^{2r} \boldsymbol{\alpha}_j^T \boldsymbol{\delta}_j \\
\text{s. t} \quad & \mathbf{A} \sum_{j=0}^{2r} \boldsymbol{\delta}_j = \mathbf{c} \\
& \mathbf{s}_l \leq \boldsymbol{\delta}_0 \leq \mathbf{z}_0 \\
& \mathbf{0} \leq \boldsymbol{\delta}_j \leq \mathbf{z}_j - \mathbf{z}_{j-1}, \qquad j = 1, \ldots, 2r-1 \\
& \mathbf{0} \leq \boldsymbol{\delta}_{2r} \leq \mathbf{s}_u - \mathbf{z}_{2r},
\end{aligned}
$$

that has $(2r+1)n$ variables.

Sequential Piecewise Linear Programming

- General optimization problem:

$$\begin{array}{ll} \min & f(\mathbf{x}) \\ \text{s. t} & \mathbf{c}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \end{array}$$

  where $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$ are functions with first partial derivatives Lipschitz continuous.

- Normal step (solution of a LP - infeasibility reduction)
- Tangent step (solution of a piecewise LP - objective function reduction)
- Trust regions (to ensure the global convergence of the algorithm)
- Merit function (to decide if the new point will be accepted or rejected)

Sequential Piecewise Linear Programming

1. Normal step $\mathbf{s}_n$ - Infeasibility reduction

- Solve the LP

$$\begin{aligned} \min \quad & \bar{M}(\mathbf{x}, \mathbf{s}, \mathbf{z}) = \mathbf{e}^T \mathbf{z} \\ \text{s. t.} \quad & \mathbf{A}(\mathbf{x})\mathbf{s} + \mathbf{E}(\mathbf{x})\mathbf{z} + \mathbf{c}(\mathbf{x}) = \mathbf{0} \\ & \max\{-0.8\Delta\,,\ \mathbf{x}_l - \mathbf{x}\} \leq \mathbf{s} \leq \min\{0.8\Delta,\ \mathbf{x}_u - \mathbf{x}\} \\ & \mathbf{z} \geq \mathbf{0} \end{aligned}$$

2. Tangent step $\mathbf{s}_c$ - Objective function reduction

- If $\bar{M}(\mathbf{x}^{(k)}, \mathbf{s}_n, \mathbf{z}) = 0$, solve the piecewise linear problem

$$\begin{aligned} \min \quad & \nabla f(\mathbf{x})^T \mathbf{s} + \Gamma(\mathbf{s}) \\ \text{s. t.} \quad & \mathbf{A}(\mathbf{x})\mathbf{s} + \mathbf{c}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{s}_l \leq \mathbf{s} \leq \mathbf{s}_u \end{aligned}$$

  that is converted into a LP .

- Otherwise, set $\mathbf{s}_c \leftarrow \mathbf{s}_n$.

Computational Results

- Comparison between the Sequential Piecewise Linear Programming (SPLP) proposed here and the globally convergent version of the Sequential Linear Programming (SLP) presented by Gomes & Senne (2011).
- We used 3 interpolation points ($r = 1$) for each variable in the SPLP algorithm.
- The penalty parameter p of the SIMP model was set to 1, 2 and 3, consecutively, combined with the weighted mean density filter - Bruns & Tortorelli (2003).
- The linear systems of the Newton method were solved using the package CHOLMOD 1.7 in C++ - Davis (2008)

