10th World Congress on Structural and Multidisciplinary Optimization
May 19 -24, 2013, Orlando, Florida, USA

Element energy based method for topology optimization

Vladimir Uskov[1]

[1] Central Aerohydrodynamic Institute (TsAGI), Zhukovsky, Russia

1. Abstract

A simple topology optimization method for minimization of structural compliance at given volume is proposed. It is supposed that the external loads and self-weight forces are applied. It is based on the SIMP method by limit transfer in new design variables. The search direction and Lagrange multiplier are calculated by using two different variants for formulation of the volume constraint. The filtering procedure in new design variables permits us to avoid "grey" solutions and to obtain clear structural boundaries. The proposed method is demonstrated with several examples of cantilever and MBB beam. The Wolfram Mathematica code is given.

2. Keywords: Topology optimization, SIMP, Filters, Self-weight, Mathematica

Introduction

The SIMP method [1] is widely used in topology optimization of structural compliance. This method uses the transfer from integer design variables 0-1 to real ones which have also the values in the range (0, 1). It gives possibility to use the gradient approaches to optimization. The "grey" solutions are in result of the application of these methods. In many cases these results are suitable for engineering interpretation for the following design stages. However in some cases the "grey" solution can essentially differ from optimal "black-white" solution as, for example, in the considered self-weight problem in this paper. Also it is desirably to have possibility to extract the surface of the obtained structure. In the SIMP method a fraction of the "grey" is reduced by means of the increase of the penalization parameter p. Usually a moderate parameter value p (p=3) is used and sometimes more larger values (for example, p=9) is used additional iterations. Using of the large values of p leads to the local minimum and to avoid this problem such SIMP method with continuation is used. Also in this case some filtering procedure is employed. In addition the filtering can control structural topology layout by decrease of the number of holes. In contrary the filtering blurs the boundaries of structure. The decrease the blur effect can be done by performing the filtering in new variables. A simple topology optimization method is proposed in the paper. It is built by take limit when $p \to \infty$ in new design variables.

The self-weight problems have the non-monotonic function of compliance and it is known that some optimization procedures fail with these problems. Also different approximation scheme of the MMA family have large difficulties with solving the problem [2]. The parasite effect at low densities is appeared at using power law of the SIMP method. This paper proposes the simple optimization procedure which can be suitable for the problems with the non-monotonic function of compliance. The correction term is proposed in the density law which can exclude the parasite effect.

3. Problem statement

The following discrete problem for searching the optimal topology of structure is solved:

$$\min C\left(x_1, x_2, \ldots, x_n\right) = \mathbf{f}^T \mathbf{u}$$

$$\text{subject to} \quad \begin{array}{c} \mathbf{K}\mathbf{u} = \mathbf{f} \\ V(x) = \sum_{i=1}^{n} x_i = V_0 \end{array}$$

where C is potential strain energy (compliance), x_i, $i =$ 1,…, n are equaled to 0 or 1 (design variables), n is the number of finite elements and V_0 is a given volume, $\mathbf{u}$ is the vector of displacements, $\mathbf{K}$ is the global stiffness matrix and $\mathbf{f}$ is the vector of forces.

Two-dimensional problem of theory of elasticity is considered. The initial structure is modeled by square finite elements with uniform thickness. The used material is isotropic. Boundary conditions are specified on some part of structure as fixation of nodal displacements.

4. The proposed method

It was noted in the paper [3] that the element sensitivity at $p \to \infty$ is proportional to potential energy of element. Therefore, the relation of the ESO method and the SIMP method is revealed. The ESO method gives the

