FRIAS
FREIBURG INSTITUTE FOR ADVANCED STUDIES
ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG

A 99 Line topology optimization code written by Ole Sigmund in Matlab

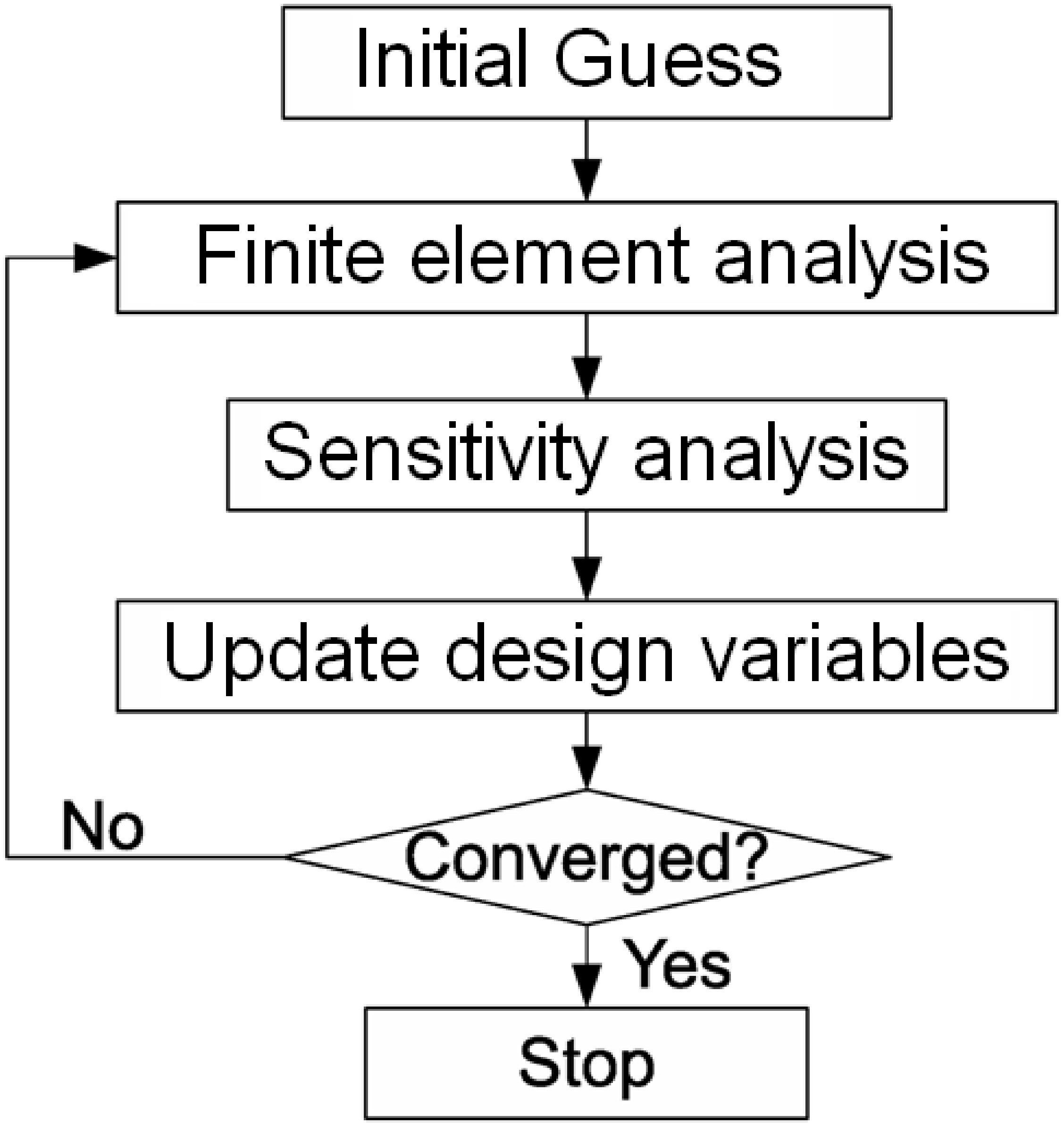
Feng Jia
School of Soft Matter Research
Freiburg Institute for Advanced Studies
University of Freiburg, Germany

Outline

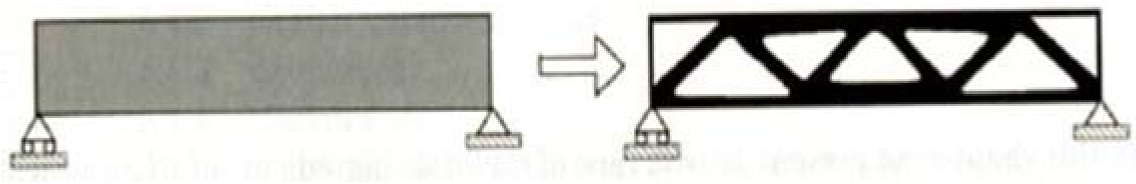
- Three components of an optimization problem
- What is topology optimization?
- Which problem is the code used to solve?
- Optimal condition
- MATLAB implementation
- Numerical examples and Numerical instability
- Mathematical programming methods
- An efficient 88-line code

Three components of an optimization problem

Objective function
Design variable
constraints

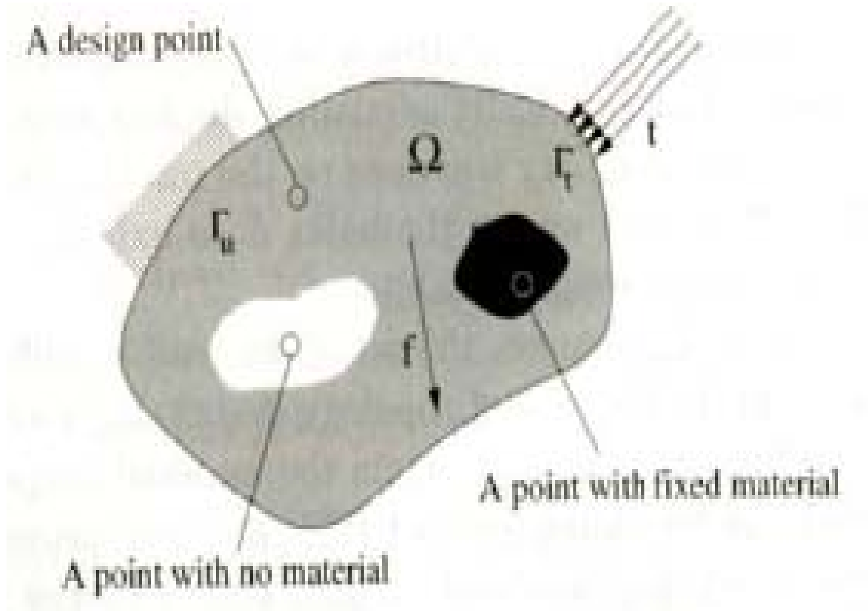


What is topology optimization?



Purpose:

To find the optimal lay-out of a structure within a specified region.



Known Quatity:

Applied loads
Possible support conditions
Volume constraint of the structure
Additional design restrictions
Location and size of prescribed holes or solid areas.

