A first laboratory exercise in topology optimization using matlab.

Andreas Rietz
Department of Mechanical Engineering / Department of Mathematics
Linköping University
SE-58183 Linköping, Sweden
Abstract

The purpose of this laboratory exercise is to give you experience with the use of topology optimization as a first design tool in doing a construction. A 99 line topology optimization code written in Matlab will be used. The code is available at

http://www.topopt.dtu.dk.

The details are explained in the paper “A 99 line topology optimization code written in Matlab” which is published in the journal “Structural and Multidisciplinary Optimization” by Ole Sigmund, dept. of solid mechanics at Technical University of Denmark. The matlab code is also included as an appendix.

This matlab code should give understanding of the basics of a topology optimization program, and the user can easily modify the code.

The laboratory exercise is divided into two parts. The first part is about the design of bicycles and is intended to make you familiar with the methodology and the matlab code. The second part consists of a bunch of projects that can be treated in the same way as the bicycle. It is advisable to choose one of the projects in the second part.

LiTH-IKP March 2000

Some information is updated and some errors are corrected.

LiTH-MAI May 2006
Chapter 1

Introduction

More information about the matlab code and methodology used in this laboratory exercise is found on

http://www.topopt.dtu.dk

The finite element mesh we use is composed by squares of size 1 times 1. The density of material in each square is denoted $x$, and we assume that $0.001 \leq x \leq 1$. The reason for having a lower bound greater than zero is to make the stiffness matrix in the finite element calculation nonsingular. The correspondence of the indices of the density matrix and the finite element mesh is displayed in figure 1.1. The displacements are enumerated columnwise from left to right. The displacements of an arbitrary finite element is shown in figure 1.2. The displacement is denoted by the variable $u$. 
Figure 1.1: The index order of the density matrix.

Figure 1.2: The displacements of a single finite element.
Chapter 2

Design of a bicycle.

The difference between man’s and lady’s bicycles is the design of frame. The lady’s bicycles are designed to make it easier to mount, whereas man’s bicycles are more solid. In this section we shall try to find the “best” frame.

The design problem is illustrated in figure 2.1. Our task is to construct the front part of the frame to make the bicycle as stiff as possible. This part connects the handlebar and the saddle. The design domain is illustrated schematically in figure 2.2. One has to consider the following:

- The handlebar produces a force of in the vertical direction.
- The rear frame acts as a support.
- We have to declare a part of the design domain void to make place for the front wheel.

2.1 Load and support.

Let us first only consider the load and support. These are easily taken care of by changing line 78 and 79 in the matlab code. Open the matlab code in some text editor rewrite these lines:

78 F(2,1) = 1;
79 fixeddofs = 2*nelx*(nely+1)+1:2*(nelx+1)*(nely+1);

Save the code and start matlab in the same directory. Then start the 99-line code by

>> top(20,20,0.33,3.0,1.5);

Your result will hopefully be similar to figure 2.3. The design domain is discretized into 20 times 20 finite elements and black elements are filled while white elements are empty.

Maybe you feel that the measure of the force is unrealistic and that the Young’s modulus should be corrected on line 87. Usually \( E \approx 2 \cdot 10^{11} \text{N/m}^2 \). Also the finite element size is
defined to be 1 times 1 units. These measures has to be changed to get correct values of the compliance, but it is not necessary to do these changes to get correct results from the optimization, since they are only corrections of the scales. However if you wish to do these changes you better correct line 39 to keep the accuracy while solving the equations:

39 while ((l2-l1)/l2 > 1e-4)

2.2 Penalization and filter radius.

The syntax of the top-function is

>> top(nelx,nely,volfrac,penal,rmin)
where the variables denote the following:

- $nelx$ is the number of finite elements in the horizontal direction.
- $nely$ is the number of finite elements in the vertical direction.
- $volfrac$ is the fraction of volume in the design domain.
- $penal$ is the penalization of intermediate densities. A high penalization will make the solution black and white, that is the finite elements will either be filled or empty. Usually $penal = 3$ sufficies. A penalization $penal = 1$ means that there is no penalty of the intermediate densities.
- $rmin$ is a filter radius for a filter which makes the design mesh-independent.  