"black-white" solution while the SIMP method leads to "grey" solution. It was an impact to develop the method which is free of some drawbacks of the ESO and SIMP methods.
The artificial power dependence with the parameter p is used is used in the SIMP method for calculation of Young modulus in dependence on design variables x_i which can take the values in the range $0 \le x_i \le 1$. In this case the function $C\left(x_1^p, x_2^p, \ldots, x_n^p\right)$ is minimized. The equivalent problem statement can be formulated with different independent variables $y_i = x_i^p$ [4]. The goal function $C\left(y_1, \mathrm{y}_2, \ldots, \mathrm{y}_n\right)$ coincide with the goal function of initial problem. However, now the constraint on the volume is nonlinear function of design variables

$$V(y) = \sum\nolimits_{i=1}^{n} y_i^{1/p} = V_0 \, .$$

Note that for intermediate values of y_i the inequality $\sum\nolimits_{i=1}^{n} y_i < V_0$ is valid. If the reduced value of the volume is used in the optimization procedure then the same solution would be obtain on the optimization step as for initial formulation. So the idea of the SIMP method can be interpreted as the solution without penalization (p=1), but with the reduced value of volume. The proposed here method uses this observation.
The derivatives of compliance with respect to design variable y can be calculated by formula

$$\frac{\partial C}{\partial y} = 2\mathbf{u}^T \frac{\partial \mathbf{f}}{\partial y} - \mathbf{u}^T \frac{\partial \mathbf{K}}{\partial y}\mathbf{u} \qquad (1)$$

The Young modulus of element and element density are linear of the design variable y

$$E = E(y) = yE_0 \, ,$$

$$\rho = \rho(y) = y\rho_0 \, ,$$

where E_0 is Young modulus of material and ρ_0 is material density. Here and below the low index of element number is omitted. For the stiffness matrix we will have

$$\mathbf{K} = y\mathbf{K}_0 \, ,$$

where $\mathbf{K}_0$ is the element stiffness matrix with the modulus E_0. The forces in the self-weight problems are equal:

$$\mathbf{f} = y\mathbf{g} \, ,$$

where the multiplier $\mathbf{g}$ is the constant depending on acceleration of gravity and material density.
Substitute the expression for $\mathbf{K}$ and $\mathbf{f}$ to the Eq.(1) we obtain that the derivative of the compliance does not depend on y:

$$\frac{\partial C}{\partial y} = 2\mathbf{u}^T\mathbf{g} - \mathbf{u}^T\mathbf{K}_0\mathbf{u} \qquad (2)$$

The Lagrange function for the optimization problem can be written as follows

$$L = C - \lambda(V - V_0) \, ,$$

where λ is the Lagrange multiplier.
The derivative of Lagrange function is equal

$$\frac{\partial L}{\partial y} = \frac{\partial C}{\partial y} - \lambda \frac{1}{p} y^{\frac{1}{p}-1}$$

At $p \to \infty$ this relationship can be simplified in the following way

$$\frac{\partial L}{\partial y} = \frac{\partial C}{\partial y} - \frac{\lambda}{y} \qquad (3)$$

Introduce new design variables x

$$x(y) = \tanh^{-1}(2y-1) \, .$$

The value y from the range (0, 1) is mapped to (-∞,∞).
The inverse function is

$$y(x) = \frac{1+\tanh(x)}{2} \, .$$

The graph of this function is presented in Figure 1.

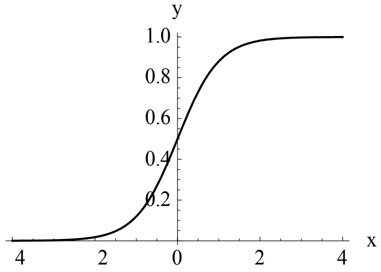


Figure 1: The function $y(x)$

The derivative $dy/dx = 2(y-1)y$ has simple form.

Using these new variable the following algorithm can be built. We will use the anti-gradient of the Lagrange function to stream to minimum

$$x^{new} = x - h\frac{\partial L}{\partial x} = x - h\frac{\partial L}{\partial y}\frac{dy}{dx}$$

The derivative dy/dx plays role as the relaxation multiplier for changing of design variables on the boundaries: $y=0$ and $y=1$. It can be enforced by introducing the power coefficient $\mu \geq 1$. Value of $\mu = 2$ is used for all calculations.

$$x^{new} = x - h\frac{\partial L}{\partial y}\left(\frac{dy}{dx}\right)^{\mu}$$

The chart $\left(2\,dy/dx\right)^{\mu}$ is presented in Figure 2.