Advantage:

The physical size and the shape and connectivity of the structure do not necessary provided in advance.

Which problem is the code used to solve? -- Minimum Compliance Design

energy bilinear form: $a(u,v) = \int_\Omega E_{ijkl}(x)\varepsilon_{ij}(u)\varepsilon_{kl}(v)d\Omega$

Linearized strain: $\varepsilon_{ij}(u) = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_j})$

Load linear form: $l(u) = \int_\Omega fud\Omega + \int_{\Gamma T} tuds$

Minimum compliance design:

$$\min_{u,E_e} l(u)$$
$$s.t.: \quad a_E(u,v) = l(v), \qquad v \in U, E \in E_{ad}$$

State variable and design variable: u, E

Discretized form of compliance design:

$$\min_{u,E_e} \frac{1}{2} u^T K u$$
$$s.t.: \quad K(E_e)u = f, \qquad E_e \in E_{ad}$$

Minimum Compliant Design – Design variable

An optimal distribution of a given isotropiematerial in space

$$E_{ijkl} = l_{\Omega^{mat}} E^0_{ijkl}, \qquad l_{\Omega^{mat}} = \begin{cases} 1 \quad if \quad x \in \Omega^{mat} & \textit{Material point} \\ 0 \quad if \quad x \in \Omega^{void} & \textit{Void (no material)} \end{cases}$$

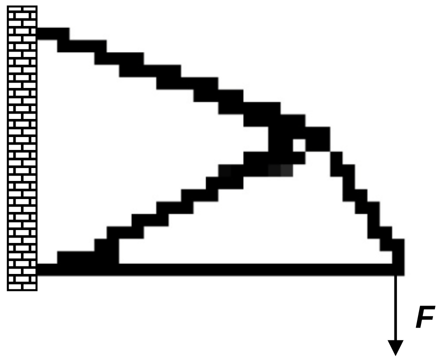
$$\int_\Omega l_{\Omega^{mat}} d\Omega = Vol(\Omega^{mat}) \le V_0$$

This is integral programming problem.

Nonsmoothness of the objective functions and design variable cannot use the gradient information.

Many times objective function eualuation is computatuionally prohibition.

Only works for a small scale problem with simulated annealing / genetic algorithims.



Black–White raster representation of the geometry with *Pixels* when using finite element discretization

Minimum Compliant Design – Design variable

Replace integral variable with continuous variables

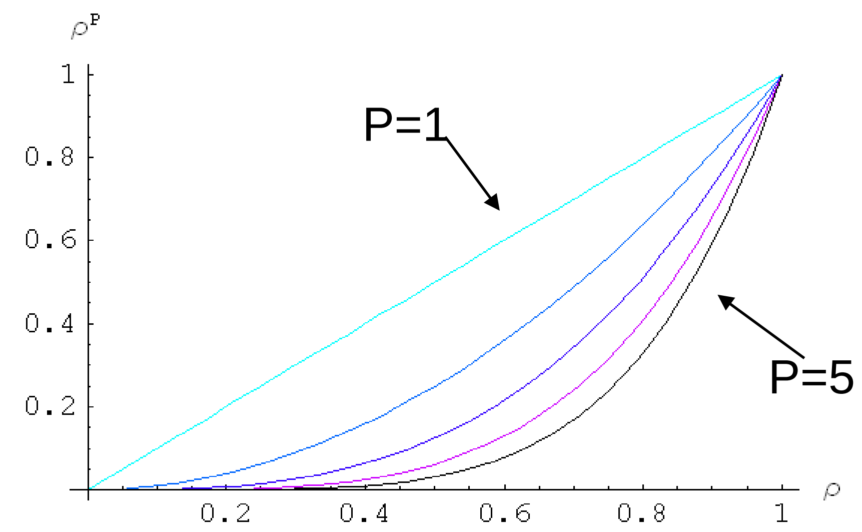
$$E_{ijkl}(x) = \rho(x)^p E^0_{ijkl}, \qquad p > 1, \quad \rho \longrightarrow \text{ Design variable}$$

$$\int_\Omega \rho(x)d\Omega \leq V; \qquad 0 \leq \rho(x) \leq 1, \qquad x \in \Omega$$

Penalty method to steers the final solution to discrete integral variable

$$E_{ijkl}(x) = E^0_{ijkl} \leftarrow \quad \rho = 1$$

$$E_{ijkl}(x) = E^{\min}_{ijkl} \leftarrow \quad \rho < 1$$



Implementation of Penalty: SIMP Method

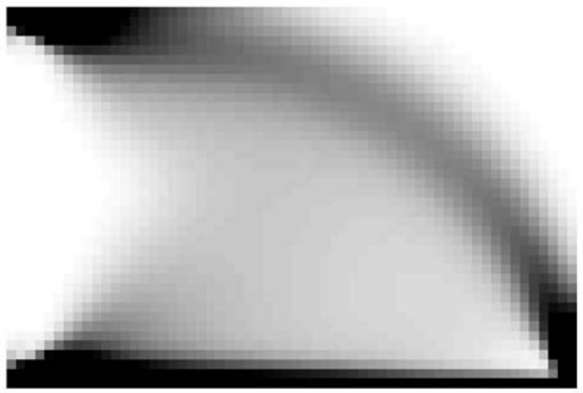
(*Solid Isotropic Material with Penalization*)

Intermediate density value are unfavourable

Penalization is achievedwith out using any explicit penalization term.

$$p \geq \max\left\{\frac{2}{1-v^0}, \frac{4}{1+v^0}\right\} \qquad (in \quad 2-D),$$

$$p \geq \max\left\{15\frac{1-v^0}{7-5v^0}, \frac{3}{2}\frac{1-v^0}{1-2v^0}\right\} \qquad (in \quad 3-D),$$



Optimal condition

For structural optimization with continuous design variable

$$\begin{aligned}
&\underset{u,\rho}{Min} \quad l(u) \\
&s.t. \quad a_E(u,v) = l(v) \quad for \quad all \quad v \in U, \\
&\qquad E_{ijkl}(x) = \rho(x)^p E^0_{ijkl}, \\
&\qquad \int_\Omega \rho(x) d\Omega \le V_0; \quad 0 < \rho_{\min} \le \rho \le 1
\end{aligned}$$

Necessary condition for SIMP Method.

$\rho_{\min} = 10^{-3} \sim 10^{-6}$ Low bound of density in order to prevent possible singularity of the equilibrium equation.