Now redo the task in the previous section and change the values of these parameters. Try to answer the following questions:

1. Is the design dependent on the mesh-size? Compare the outcome from the following commands:

```matlab
>> top(12,12,0.33,3.0,0.9);
>> top(16,16,0.33,3.0,1.2);
>> top(20,20,0.33,3.0,1.5);
```

\[1\] While this laboratory exercise was written, there was no mathematical justification that this filter yields mesh independancy, but lots of practical tests indicate this.
Note that the filter radius has to be increased proportionally to the discretization in the vertical and the horizontal direction.

2. How much better design do we get if we do not require it to be black and white? Check the compliance of the final design from the following two test cases:

```matlab
>> top(20,20,0.33,1.0,1.5);
>> top(20,20,0.33,3.0,1.5);
```

3. Let us now study the mesh-independency filter. The filter is turned off by choosing a filter radius $r_{\text{min}}$ less than one or by making line 25 inactive:

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

Do not forget to remove the \% afterwards.
Look at the outcome of the testcase

```matlab
>> top(20,20,0.33,3.0,1.5);
```

with and without filter, and try different values of $r_{\text{min}}$. 

---

Figure 2.3: The first result, taking care of support and load.
2.3 Defining void regions.

The result in figure 2.3 does not leave any void region for the front wheel. Let us call the finite elements in this void passive, and define a matrix with zeros at free elements and ones at passive. Add the following lines to the matlab code between line 4 and 5 to do this:

```matlab
for ely = 1:nely
    for elx = 1:nelx
        if ((elx)^2+(ely-nely)^2) < (0.65*nelx)^2
            passive(ely,elx) = 1;
        else
            passive(ely,elx) = 0;
        end
    end
end
x(find(passive))=0.001;
```

The last command initializes all the elements in the void region to the low value 0.001. We also have to update line 27 and 37 and insert an additional line between 41 and 42:

```matlab
27 \[x\] = OC(nelx,nely,x,volfrac,dc,passive);
37 function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
41b xnew(find(passive)) = 0.001;
```

Do these changes and redo the command:

```matlab
>> top(20,20,0.33,3,1.5)
```

This will end up in something more similar to a man’s bicycle, the front frame is shown in figure 2.4.

To make a lady’s bicycle we have to define another void, which makes it easier for the user to mount the bicycle, but no clues will be left for this task. It is interesting to note the change in compliance for this additional void.
Figure 2.4: Making a void for the front wheel results in a man’s bike.
Chapter 3

Projects

A common hint for all projects is to change line 4 to:

\[ x(1:\text{nely},1:\text{nelx}) = 0.001; \]

which makes more sense to the optimality criteria update function if \( \text{volfrac} \) is chosen greater than 1.

3.1 Making a supporting frame for a billboard.

Figure 3.1 illustrates a billboard that is exposed to a windfield. The support of the billboard needs some kind of supporting frame to make it stiffer.

A two dimensional illustration of the loads and support is given in figure 3.2. A volume-fraction of approximately 0.2 of the design domain is expected. Let us assume that the billboard is very stiff \(^1\), and the available legs are made of the same material as the frame.

3.2 Reinforcement of a cablecar support.

The cablecar in figure 3.3 needs reinforcement of its support. Suggest a reinforcement from the information given in figure 3.4.

The caretaker of the cablecar would also like that you suggest improvements so that the support can carry two cablecars, which is illustrated in figure 3.5. This generalization has to be treated by multiple loading methodology, as discribed in the paper “A 99 line topology optimization code written in matlab” by Ole Sigmund in the journal Structural and Multidisciplinary Optimization.

\(^1\)This can be accomplished by assuming \( x = 10 \) in some finite elements.
3.3 Construction of a wheelchair.

A wheelchair constructor is considering to use topology optimization in his next design. It has to be a light and solid construction.