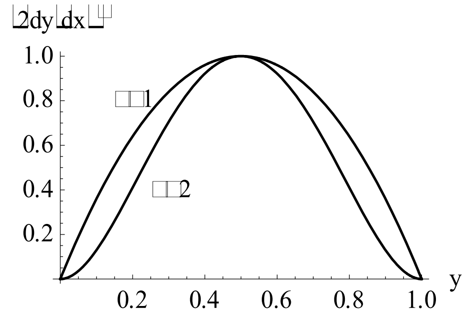


Figure 2: Relaxation multiplier $\left(2\,dy/dx\right)^{\mu}$

By using Eq.(3) new recalculation formula can be written

$$x^{new} = x - h(y-1)^{1+\mu}y^{\mu}(\frac{\partial C}{\partial y}y - \lambda).$$

where with taken into consideration Eq.(2)

$$\frac{\partial C}{\partial y}y = 2\mathbf{u}^T\mathbf{f} - \mathbf{u}^T\mathbf{K}\mathbf{u}\,.$$

The Lagrange multiplier can not be determined from the equality $V(y) = V_0$ since this expression is degenerated when $p \to \infty$. The value of λ is found from the condition $x^s_{nd} + x^s_{nd+1} = 2x_{del}$, where $x^s = sort(x^{new})$ is sorted list of new values of design variables x^{new}, the index nd is equal to the number of removing elements and x_{del} is the specified value. The secant method is used and usually 1-2 iteration are required because the values of λ are weakly changing by steps.

5. Numerical results

The design domain, boundary conditions, and external load for the optimization of considered topology optimization examples are shown in Figures 3 and 4.

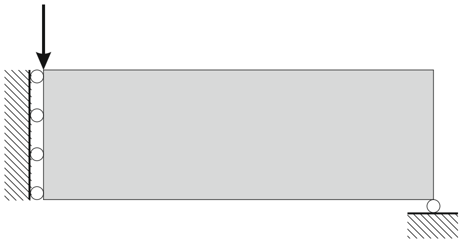


Figure 3: The design domain, boundary conditions, and external load for the optimization of a symmetric MBB beam

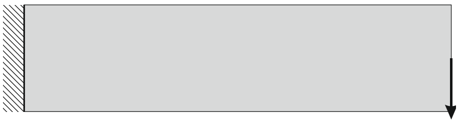
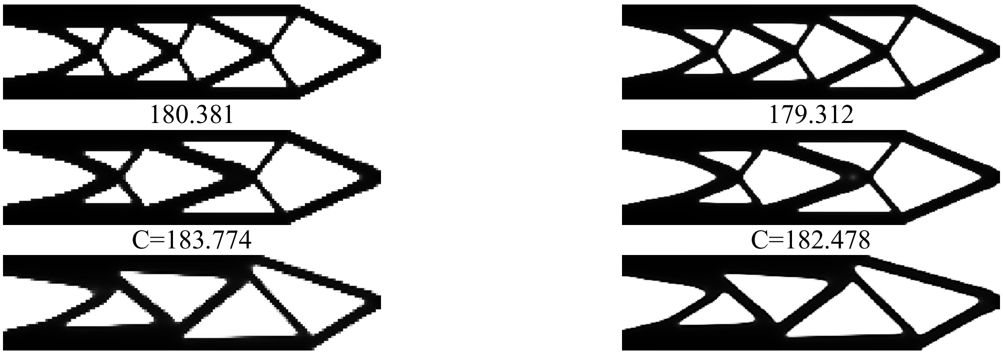


Figure 4: The design domain, boundary conditions, and external load for the optimization of cantilever

The optimization results for the fixed external forces are shown in Figures 5 and 6.