With Lagrange multiplier Λ for area constraint and λ for continuous design variable constraint

$$\begin{aligned}
L = {} & l(u) - \{a_E(u,\bar{u}) - l(\bar{u})\} + \Lambda(\int_\Omega \rho(x) d\Omega - V_0) \\
& + \int_\Omega \lambda^+(x)(\rho(x)-1) d\Omega + \int_\Omega \lambda^-(x)(\rho_{\min} - \rho(x)) d\Omega
\end{aligned}$$

Optimal condition(2)

The optimal condition for continuous design variable *ρ* is

$$\frac{\partial E_{ijkl}}{\partial \rho}\varepsilon_{ij}(u)\varepsilon_{kl}(u)=\Lambda+\lambda^{+}-\lambda^{-}$$

With the switch condition of the Lagrange multiplier

$$\lambda^{-}\geq 0,\qquad \lambda^{+}\geq 0,\qquad \lambda^{-}(\rho_{\min}-\rho(x))=0,\qquad \lambda^{+}(\rho(x)-1)=0$$

For all intermediate densities $\rho_{\min}<\rho<1$

$$p\rho(x)^{p-1}E^{0}_{ijkl}\varepsilon_{ij}(u)\varepsilon_{kl}(u)=\Lambda$$

Update scheme for the continuous design variable – Optimality criteria method

$$\rho_{K+1}=\begin{cases}\max\{(1-\zeta)\rho_K,\rho_{\min}\} & if \quad \rho_K B_K^{\eta}\leq \max\{(1-\zeta)\rho_K,\rho_{\min}\},\\ \min\{(1+\zeta)\rho_K,1\} & if \quad \min\{(1+\zeta)\rho_K,1\}\leq \rho_K B_K^{\eta},\\ \rho_K B_K^{\eta} & otherwise.\end{cases}$$

$$B_K=\Lambda_K^{-1}p\rho(x)^{p-1}E^{0}_{ijkl}\varepsilon_{ij}(u_K)\varepsilon_{kl}(u_K)$$

$\zeta,\eta\rightarrow$ *Control parameter, A typical value is 0.2 and 0.5*

Local optimal → B_K=1

MATLAB implementation

Optimization model

$$\left.\begin{aligned} \min_{\mathbf{x}}: &\quad c(\mathbf{x}) = \mathbf{U}^T\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} (x_e)^p\ \mathbf{u}_e^T\ \mathbf{k}_0\ \mathbf{u}_e \\ \text{subject to}: &\quad \frac{V(\mathbf{x})}{V_0} \leq \mathbf{1} \\ : &\quad \mathbf{K}\mathbf{U} = \mathbf{F} \\ : &\quad \mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1} \end{aligned}\right\}$$

Sensitivity

$$\frac{\partial c}{\partial x_e} = -p(x_e)^{p-1}\ \mathbf{u}_e^T\ \mathbf{k}_0\ \mathbf{u}_e$$

Update rule

- OPTIMALITY CRITERIA METHOD

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

$$B_e = \frac{-\dfrac{\partial c}{\partial x_e}}{\lambda \dfrac{\partial V}{\partial x_e}}$$

Matlab Code – Main Code

```
x(1:nely,1:nelx) = volfrac;                          % INITIALIZE
loop = 0;  change = 1.;
while change > 0.01                                   % START ITERATION
  loop = loop + 1;
  xold = x;
  [U]=FE(nelx,nely,x,penal);                          % FE-ANALYSIS
  [KE] = lk;
  c = 0.;

  for ely = 1:nely
    for elx = 1:nelx
      n1 = (nely+1)*(elx-1)+ely;
      n2 = (nely+1)* elx   +ely;
      Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
      c = c + x(ely,elx)^penal*Ue'*KE*Ue;             % OBJECTIVE FUNCTION
      dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;  % SENSITIVITY ANALYSIS
    end
  end

  [dc]     = check(nelx,nely,rmin,x,dc);              % FILTERING OF SENSITIVITIES
  [x]      = OC(nelx,nely,x,volfrac,dc);              % OPTIMALITY CRITERIA METHOD
  change = max(max(abs(x-xold)));
end
```

Matlab Code – Element Stiffness Matrix

Element Stiffness Matrix

```
function [KE]=lk
E = 1.;
nu =1/3.;
k=[ 1/2-nu/6      1/8+nu/8   -1/4-nu/12 -1/8+3*nu/8 ...
    -1/4+nu/12  -1/8-nu/8      nu/6          1/8-3*nu/8];
KE = E/(1-nu^2)* ...
             [ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
               k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
               k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
               k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
               k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
               k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
               k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
               k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

Matlab Code – FEM ANALYSIS

```
 function [U]=FE(nelx,nely,x,penal)
[KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx   +ely;
    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
    K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
  end
end

F(2*(nelx+1)*(nely+1),1)=-1;
fixeddofs=union([1,2],[2*nely+1:2*(nely+1)]);

alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
```

Matlab Code – OPTIMALITY CRITERIA

