**Topology optimization on Matlab**

**J. Morlier\*[[1]](#footnote-1)**

The Finite Element mesh is composed of quadrangle of size 1 \* 1. The density called x, is defined tq 0.001 ≤ x ≤ 1. (the bound inf is non-zero, so that K is not singular). The correspondence between the elements of the density matrix and the model EF is given in figure 1.1.

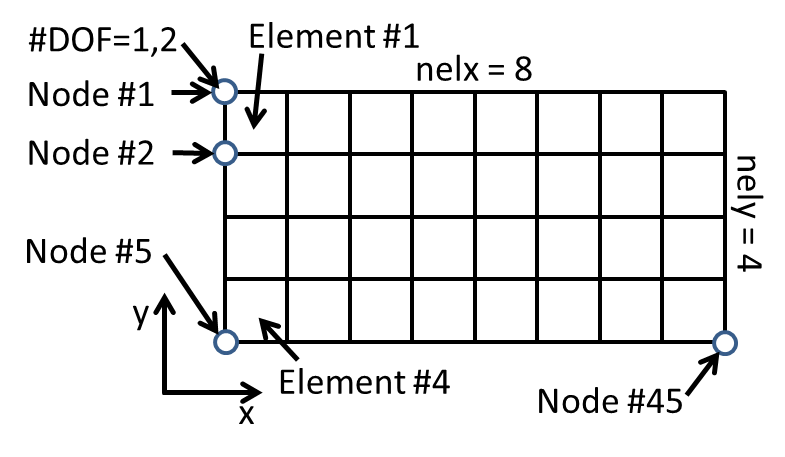


Figure 1: Index

**Let’s try to do this exercise (by reading the article associated with code top.m): Optimizing the design of a bicycle frame.**

The design domain is given below (Figure 3)