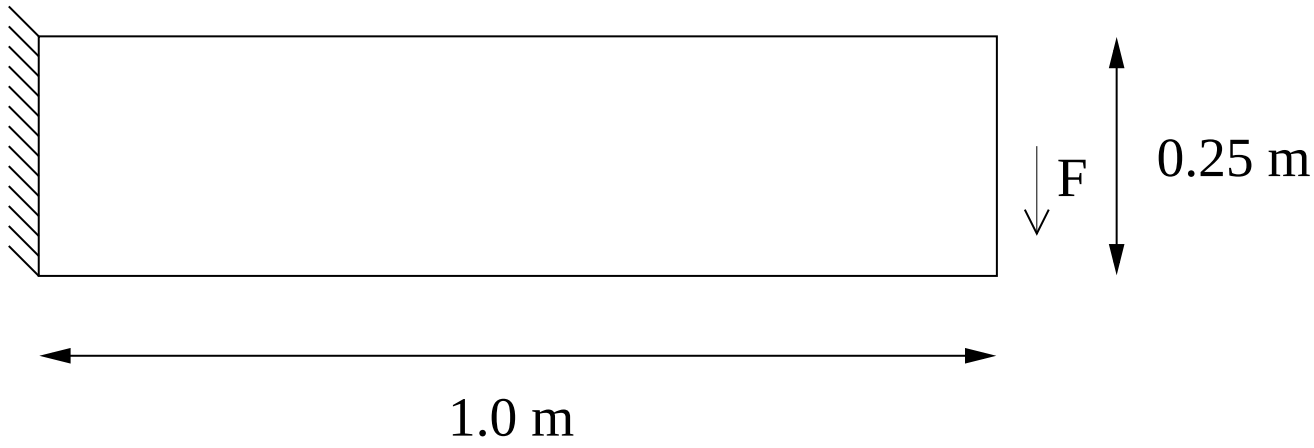
Computational Results

- The results were obtained by solving the scaled problems.
- The LP subproblems were solved using the package CPLEX 12.1 in C++.
- Stopping Criterion: $\|g_P(\mathbf{x}^{(k)})\|_\infty < 10^{-3}$, where $g_P(\mathbf{x}^{(k)}) =$ projected gradient of the objective function onto the null space of the constraints, solution of

$$\begin{array}{ll} \min & \frac{1}{2}\mathbf{d}^T\mathbf{d} + \nabla f(\mathbf{x})^T\mathbf{d} \\ \text{s. t.} & \mathbf{A}(\mathbf{x})\mathbf{d} = \mathbf{0} \\ & \mathbf{s}_l \leq \mathbf{d} \leq \mathbf{s}_u \end{array}$$

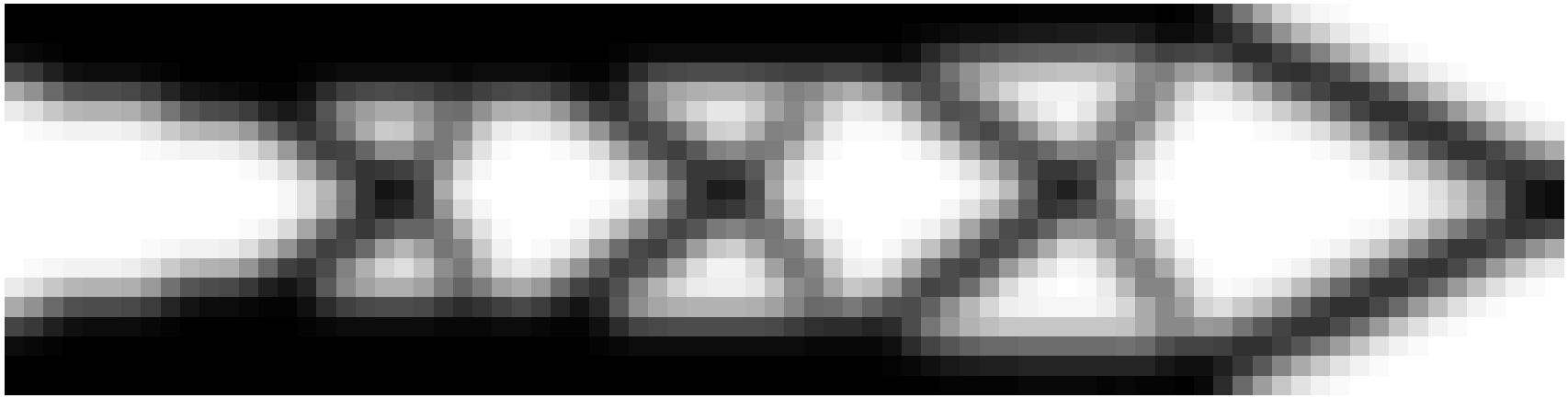
or maximum number of iterations $= 1600$.

Example 1: Cantilever beam
Buhl, Pedersen & Sigmund (2000)