```
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
  lmid = 0.5*(l2+l1);
  xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
  if sum(sum(xnew)) - volfrac*nelx*nely > 0;
    l1 = lmid;
  else
    l2 = lmid;
  end
end
```

lmid: 50.0000	l1: 0.0000	l2: 50.0000
lmid: 25.0000	l1: 0.0000	l2: 25.0000
lmid: 12.5000	l1: 0.0000	l2: 12.5000
lmid: 6.2500	l1: 6.2500	l2: 12.5000
lmid: 9.3750	l1: 6.2500	l2: 9.3750
lmid: 7.8125	l1: 6.2500	l2: 7.8125
lmid: 7.0313	l1: 7.0313	l2: 7.8125
lmid: 7.4219	l1: 7.0313	l2: 7.4219
lmid: 7.2266	l1: 7.2266	l2: 7.4219
lmid: 7.3242	l1: 7.3242	l2: 7.4219
lmid: 7.3730	l1: 7.3242	l2: 7.3730
lmid: 7.3486	l1: 7.3242	l2: 7.3486
lmid: 7.3364	l1: 7.3364	l2: 7.3486
lmid: 7.3425	l1: 7.3425	l2: 7.3486
lmid: 7.3456	l1: 7.3425	l2: 7.3456
lmid: 7.3441	l1: 7.3441	l2: 7.3456
lmid: 7.3448	l1: 7.3448	l2: 7.3456
lmid: 7.3452	l1: 7.3448	l2: 7.3452
lmid: 7.3450	l1: 7.3450	l2: 7.3452
lmid: 7.3451	l1: 7.3450	l2: 7.3451

Matlab code command