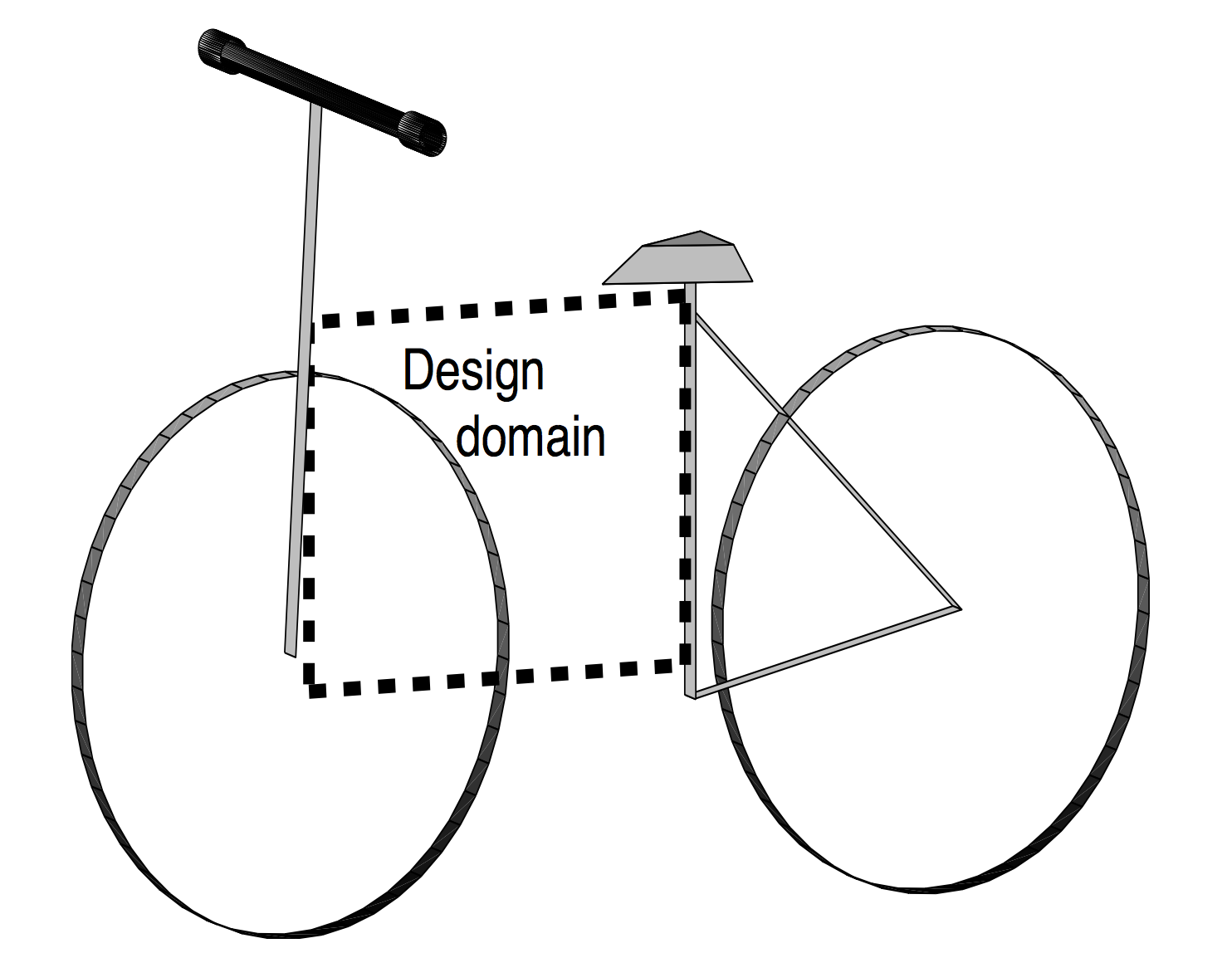


Figure 3: Design domain

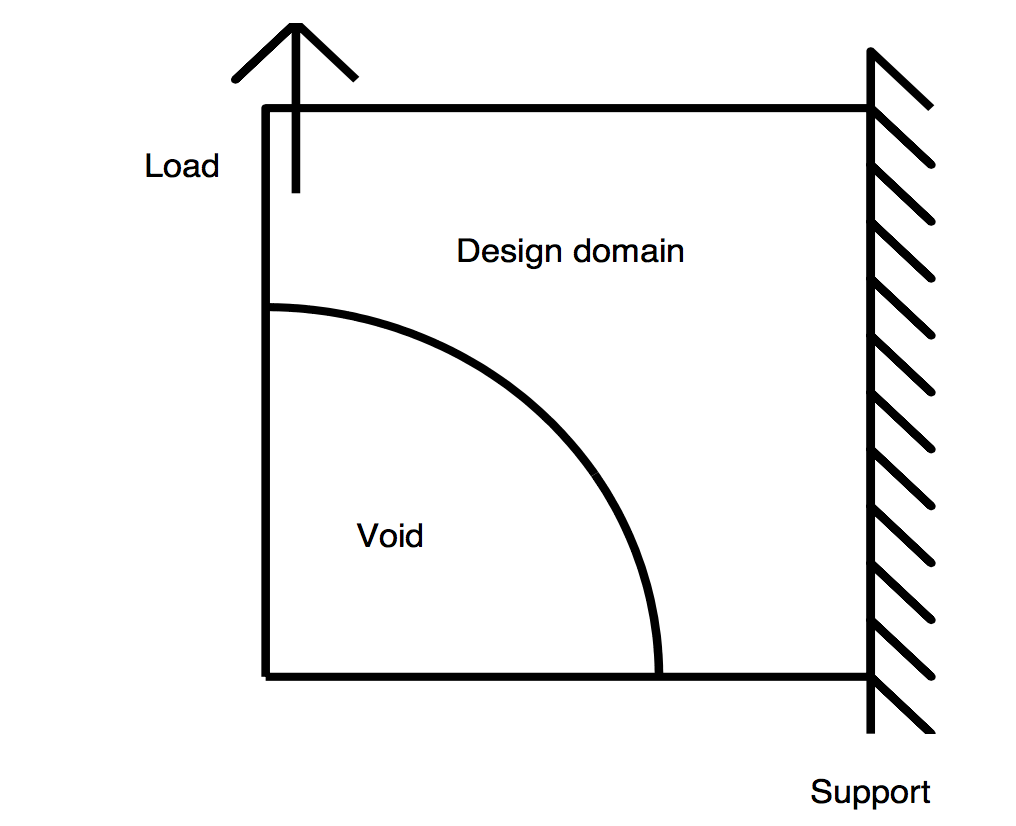
The optimal design problem is schematized in the form of figure 4

Figure 4: Simplified scheme

The reaction of the wheel produces a vertical force at the front, the space of the wheel is modeled by the void, and finally the bar under the saddle is recessed.

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

where the variables denote the following:

• nelx is the number of finite elements in the horisontal 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.

Question 0:

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

The results is the default’s MMB testcase 

F(2,1) = - 1;

specifies that there is a force of -1 at the second row and first column of the vector. According to the numbering convention of this code, this is to say that in the y direction of the first node, there is a downward force of magnitude 1.