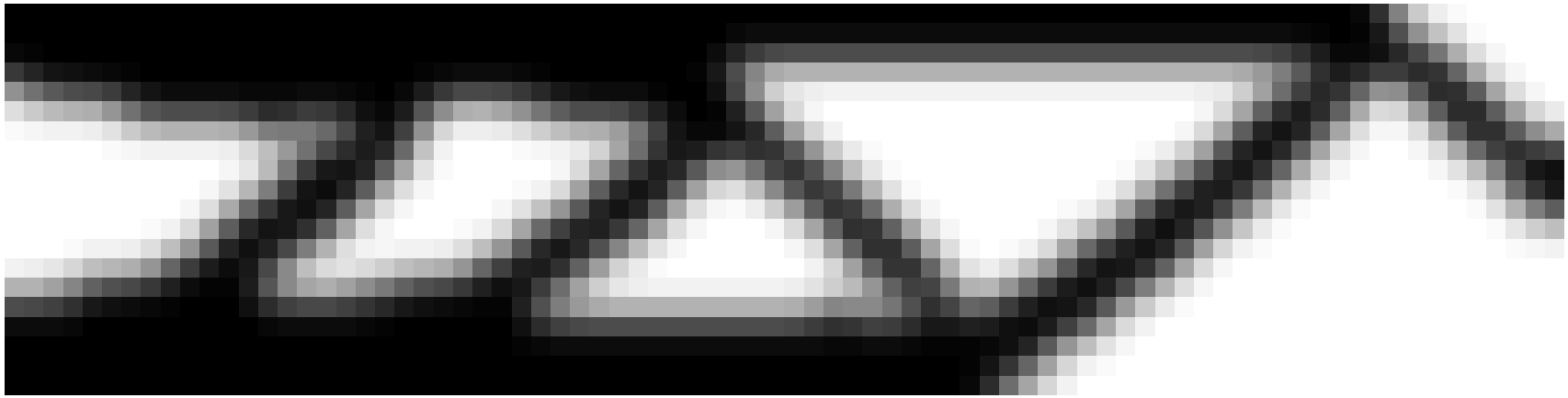
- $F = 12000$ and $240000\,N$
- Thickness: $e = 0.1\,m$
- Young modulus: $E = 3.0 \times 10^9\,N/m^2$
- Poisson coefficient: $\nu = 0.4$
- Volume fraction $= 50\,\%$
- Domain discretized in 1600 rectangular finite elements

F = 12000 N



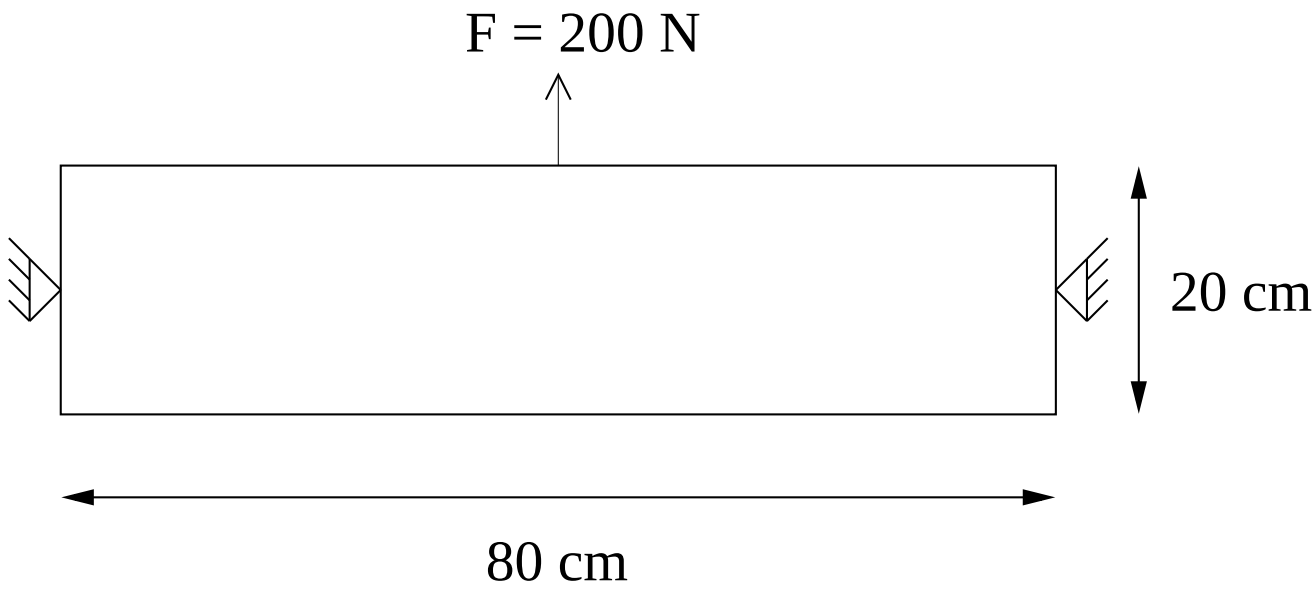
	SPLP	**SLP**
obj. func.	1.9620×10^2	1.9616×10^2
ext. iter.	597	517
int. iter.	610	528
time (sec.)	125.02	109.34