top(nelx, nely, volfrac, penal, rmin)

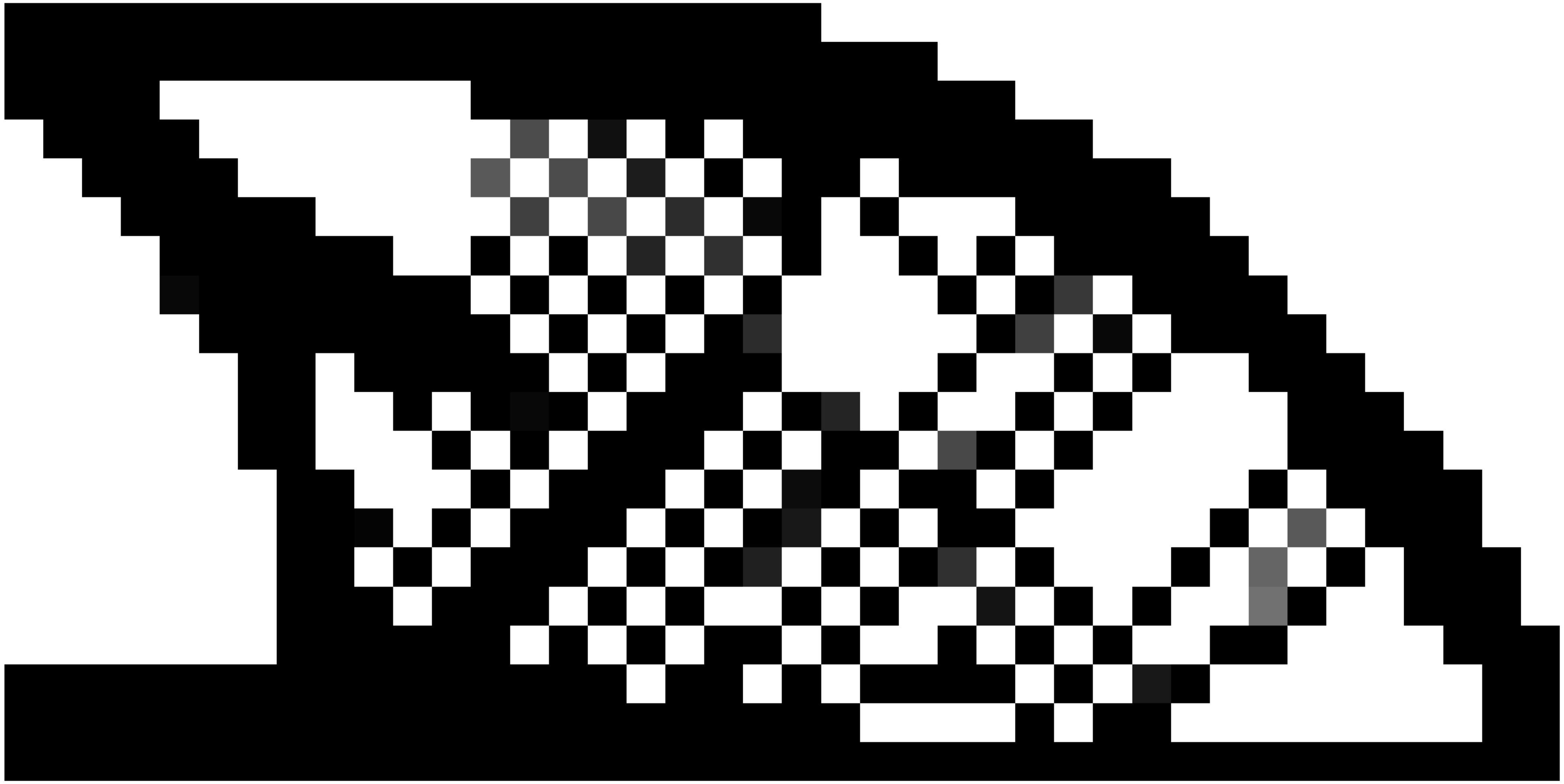
- nelx and nely: number of elements in the horizontal and vertical directions,
- volfrac: volume fraction,
- penal: penalization power,
- rmin: filter size(divided by element size).

Example 1

top(40, 20, 0.5, 3, 1.0)

Example 1

top(40, 20, 0.5, 3, 1.0)



Checkerboard Pattern

Example 1 -- Checkerboard Pattern Problem

Solution: Low-pass filter

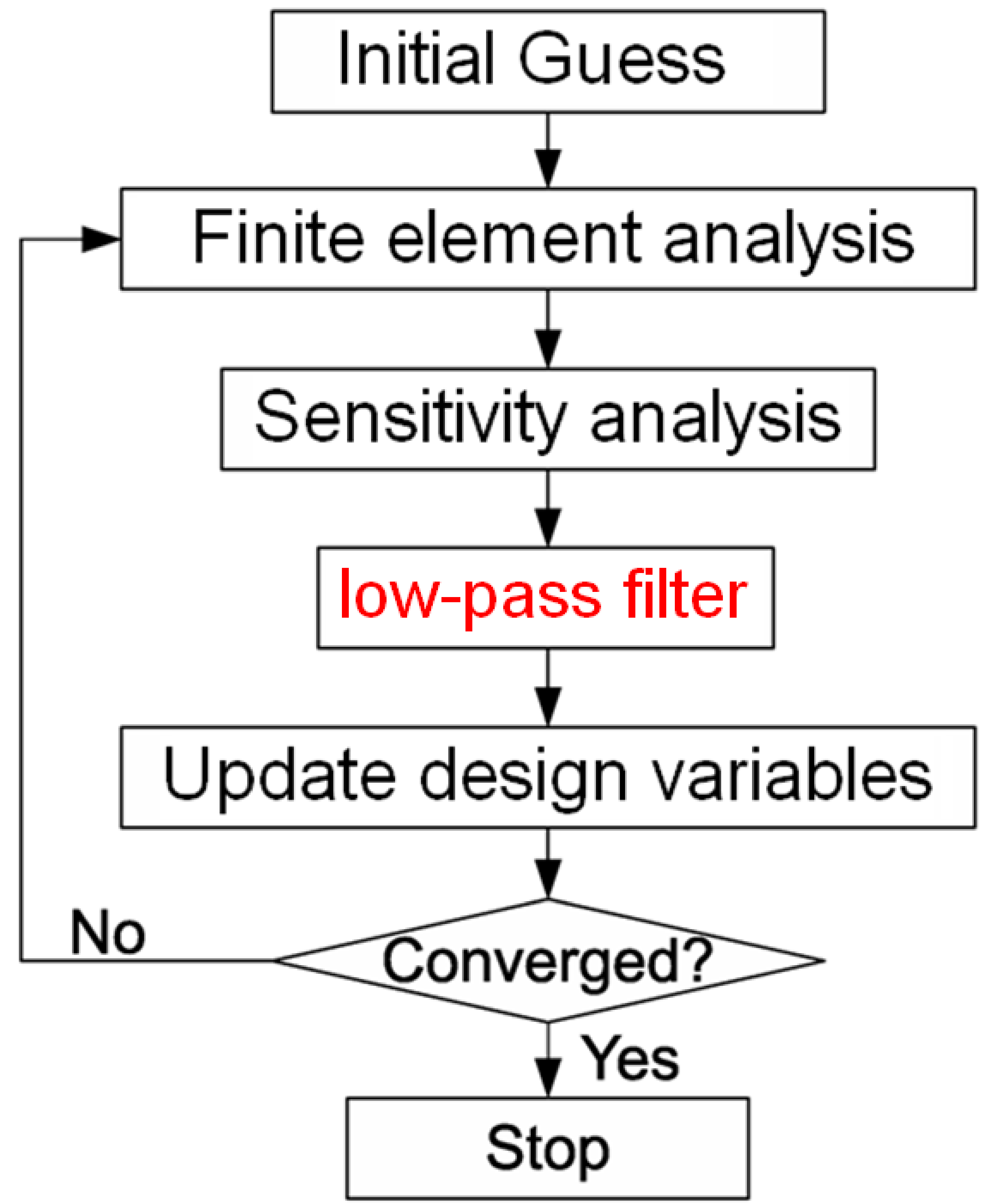
$$\widehat{\frac{\partial c}{\partial x_e}} = \frac{1}{x_e \sum_{f=1}^{N} \hat{H}_f} \sum_{f=1}^{N} \hat{H}_f \, x_f \, \frac{\partial c}{\partial x_f} .$$

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

$$\{f \in N \mid \mathrm{dist}(e, f) \le r_{\min}\}, \quad e = 1, \ldots , N$$

[dc] = check(nelx, nely, rmin, x, dc);

% FILTERING OF SENSITIVITIES

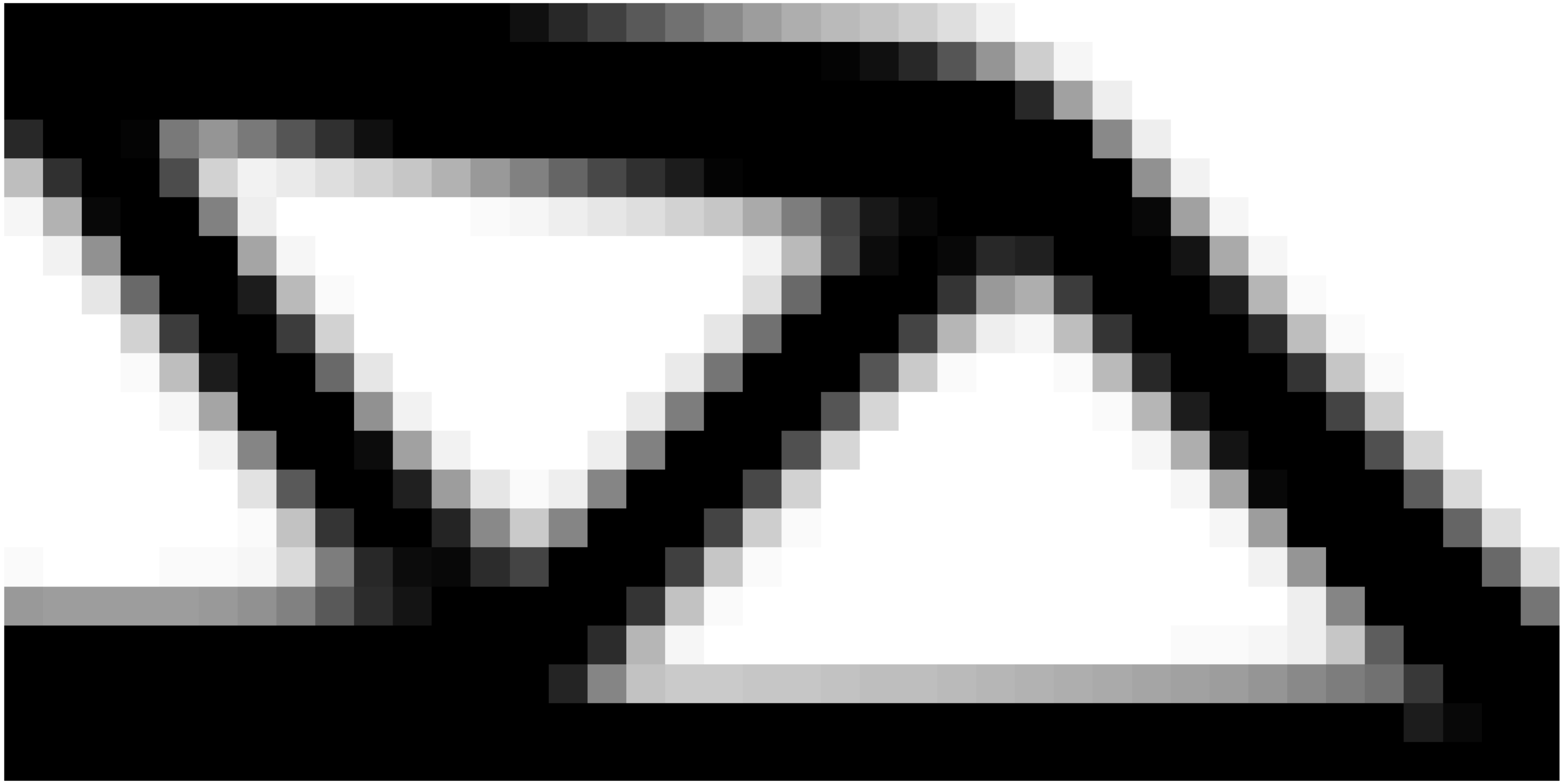


Example 2

top(40, 20, 0.5, 3, 1.5)

Example 2

top(40, 20, 0.5, 3, 1.5)

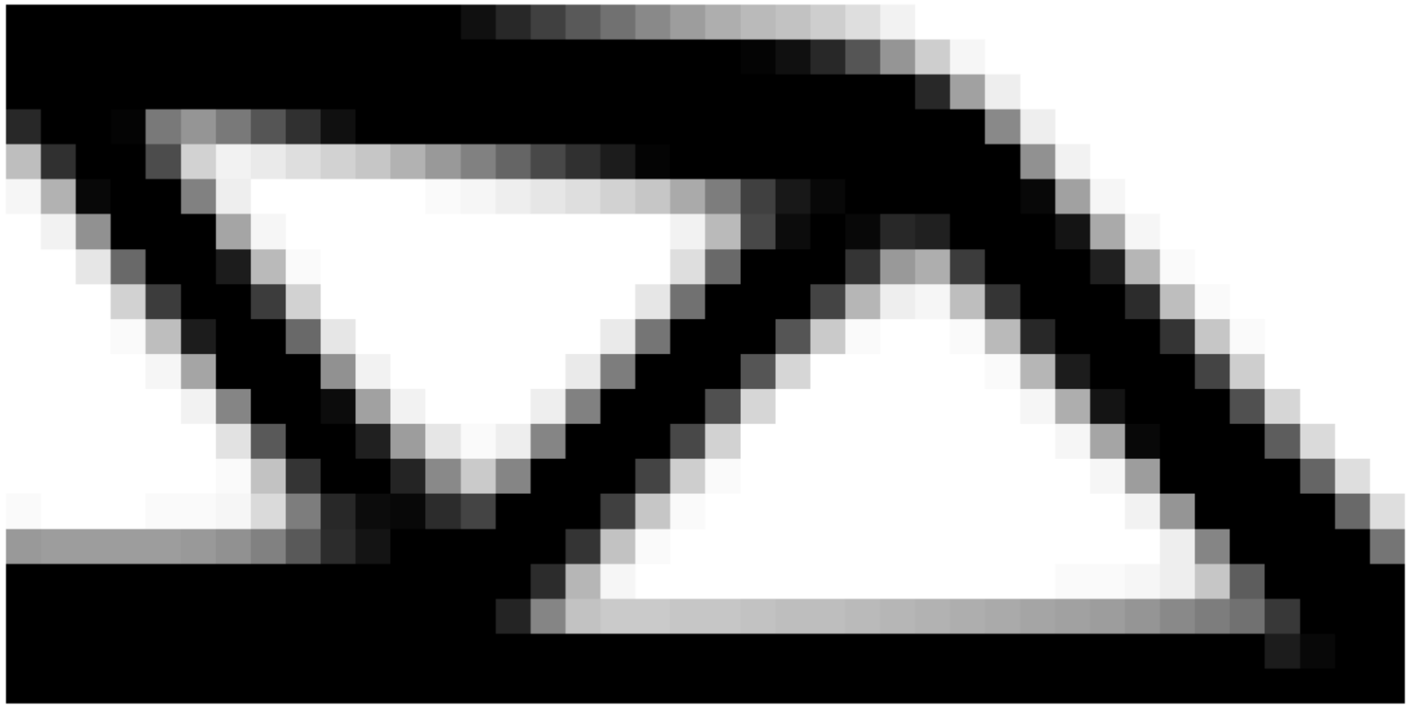


