ERATA

Topology Optimization – Theory, Methods and Applications

Main changes in the 2nd printing

Martin P. Bendsøe and Ole Sigmund

October 7, 2003

In the following we write the changes that were done in the second printing of our book (due late fall 2003). A number of references have also been updated.

p. 34, Eq. (1.26) changed to

$$
\begin{aligned}
&E_{ijkl}(x) = ((\rho * K)(x))^p \, E^0_{ijkl}, \;\; \rho \in L^\infty(\Omega) \; , \\
&(\rho * K)(x) = \frac{1}{< K >} \int_\Omega \rho(y) K(x-y) \, \mathrm{d}y \; , \quad < K > = \int_{R^n} K(y) \, \mathrm{d}y \; , \\
&\int_\Omega \rho(x) \mathrm{d}\Omega \leq V; \quad 0 \leq \rho(x) \leq 1, \quad x \in \Omega \; ,
\end{aligned}
$$

p. 53, Eq. (1.35) index on C changed:

$$
\min_{E \in \mathsf{E}_{\mathrm{ad}}} \; \min_{\substack{\mathrm{div}\sigma^k + f^k = 0 \text{ in } \Omega, \\ \sigma^k \cdot n = t^k \text{ on } \Gamma_T^k \\ k=1,\ldots,M}} \left\{ \frac{1}{2} \int_\Omega \sum_{k=1}^M w^k C_{ijpq} \sigma^k_{ij} \sigma^k_{pq} \mathrm{d}\Omega \right\} .
$$

p. 54, index on E^0 changed:

$$
B_K = \Lambda_K^{-1} p \rho(x)^{(p-1)} E^0_{ijnm} \sum_{k=1}^M w^k \varepsilon_{ij}(u_K^k) \varepsilon_{nm}(u_K^k) \; .
$$

p. 55, Eq. (1.36) ρ corrected to h:

$$\begin{aligned} &\min_{u,h} \quad l(u) \\ &\text{s.t.} : a_h(u,v) \equiv \int_\Omega h(x) E^0_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v) \mathrm{d}\Omega = l(v), \ \text{for all}\, v \in U \ , \\ &\qquad \int_\Omega h(x) \mathrm{d}\Omega \le V, \ h_{\min} \le h \le h_{\max} < \infty \ . \end{aligned}$$

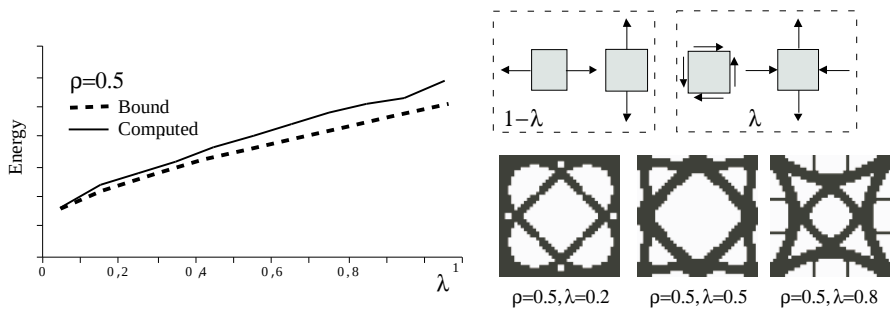
p. 55, Eq. (1.37) changed to

$$\begin{aligned} &\min_{\substack{h \in L^\infty(\Omega), \\ h_{\min} \le h \le h_{\max} < \infty \\ \int_\Omega h(x) \mathrm{d}\Omega \le V}} \quad c(h) \\ &c(h) = \max_{v \in U} \left\{ 2l(v) - \int_\Omega h(x) E^0_{ijkl} \varepsilon_{ij}(v) \varepsilon_{kl}(v) \mathrm{d}\Omega \right\} . \end{aligned}$$

p. 80, last line in figure text:
"lower" changed to "higher"

p. 81, 3rd line in figure text:
added "hatched area"

p. 188, Fig. 3.14 corrected to:



p. 234, l. 14:
"$\mathbf{u}^T \widetilde{\mathbf{A}} \mathbf{u} \ge$" corrected to "$\mathbf{u}^T \widetilde{\mathbf{A}} \mathbf{u} \ge 0$"

p. 234, 3rd line of last paragraph:
"Modern interior ...." changed to "Modern interior point algorithms solve SDP's like (4.20) in polynomial time [33]."