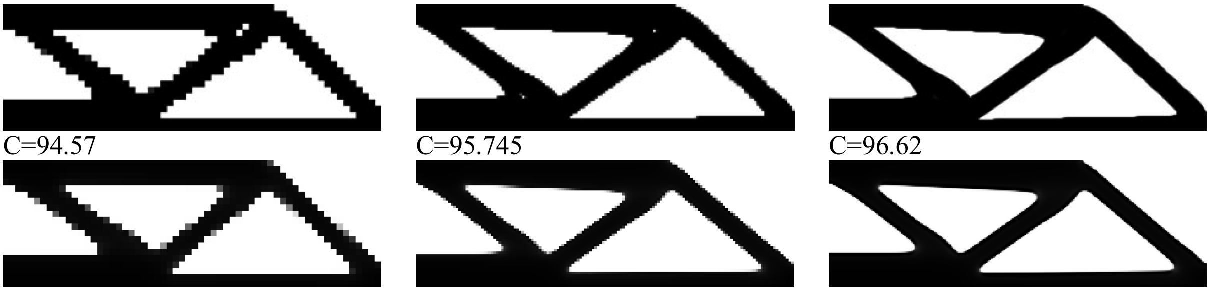


180.381 179.312

C=183.774 C=182.478

C=183.444 C=183.029

Figure 5: Results for different meshes and filtering parameters

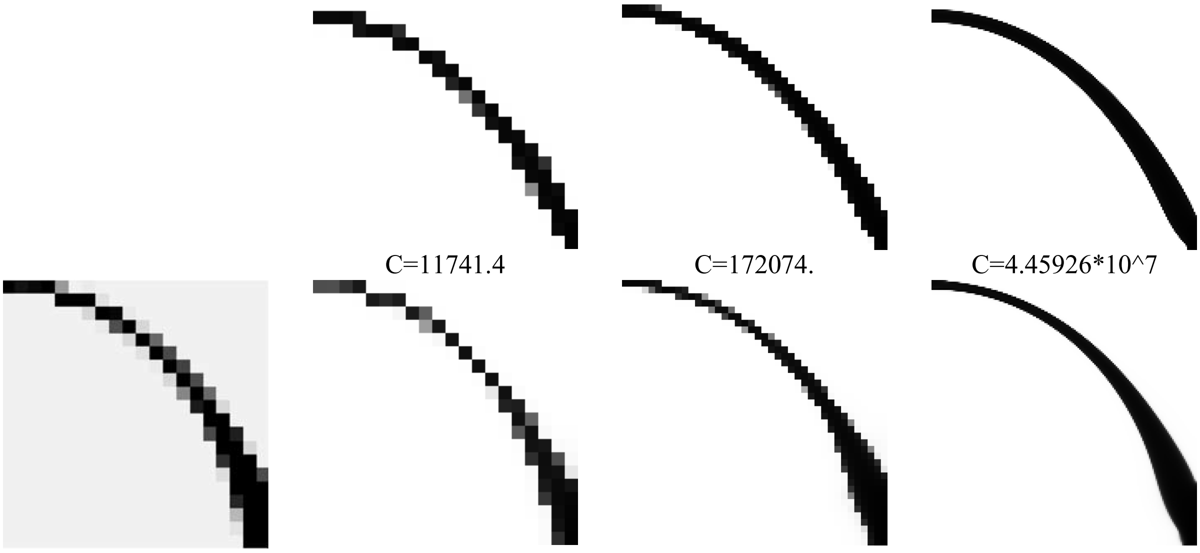


C=94.57 C=95.745 C=96.62

C=94.6165 C=95.2427 C=96.1395

Figure 6: Results for different meshes and filtering parameters (top [5], bottom [this method])

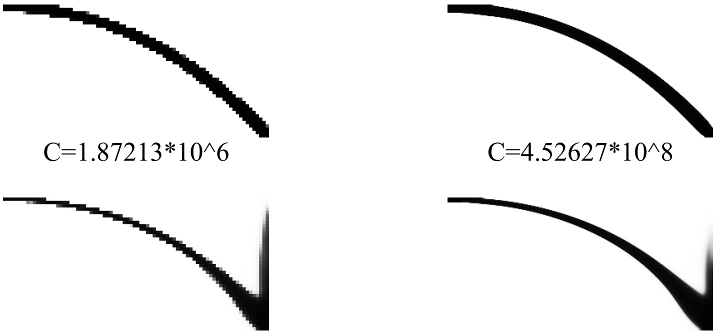
The optimization results for the self-weight forces are shown in Figures 7-9.