top(40, 20, 0.5, 3, 3) what will happen?

Example 2

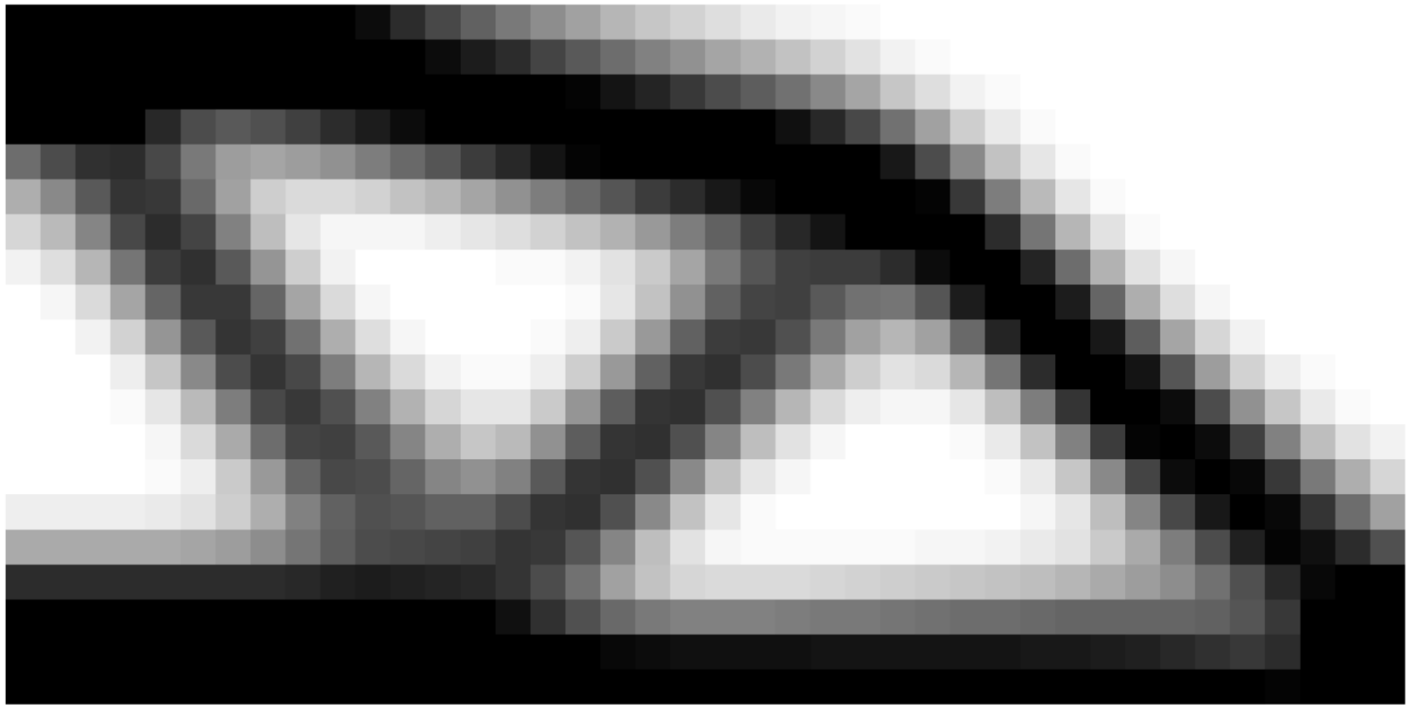
rmin=1.5

Obj=82.7562;



rmin=3

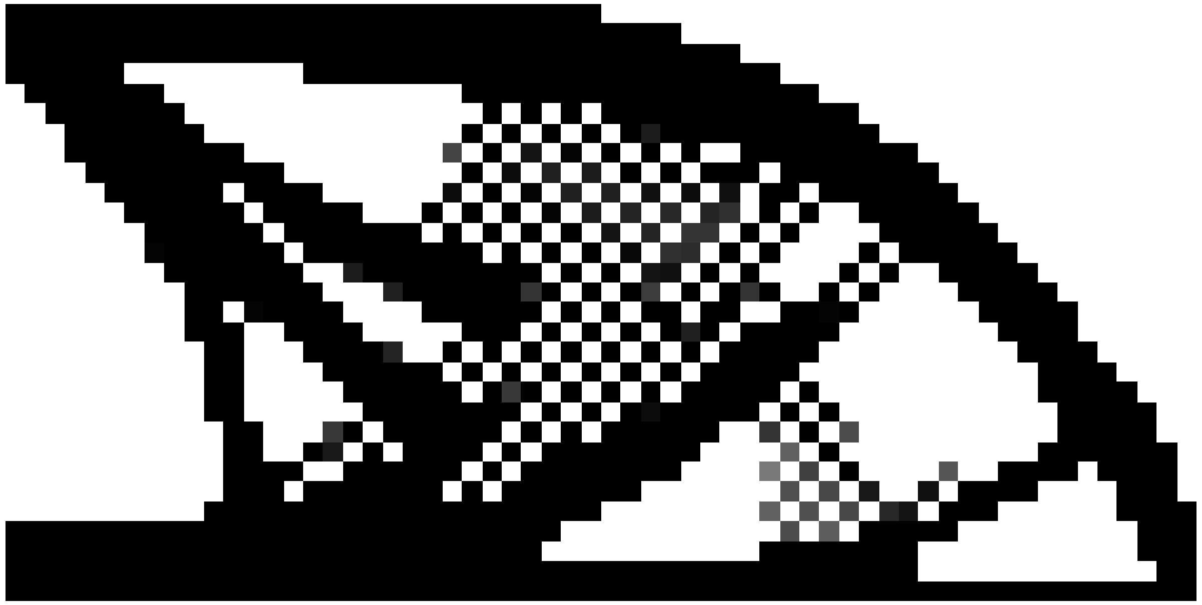
Obj=99.1929;



Example 3

top(60, 30, 0.5, 3, 1.0)

Obj: 83.0834



top(40, 20, 0.5, 3, 1.0)

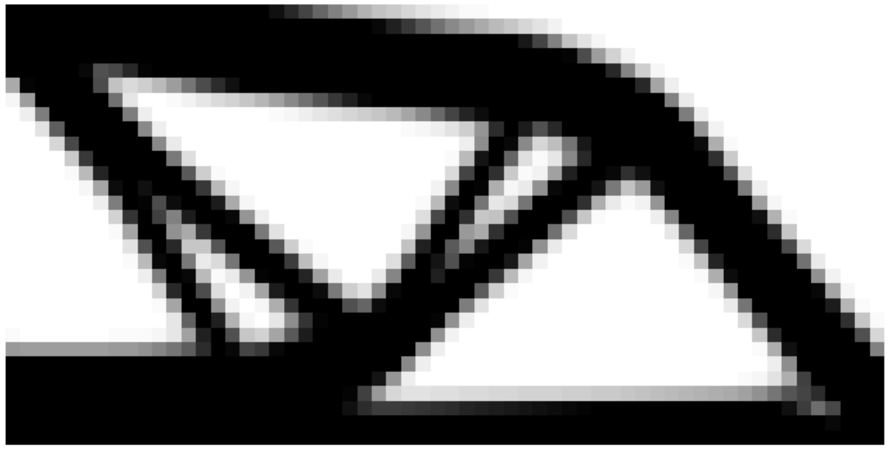
Obj: 80.4086;

Mesh dependency

Example 3

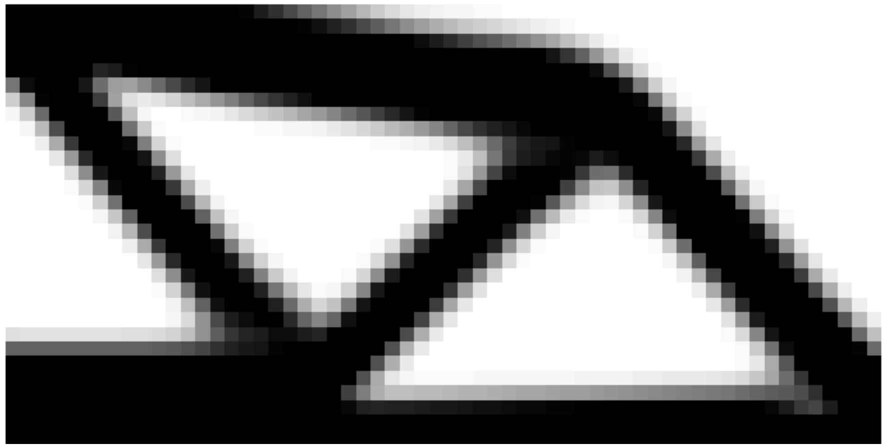
top(60, 30, 0.5, 3, 1.5)

Obj: 81.3491



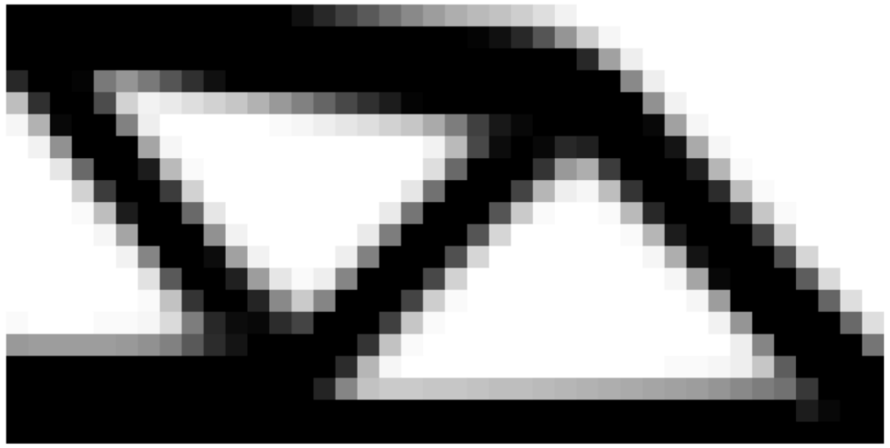
top(60, 30, 0.5, 3, 2.25)

Obj: 83.5963



top(40, 20, 0.5, 3, 1.5)

Obj=82.7562;



Numerical instability

Numerical Instabilities in Structural Topology Optimization

- Checkerboard Pattern
- Mesh – dependency
- Local Minimal solution (non-convex optimization)

Remedy

- Low-pass filter.
- Add other constraints to the optimization problem [1].

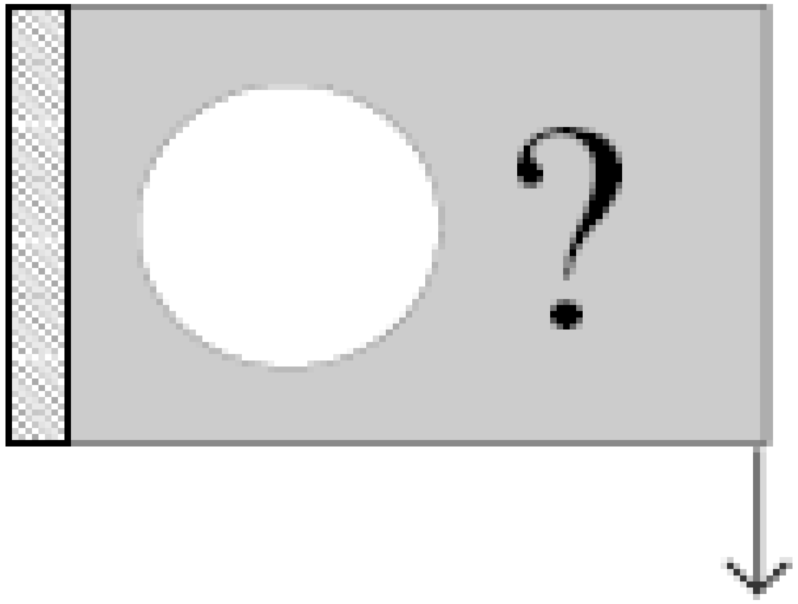
Example 4 – passive elements

- Add passive to the call in lines 29 and 39.
- Add line after line 43