Figure 3.6 illustrates the general layout of the design and figure 3.7 shows the location of the loads and supports in more detail. Note that the frontwheel only gives support in the horizontal direction. About 20% of the design domain can be filled.
Figure 3.2: Load and support of the billboard.
Figure 3.3: Reinforcement of the support of a cablecar.
Figure 3.4: Load and support.
Figure 3.5: Multiple loading has to be used if the support is to carry two cablecars.
Figure 3.6: Construction of a wheelchair.
Figure 3.7: A topology optimizers view of the wheelchair.
MATLAB CODE

1 % % % A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 % %
2 function top(nelx,nely,volfrac,penal,rmin);
3 % INITIALIZE
4 x(1:nely,1:nelx) = volfrac;
5 loop = 0; change = 1.;
6 % START ITERATION
7 while change > 0.01
8   loop = loop + 1;
9   xold = x;
10 % FE-ANALYSIS
11 [U]=FE(nelx,nely,x,penal);
12 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
13 [KE] = lk;
14 c = 0.;
15 for ely = 1:nely
16   for elx = 1:nelx
17     n1 = (nely+1)*(elx-1)+ely;
18     n2 = (nely+1)* elx +ely;
19     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);
20     c = c + x(ely,elx)^penal*Ue'*KE*Ue;
21     dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
22   end
23 end
24 % FILTERING OF SENSITIVITIES
25 [dc] = check(nelx,nely,rmin,x,dc);
26 % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
27 [x] = OC(nelx,nely,x,volfrac,dc);
28 % PRINT RESULTS
29 change = max(max(abs(x-xold)));
30 disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
31       ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
32       ' ch.: ' sprintf('%6.3f',change )])

18
% PLOT DENSITIES
33 colormap(gray);imagesc(-x);axis equal;axis tight;axis off;pause(1e-6);
35 end
36 %%%%%%%%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37 function [xnew]=OC(nelx,nely,x,volfrac,dc)
38 l1 = 0; l2 = 100000; move = 0.2;
39 while (l2-l1 > 1e-4)
40 lmid = 0.5*(l2+l1);
41 xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid))));
42 if sum(sum(xnew)) - volfrac*nelx*nely > 0;
43 l1 = lmid;
44 else
45 l2 = lmid;
46 end
47 end
48 %%%%%%%%%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49 function [dcn]=check(nelx,nely,rmin,x,dc)
50 dcn=zeros(nely,nelx);
51 for i = 1:nelx
52 for j = 1:nely
53 sum=0.0;
54 for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
55 for l = max(j-round(rmin),1):min(j+round(rmin),nely)
56 fac = rmin-sqrt((i-k)^2+(j-l)^2);
57 sum = sum+max(0,fac);
58 dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
59 end
60 end
61 dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
62 end
63 end
64 %%%%%%%%%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65 function [U]=FE(nelx,nely,x,penal)
66 [KE] = lk;
67 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
68 F = sparse(2*(nely+1)*(nelx+1),1); U = sparse(2*(nely+1)*(nelx+1),1);
69 for ely = 1:nely
70 for elx = 1:nelx
71 n1 = (nely+1)*(elx-1)+ely;
72 n2 = (nely+1)* elx +ely;
73 edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
74 K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
F(2,1) = -1;

fixeddofs = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);

% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:) = 0;

% ELEMENT STIFFNESS MATRIX

function [KE]=lk
E = 1.;
u = 0.3;
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)];

% This Matlab code was written by Ole Sigmund, Department of Solid
% Mechanics, Technical University of Denmark, DK-2800 Lyngby, Denmark.
% Please send your comments to the author: sigmund@fam.dtu.dk
% %
% The code is intended for educational purposes and theoretical details
% are discussed in the paper
% "A 99 line topology optimization code written in Matlab"
% by Ole Sigmund (To appear in Structural Optimization).
% %
% The code as well as a postscript version of the paper can be downloaded
% from the web-site: http://www.topopt.dtu.dk
% %
% Disclaimer:
% The author reserves all rights but does not guaranty that the code is free
% from errors. Furthermore, he shall not be liable in any event caused by
% the use of the program.