F = 240000 N



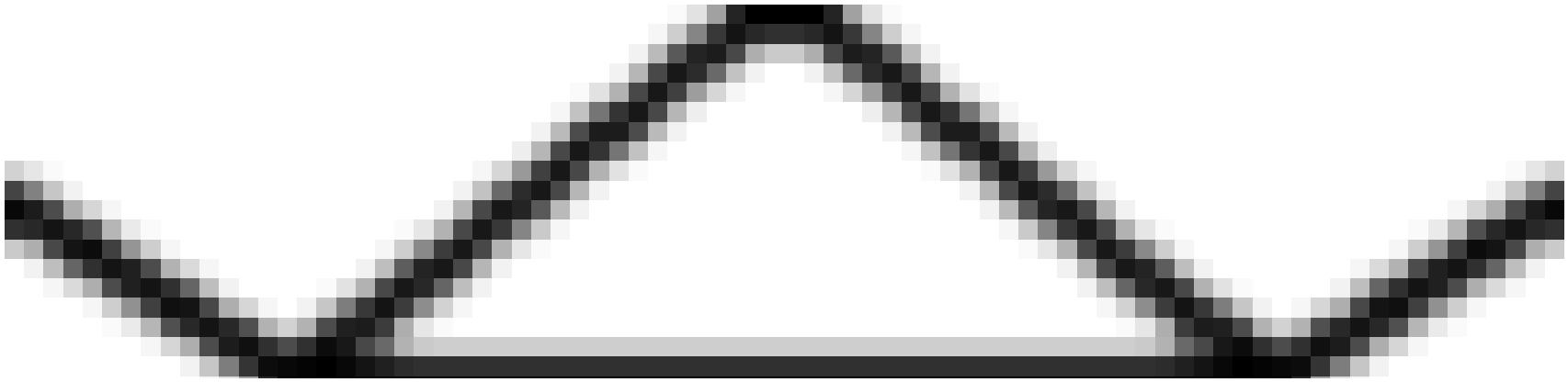
	SPLP	SLP
obj. func.	7.0280×10^4	7.0295×10^4
ext. iter.	296	491
int. iter.	311	503
time (sec.)	81.35	126.22

Example 2: Plate - Gea & Luo (2001)

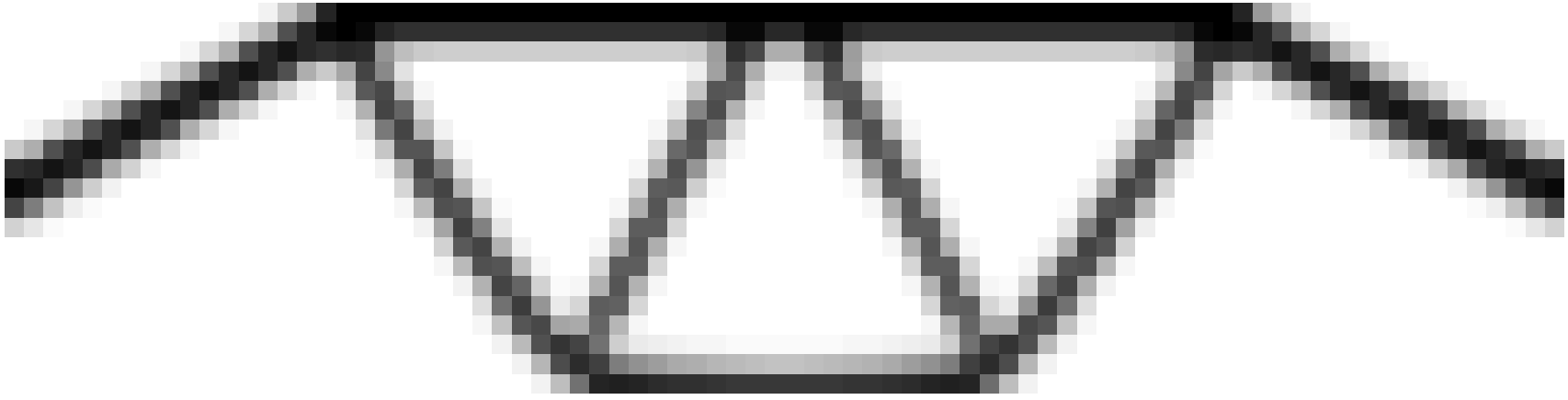


- Thickness: $e = 0.1\,cm$
- Young modulus: $E = 1.0 \times 10^5\,N/cm^2$
- Poisson coefficient: $\nu = 0.3$
- Volume fraction: $= 20\,\%$
- Domain discretized in 1600 rectangular finite elements

Small displacements



Large displacements



	Small displ.		Large displ.	
	SPLP	**SLP**	**SPLP**	**SLP**
obj. func.	5.0983×10^2	5.0983×10^2	4.3158×10^2	4.3414×10^2
ext. iter.	159	190	661	890
int. iter.	172	193	675	897
time (sec.)	7.64	6.95	136.95	167.91

Conclusions and Future Work

- The results show that the SPLP algorithm presented here is promising and competitive in relation to the SLP one.
- Maybe it is necessary to perform a fine tuning on choosing the interpolation points, the diagonal Hessian approximation of the objective function and the update scheme of the trust region radius to improve the performance of the SPLP algorithm.
- We will prove the global convergence property of the SPLP algorithm and make tests for compliant mechanisms.

Thanks a lot!!!