```
xnew(find(passive))=0.001;
```

- Add the follwing 10 lines to the main program (after line 5)

```
for ely= 1:nely
 for elx=1:nelx
  if sqrt((ely-nely/2.)^2+(elx-nelx/3.)^2) < nely/3.
     passive(ely,elx)=1;
     x(ely,elx)=0.001;
   else
     passive(ely,elx)=0;
   end
 end
end
```



top (45,30,0.5,3,1.5)

Update design variables – mathematical programming methods

At each iteration step, updated design variables can be obtained by solving a simpler approximate optimization subproblem.

These subproblems are constructed based on sensitivity information at the current iteration step as well as some iteration history. [1]

Our optimization problem can be transformed into the following general form

$$\min_{\mathrm{x}} f_0(\mathrm{x})$$

$$\text{s.t.} \quad f_i(\mathrm{x}) \leq 0, \qquad i = 1, \ldots, m$$

$$\mathrm{x} \in \chi = \{\mathrm{x} \in \Re^n, x_j^{\min} \leq x_j \leq x_j^{\max}, j = 1, \ldots, n\}.$$

Sequential linear programming (SLP)

The subproblem at iteration *k* is as follows [2]

$$
\begin{aligned}
&\min_{\mathrm{x}} \; f_0(\mathrm{x}^k) + \nabla f_0(\mathrm{x}^k)^T(\mathrm{x}-\mathrm{x}^k) \\
&\text{s.t.} \;\; f_i(\mathrm{x}^k) + \nabla f_i(\mathrm{x}^k)^T(\mathrm{x}-\mathrm{x}^k) \le 0, \qquad i = 1,\ldots,m \\
&\qquad \mathrm{x} \in \chi, \\
&\qquad l_j^k \le x_j - x_j^k \le u_j^k,\; j = 1,\ldots,n.
\end{aligned}
$$

Here, l_j^k and u_j^k are move limits.

This subproblem may be solved by the simplex algorithm.

Sequential quadratic programming (SQP)

The subproblem at iteration *k* is as follows [2]

$$\min_{\mathrm{x}} \ f_0(\mathrm{x}^k) + \nabla f_0(\mathrm{x}^k)^T(\mathrm{x}-\mathrm{x}^k) + \frac{1}{2}(\mathrm{x}-\mathrm{x}^k)^T H(\mathrm{x}^k)(\mathrm{x}-\mathrm{x}^k)$$

$$\text{s.t.} \ \ f_i(\mathrm{x}^k) + \nabla f_i(\mathrm{x}^k)^T(\mathrm{x}-\mathrm{x}^k) \le 0, \qquad i = 1,\ldots,m$$

$$\mathrm{x} \in \chi,$$

Here, $H(\mathrm{x}^k)$ is a positive define first order approximation of the Hessian of f_0 at x^k.

SLP and SQP are designed to solve general nonlinear optimization problems.

Convex linearization (CONLIN)

The subproblem at iteration *k* is as follows [2]

$$\min_{\mathrm{x}} f_0^{C,k}(\mathrm{x})$$

$$\text{s.t.} \quad f_i^{C,k}(\mathrm{x}) \le 0, \qquad i = 1, \ldots, m$$

$$\mathrm{x} \in \chi := \{\mathrm{x} \in \Re^n, 0 < x_j^{\min} \le x_j \le x_j^{\max}, j = 1, \ldots, n\}.$$

Where

$$f_i^{C,k}(\mathrm{x}) := f_i(\mathrm{x}^k) + \sum_{j \in \Omega_+} f_{ij}^{L,k}(\mathrm{x}) + \sum_{j \in \Omega_-} f_{ij}^{R,k}(\mathrm{x}),$$

$$\text{Here}, \Omega_+ := \{ j : \partial f_i(\mathrm{x}^k) / \partial x_j > 0\} \text{ and } \Omega_- := \{ j : \partial f_i(\mathrm{x}^k) / \partial x_j \le 0\},$$

$$f_{ij}^{L,k}(\mathrm{x}) = \partial f_i(\mathrm{x}^k) / \partial x_j (x_j - x_j^k) \text{ and}$$

$$f_{ij}^{R,k}(\mathrm{x}) = \frac{\partial f_i(\mathrm{x}^k)}{\partial x_j} \frac{x_j^k (x_j - x_j^k)}{x_j}.$$

Method of moving asymptotes (MMA)

The subproblem at iteration *k* is as follows [2]

$$\min_{\mathrm{x}} f_0^{M,k}(\mathrm{x})$$

$$\text{s.t. } f_i^{M,k}(\mathrm{x}) \le 0, \qquad i = 1, \ldots, m$$

$$\alpha_j^k \le x_j \le \beta_j^k,\ j = 1, \ldots, n. \text{ Here, } \alpha_j^k \text{ and } \beta_j^k \text{ are move limits.}$$

Where

$$f_i^{M,k}(\mathrm{x}) := f_i(\mathrm{x}^k) - \sum_{j=1}^{n} \left(\frac{p_{ij}^k}{U_j^k - x_j^k} + \frac{q_{ij}^k}{x_j^k - L_j^k} \right) + \sum_{j=1}^{n} \left(\frac{p_{ij}^k}{U_j^k - x_j} + \frac{q_{ij}^k}{x_j - L_j^k} \right),$$

Here, L_j^k and U_j^k are moving asymptotes that satisfy $L_j^k < x_j^k < U_j^k$,

$$p_{ij}^k = \begin{cases} (U_j^k - x_j^k)^2 \partial f_i(\mathrm{x}^k)/\partial x_j & \text{if } \partial g_i(\mathrm{x}^k)/\partial x_j > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{ij}^k = \begin{cases} 0 & \text{if } \partial g_i(\mathrm{x}^k)/\partial x_j \ge 0 \\ -(x_j^k - L_j^k)^2 \partial f_i(\mathrm{x}^k)/\partial x_j & \text{otherwise.} \end{cases}$$

Developed by Svanberg [3]

MMA codes

Svanberg's MMA codes are used to solve the following

$$\left.\begin{array}{rl}\displaystyle\min_{\boldsymbol{x},\boldsymbol{y},z} : & f_0(\boldsymbol{x}) + a_0\ z + \displaystyle\sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) \\ \text{subject to} : & f_i(\boldsymbol{x}) - a_i\ z - y_i \leq 0, \qquad i = 1, \ldots, m \\ : & x_j^{min} \leq x_j \leq x_j^{max}, \qquad j = 1, \ldots, n \\ : & y_i \geq 0, \qquad i = 1, \ldots, m \\ : & z \geq 0 \end{array}\right\},$$

Our problem is

$$\left.\begin{array}{rl}\displaystyle\min_{\boldsymbol{x}} : & f_0(\boldsymbol{x}) \\ \text{subject to} : & f_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \ldots, m \\ : & x_j^{min} \leq x_j \leq x_j^{max}, \qquad j = 1, \ldots, n \end{array}\right\},$$

Svanberg's suggestion for our problem

$$a_0 = 1,\ a_i = 0,\ c_i = 1000,\ d = 0.$$

MMA codes

The MMA call is