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

This line specifies the nodes with fixed DOFs. [1:2:2\*(nely+1)] are x directions of all nodes to the left side of the structure, and 2\*(nelx+1)\*(nely+1) is the y direction of the last node (right bottom corner). See figure below:

**Question 1 :** Save the top.m file to topB.m

Modify boundary conditions and loading

78 F(2,1) = 1; % comments here

79 fixeddofs = 2\*nelx\*(nely+1)+1:2\*(nelx+1)\*(nely+1); % comments here

Save and launch the code:

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

…

Copy Paste the results. Comments must be placed here

…

**Question 2 :**

Is the design dependent on the mesh? Compare the results:

>> 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);

Note that the radius of the filter must be increased in proportion to the discretization (in x, y)

**Question 3 :**

Compare the results of the different commands:

>> top(20,20,0.33,1.0,1.5);

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

Influence of penal? Check the associated compliances.

**Question 4 :**

Filter Analysis:

You can disable the filter by setting rmin to less than 1 or by making line 27 inactive:

27 % [dc] = check(nelx,nely,rmin,x,dc);

Remove % for filtering.

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

Try different rmin values.

**Question 5 :**

Definition of empty regions: save the file as topB2.m

The previous results do not take into account the empty region (wheel).

We will have to classify the elements (1 passive, 0 free). Add between lines 6 and 7 to create the vacuum. (Passive = not taken into account in the optimization as design variable).

for ely = 1:nely

for elx = 1:nelx

if ((elx)^2+(ely-nely)^2) < (0.65\*nelx)^2 passive(ely,elx) = 1; % comments here

else

passive(ely,elx) = 0; % comments here

end

end

end

x(find(passive))=0.001;

The last command initializes the empty region to 0.001.

We also need to update line 29 and 39 and add the additional line between 43 and 44:

29 [x] = OC(nelx,nely,x,volfrac,dc,passive);

39 function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)

43 xnew(find(passive)) = 0.001;

Type

>> topB2(20,20,0.33,3,1.5)

What do you notice?

…

Copy Paste the results. Comments must be placed here

…

Additional Information

Movements are listed by column from left to right. The displacement (variable u) of any FE is given below.

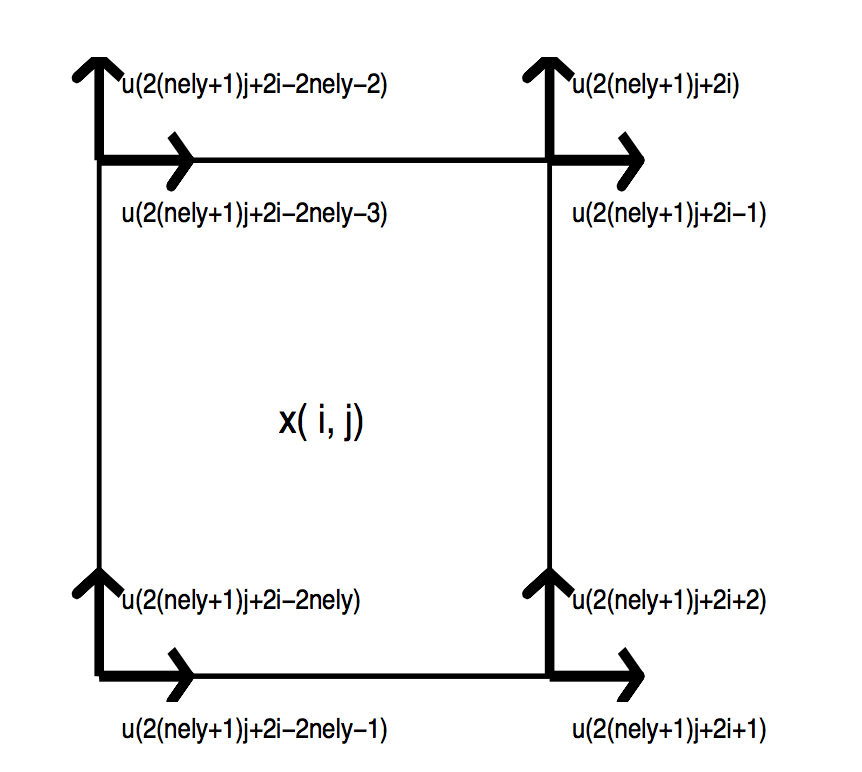


Figure 4: u function of element x(i,j)

### Plotting displacements

Insert the following lines in your program instead of the current plotting line:

% colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);   
colormap(gray); axis equal;   
for ely = 1:nely   
   for elx = 1:nelx   
     n1 = (nely+1)\*(elx-1)+ely;   
     n2 = (nely+1)\* elx +ely;   
     Ue = 0.005\*U([2\*n1-1;2\*n1; 2\*n2-1;2\*n2; 2\*n2+1;2\*n2+2; 2\*n1+1;2\*n1+2],1);   
     ly = ely-1; lx = elx-1;   
     xx = [Ue(1,1)+lx Ue(3,1)+lx+1 Ue(5,1)+lx+1 Ue(7,1)+lx ]';   
     yy = [-Ue(2,1)-ly -Ue(4,1)-ly -Ue(6,1)-ly-1 -Ue(8,1)-ly-1]';   
     patch(xx,yy,-x(ely,elx))   
   end   
end   
drawnow; clf;

### Efficient matrix assembly for large problems

To handle large problems, you may substitute the transparent and easy-to-read stiffness matrix assembly:

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

with the much more efficient assembly using a list of triplets:

I = zeros(nelx\*nely\*64,1); J = zeros(nelx\*nely\*64,1); X = zeros(nelx\*nely\*64,1);   
F = sparse(2\*(nely+1)\*(nelx+1),1); U = zeros(2\*(nely+1)\*(nelx+1),1);   
ntriplets = 0;   
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];   
     xval = x(ely,elx)^penal;   
     for krow = 1:8   
       for kcol = 1:8   
         ntriplets = ntriplets+1;   
         I(ntriplets) = edof(krow);   
         J(ntriplets) = edof(kcol);   
         X(ntriplets) = xval\*KE(krow,kcol);   
       end   
     end   
   end   
end   
K = sparse(I,J,X,2\*(nelx+1)\*(nely+1),2\*(nelx+1)\*(nely+1));

[Link](http://blogs.mathworks.com/loren/2007/03/01/creating-sparse-finite-element-matrices-in-matlab/#9) to more details on creating sparse FE matrices.

### The element mass matrix

m0 = [4/9 0 2/9 0 1/9 0 2/9 0   
   0 4/9 0 2/9 0 1/9 0 2/9   
   2/9 0 4/9 0 2/9 0 1/9 0   
   0 2/9 0 4/9 0 2/9 0 1/9   
   1/9 0 2/9 0 4/9 0 2/9 0   
   0 1/9 0 2/9 0 4/9 0 2/9   
   2/9 0 1/9 0 2/9 0 4/9 0   
   0 2/9 0 1/9 0 2/9 0 4/9]/4/(nelx\*nely);

### The strain displacement matrix

bmat = [-1/2 0 1/2 0 1/2 0 -1/2 0   
   0 -1/2 0 -1/2 0 1/2 0 1/2   
   -1/2 -1/2 -1/2 1/2 1/2 1/2 1/2 -1/2];

### The constitutive matrix for plane stress

Emat = E/(1-nu^2)\*[ 1 nu 0   
   nu 1 0   
   0 0 (1-nu)/2];

**Short biblio**

Bendsøe, M. P. and Sigmund, O.: 2004, Topology Optimization - Theory, Methods and Applications, Springer Verlag, Berlin Heidelberg.

Jensen, J. S.: 2009, A note on sensitivity analysis of linear dynamic systems with harmonic ex- citation, Report, Department of Mechanical Engineering, Technical University of Denmark.

Sigmund, O.: 1997, On the design of compliant mechanisms using topology optimization, Mechanics of Structures and Machines 25(4), 493􏰕524.

Sigmund, O.: 2001, A 99 line topology optimization code written in MATLAB, Structural and Multidisciplinary Optimization 21, 120􏰕127. MATLAB code available online at: www.topopt.dtu.dk.

Sigmund, O.: 2007, Morphology-based black and white filters for topology optimization, Structural and Multidisciplinary Optimization 33(4-5), 401􏰕424.

Svanberg, K.: 1987, The Method of Moving Asymptotes - A new method for structural opti- mization, International Journal for Numerical Methods in Engineering 24, 359􏰕373.

*Do you find the strength and the modulus of Young not realistic (as well as the dimensions, hence an unrealistic flexibility ...) ???*

*Rendez-vous for example Line 87 (E ≈ 2 · 1011N / m2) ... It is common to work in non-dimensional, get the result of TopOpt and then do parametric optimization. However, you could have corrected the units with line 39 to get a correct solution.*

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

1. **\***From website:

   http://www.topopt.dtu.dk [↑](#footnote-ref-1)