C=11741.4 C=172074. C=4.45926*10^7

C=1.26263*10^8 C=143091. C=3.98207*10^7

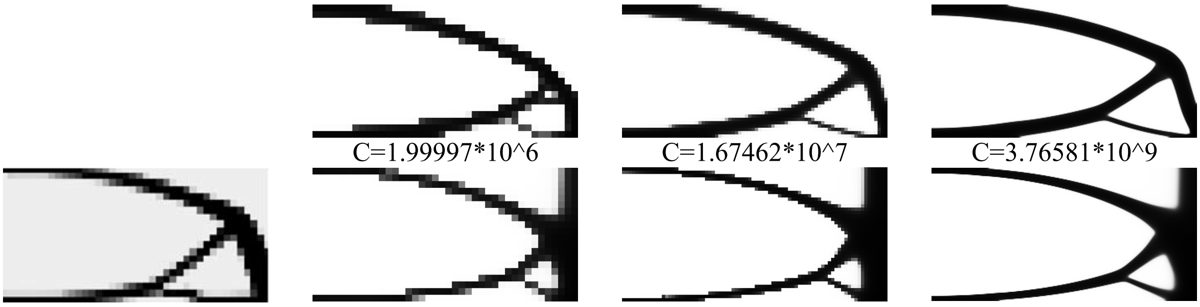
Figure 7: Results for different meshes (first column [2], column 2-4 [this method])



C=1.87213*10^6 C=4.52627*10^8

C=1.3003*10^6 C=3.08196*10^8

Figure 8: Results for different meshes



C=1.99997*10^6 C=1.67462*10^7 C=3.76581*10^9

C=1.45041*10^6 C=1.22025*10^7 C=2.80808*10^9

Figure 9: Results for different meshes (first column [2], column 2-4 [this method])

6. References

[1] M.P. Bendsoe and O. Sigmund, *Topology Optimization: Theory, Methods and Applications*, Springer-Verlag, Berlin, 2003.

[2] M. Bruyneel and P. Duysinx, Note on topology optimization of continuum structures including self-weight, *Structural and Multidisciplinary Optimization*, 29, 245–256, 2005.

[3] X. Huang and Y.M. Xie, A further review of ESO type methods for topology problem, *Structural and Multidisciplinary Optimization*, 41(5), 671–683, 2010.

[4] A. Rietz, Sufficiency of a finite exponent in SIMP (power law) methods, *Structural and Multidisciplinary Optimization*, 21(2), 159–163, 2001.

[5] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, O. Sigmund, Efficient topology optimization in MATLAB using 88 lines of code, *Structural and Multidisciplinary Optimization,* 43(1), 1–16, 2011.