```
function [xmma,ymma,zmma,lam,xsi,eta,mu,zet,s,low,upp] = ...
    mmasub(m,n,iter,xval,xmin,xmax,xold1,xold2, ...
    f0val,df0dx,df0dx2,fval,dfdx,dfdx2,low,upp,a0,a,c,d);
```

Hints:

- Check the definition of the MMA variables in the mmasub.m file.
- Be careful of the difference between row and colume vectors.
- Normalize constraints and objective function. i.e. use $V(x)/V_0-1\leq 0$ instead of $V(x)\leq V_0$.

Efficient topology optimization in MATLAB using 88 lines of code [4,5]

- A valuable successor to the 99 line code.
- A speed improvement.
- Other low-pass filter methods.

Summary

Three components of an optimization problem.

Numerical instability is discussed.

Update design variables by mathematical programming methods.

refenrence

[1] M. P. Bendsøe and O. Sigmund, Topology Optimization: Theory, Methods and Applications (Second Edition), 2003 Springer Berlin Heidelberg.

[2] P. W. Christensen and A. Klarbring, An Introduction to Structural Optimization, 2009 Springer Science + Business Media B.V.

[3] K. Svanberg, The method of moving asymptotes—a new method for structural optimization. Internat. J. Numer. Methods Eng. 24:359–373 (1987).

[4] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov and O. Sigmund, Efficient topology optimization in MATLAB using 88 lines of code, Struct. Multidisc. Optim. 43:1–16 (2011).

[5] http://www.topopt.dtu.dk/

Thanks for your attention