p. 236, Eq. (4.25) τ^2 changed to τ:

$$\min_{\mathbf{u},\,\tau} \left\{ \tau - \mathbf{f}^T \mathbf{u} \right\}$$
$$\text{s.t.} : \frac{V}{2} \left[\mathbf{u}^T \hat{\mathbf{K}}_i \mathbf{u} - 2 \mathbf{g}_i{}^T \mathbf{u} \right] \leq \tau, \quad i = 1, \ldots, m \ .$$

p. 239, Eq. (4.31) changed to:

$$\min_{\substack{q_i^+ \geq 0,\, q_i^- \geq 0, \\ i=1,\ldots,m \\ \mathbf{B}(\mathbf{q}^+ - \mathbf{q}^-) = \mathbf{f}}} \sum_{i=1}^{m} \frac{l_i}{\bar{\sigma}_i} (q_i^+ + q_i^-) \ .$$

p. 239, l. 4 of last paragraph:
"... primal-dual LP-methods" changed to "primal-dual interior point LP-methods or the simplex algorithm"

p. 240, Eq. (4.32) changed to:

$$\min_{\mathbf{u},\,\tau} \left\{ \frac{1}{2} \mathbf{u}^T \left(\sum_{i \in S} s_i \mathbf{K}_i \right) \mathbf{u} - \mathbf{f}^T \mathbf{u} + \tau^2 \right\}$$
$$\text{s.t.} : \ -\tau \leq \sqrt{\frac{VE_i}{2}} \frac{\mathbf{b}_i{}^T \mathbf{u}}{l_i} \leq \tau, \quad i \in R \ .$$

p. 250, Eq. (4.44) V_τ changed to $V\tau$:

$$\inf_{\substack{\lambda^k > 0,\, \mathbf{u}^k,\, \tau \\ \sum_{k=1}^M \lambda^k = 1}} \left\{ V\tau - \sum_{k=1}^{M} \mathbf{f}^{k\,T} \mathbf{u}^k \right\}$$
$$\text{s.t.} : \sum_{k=1}^{M} \frac{1}{2\lambda^k} \mathbf{u}^{k\,T} \mathbf{K}_i \mathbf{u}^k - \tau \leq 0, \quad i = 1, \ldots, m \ .$$

p. 250, Eq. (4.45) V_τ changed to $V\tau$ and $\mathbf{A}_i$ changed to $\mathbf{K}_i$:

$$\inf_{\substack{s^k,\, \mathbf{x}^k, \tau \\ \sum_{k=1}^M (s^k)^2 = 1}} \left\{ V\tau - \sum_{k=1}^{M} s^k \ \mathbf{f}^{k\,T} \mathbf{x}^k \right\}$$
$$\text{s.t.} : \sum_{k=1}^{M} \mathbf{x}^{k\,T} \mathbf{K}_i \ \mathbf{x}^k - 2\tau \leq 0, \quad i = 1, \ldots, m \ .$$

p. 255, first equation, $\mathbf{A}_i$ changed to $\mathbf{K}_i$:

$$\mathcal{F}(\mathbf{x}) = -\min_{\mathbf{u}} \left[\max_{i=1,\ldots,m} \left\{ \frac{V}{2} \mathbf{u}^T \mathbf{K}_i(\mathbf{x}) \mathbf{u} - \mathbf{f}^T \mathbf{u} \right\} \right] \ ,$$

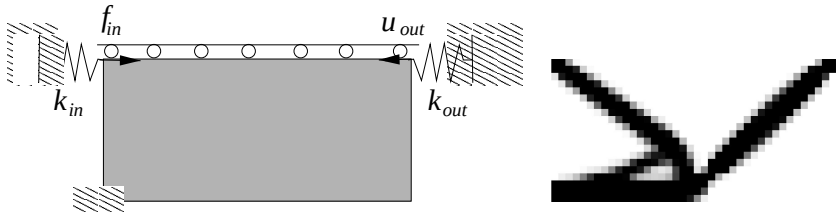


Figure 1: Topology optimization of the inverter. Left: half design domain with symmetry boundary conditions and right: resulting topology.