Appendix: Mathematica code

```
nu=0.3;
volume=0.5;
thickBound=0.4;
smooth=0.6;
dMax=0.5;
countMax=100;
smoothN=100;
(*Cantilever*)
mn={m,n}={4,1}*40;
mn1={m1,n1}=mn+1;
fta[a_,b_]:=Flatten[Transpose@Array[a,b],1];
nodes=N[fta[{##}&,mn1]];
elems=fta[With[{k=#1+#2*m1},{k-m1,k-m,k+1,k}]&,mn];
fixXs=Range[1,m1*n1,m1];
fixYs=fixXs;
fixNs=(fixXs*2-1)~Union~(fixYs*2);
nodeForces={{m1*(n1+1)/2,{0,-1}}};
Remove[(*n,*)mn,m1,n1,mn1,fta,fixXs,fixYs];

nElems=Length[elems];
nNodes=Length[nodes];
numDel=nElems-Floor[nElems*volume+0.5];

filter=Flatten[GaussianFilter[Partition[#,m],
{5*smooth,smooth},Padding->"Reversed"]]&;

elemK=Block[{k,m},
k=({3,1,-3,-1,-3,-1,0,1}+{-1,1,-1,3,1,-1,1,-3}*nu)
/({6,8,12,8,12,8,6,8}*(1-nu^2));
m={{2,3,4,5,6,7,8},{8,7,6,5,4,3},{6,7,4,5,2},
{8,3,2,5},{2,3,4},{8,7},{6},{0}};
m=PadLeft[m,{8,8}];
m=IdentityMatrix[8]+m+Transpose[m];
k[[#]]&/@m];

energyFEM[{matIJ_,gather_,pos0_,f_,flatten3_},thicks_]:=
Block[{matV,disp},matV=flatten3[Total/@Extract[Flatten[
With[{p=Partition[elemK,{2,2}]},p*#&/@thicks],2],gather]];
matV[[pos0]]=0;
disp=Partition[LinearSolve[SparseArray[matIJ->matV],f,
Method->Cholesky],2];
With[{d=Flatten[disp[[#]]]},Max[0.,d.elemK.d]]&/@elems];

prepare=Block[{matIJ,gather,nNodes,pos0,f,flatten3},
matIJ=Flatten[Outer[List,#,#]&/@elems,2];
gather=SplitBy[Ordering[matIJ],matIJ[[#]]&];
gather=Block[{i},Replace[gather,{_,i_}->i,{2}]];
matIJ=matIJ[[gather[[All,1]]]];
gather=List/@gather;
nNodes=Max[elems];
matIJ=Flatten[Transpose[With[{p=Transpose[2*matIJ-1]},
Transpose[p+#]&/@Tuples[{0,1},2]]],1];
Block[{fix=ConstantArray[False,2*nNodes]},
fix[[fixNs]]=True;
pos0=Union@@(Flatten[Position[Extract[fix,List/@#],True,1]]&
/@Transpose[matIJ]);
pos0=Extract[pos0,Position[Unequal@@@matIJ[[pos0]],True,1]];];
f=ConstantArray[{0,0},nNodes];
(f[[#[[1]]]]=#[[2]])&@Transpose[nodeForces];
f=Flatten[f];f[[fixNs]]=0;f=SparseArray[f];
flatten3=Compile[{{v,_Real,3}},Flatten[v]];
{matIJ,gather,pos0,f,flatten3}];

yFun=Compile[{{x,_Real,1}},(1.+Tanh[x])/2.];
xFun=Compile[{{y,_Real,1}},ArcTanh[2.y-1.]];

xBound=xFun[{thickBound}][[1]];
funBound=Mean[Sort[#][[{numDel,numDel+1}]]]-xBound&;

Block[{dMax1=dMax/smoothN,
rh=2.^(1/smoothN),
thicks=ConstantArray[thickBound,nElems],
x=ConstantArray[xBound,nElems],
energyPrev=ConstantArray[0.,nElems]},
Block[{lam,dlam,h,energy},
Do[
energy=energyFEM[prepare,Max[10.^-7,#]&/@thicks]*thicks;
energy=filter[energy];
energy=0.5(energyPrev+energy);
energyPrev=energy;
If[count==1,lam=-Sort[energy][[numDel+1]];
dlam=0.001lam;
h=dMax/(smoothN*Mean[energy]);];
Block[{d=0.,u=x,u0,u1,dFilter,dDer,dAll,w,f1,f2},
Do[
u0=u;
u=filter[u];
dFilter=Max[Abs[u-u0]];
u0=u;
w=h With[{y=yFun[u]},y*(1.-y)^2];
u+=w energy;
Do[
f1=funBound[u+w (lam-dlam)];
f2=funBound[u+w (lam+dlam)];
If[Abs[f1+f2]<1000000.Abs[f1-f2],
lam+=(f1+f2)/(f1-f2)dlam;];
u1=u+w lam;
If[Abs[funBound[u1]]<thickBound*0.001,Break[];];
,{j,100}];
u=u1;
dAll=Max[Abs[u-x]];
If[dMax<dAll,Break[];];
dDer=Max[Abs[u-u0]];
d+=dMax1;
If[dAll<d/rh||dDer<dFilter,h*=rh;];
,{smoothN}];
x=u;
];
thicks=yFun[x];
Print[Graphics[Raster[Partition[1.-thicks,m]]],"  ",count,"
",Sqrt[Mean[(x-xBound)^2]]," h=",h];,{count,countMax}];];];
```