p. 271, 6th to last line changed to:

```
> fixeddofs   = [nely/2+1-(nely/20):nely/2+1+(nely/20)];
```

The Matlab code and description for mechanism synthesis in Appendix 5.1.5 on page 269 has been updated and improved to:

5.1.5 A 104 line MATLAB code for compliant mechanism synthesis

The Matlab code for compliance minimization described in the previous section can be changed to a code for mechanism synthesis by changing 13 lines, deleting one and adding 6 new lines of code.[1]

As the default problem, we consider the inverter design problem sketched in Fig. 5.5. The optimization problem for compliant mechanism synthesis was discussed in Sec 2.6. The solution obtained by running the modified code which is named 'topm' with the command line input

```
topm(40,20,0.3,3.0,1.2)
```

is seen in Fig. 1(right).

Instead of listing the whole program we just show a list of the changes. This list is obtained by comparing the compliance minimization program 'top.m' with the inverter design program 'topm.m' using the UNIX command 'diff top.m topm.m'. This results in output where '<' means lines in 'top.m' and '>' means lines in 'topm.m'. In the following we briefly discuss the changes.

First we rename the code from 'top' to 'topm'

```
1,2c1,2
< %%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999 %%%
```

[1]Note that the code described uses linear analysis. Therefore, it can only be used to gain insight into compliant mechanism synthesis by topology optimization. For practical problems one should modify the code to include geometrically non-linear modelling (c.f. Sec. 2.6.5).

```
< function top(nelx,nely,volfrac,penal,rmin);
---
> %%%% A 104 LINE COMPLIANT MECHANISM DESIGN CODE BY OLE SIGMUND, MAY 2002 %%%
> function topm(nelx,nely,volfrac,penal,rmin);
```

Instead of calculating the output displacement (objective funtion) in the main program we return it from the FE subroutine and we remove its initialization

```
12c12
<   [U]=FE(nelx,nely,x,penal);
---
>   [U,c]=FE(nelx,nely,x,penal);
15d14
<   c = 0.;
```

The expression for the sensitivities (2.23) depends on the solution to the adjoint load case (second column of the displacement matrix `U`)

```
20,22c19,21
<       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;
<       dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
---
>       Ue1 = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
>       Ue2 = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],2);
>       dc(ely,elx) = penal*x(ely,elx)^(penal-1)*Ue1'*KE*Ue2;
```

We improve the convergence criteria for the bi-sectioning algorithm

```
39,40c38,39
< l1 = 0; l2 = 100000; move = 0.2;
< while (l2-l1 > 1e-4)
---
> l1 = 0; l2 = 100000; move = 0.1;
> while (l2-l1)/(l2+l1) > 1e-4 & l2 > 1e-40
```

To stabilize convergence we use a damping factor of 0.3 instead of 0.5 and we take care of the possibility of positive sensitivities

```
42c41
<   xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
---
>   xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*(max(1e-10,-dc./lmid)).^0.3))));
```

We return the output displacement to the main program

```
66c65
< function [U]=FE(nelx,nely,x,penal)
---
> function [U,Uout]=FE(nelx,nely,x,penal)
```

We allocate force and displacement vectors for the real and the adjoint load cases

```
69c68
< F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
---
> F = sparse(2*(nely+1)*(nelx+1),2); U = sparse(2*(nely+1)*(nelx+1),2);
```

Finally, we define the boundary conditions and the input and output points. Furthermore, we add external springs with stiffness 0.1 to the input and output points and we save the value of the output displacement to be returned to the main program.

```
78,80c77,84
< % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
< F(2,1) = -1;
< fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
---
> % DEFINE LOADS AND SUPPORTS (HALF FORCE INVERTER)
> din=1;
> dout=2*nelx*(nely+1)+1;
> F(din,1) = 1;
> F(dout,2) =  -1;
> K(din,din) = K(din,din) + 0.1;
> K(dout,dout) = K(dout,dout) + 0.1;
> fixeddofs   = union([2:2*(nely+1):2*(nely+1)*(nelx+1)], [2*(nely+1):-1:2*(nely+1)-3]);
85a90
> Uout = U(dout,1);
100,119d104
```