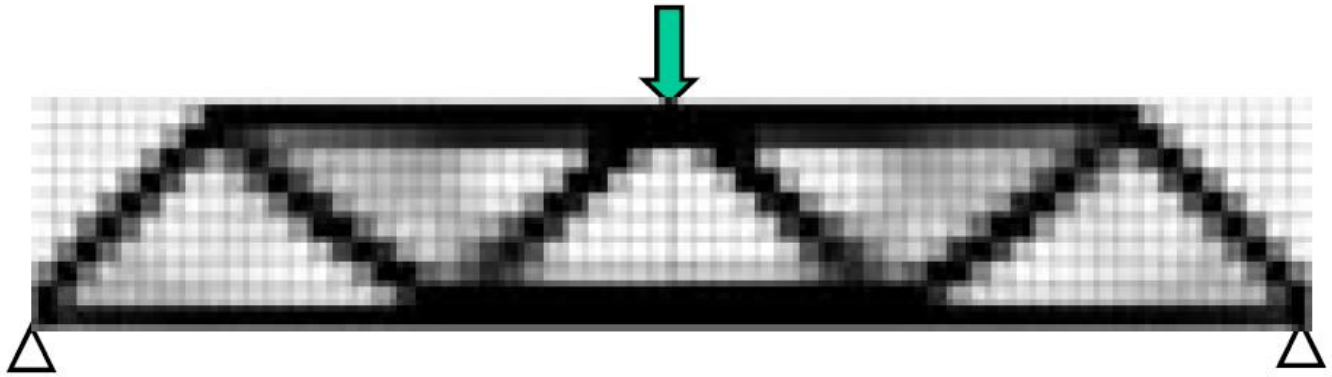
# Método SIMP



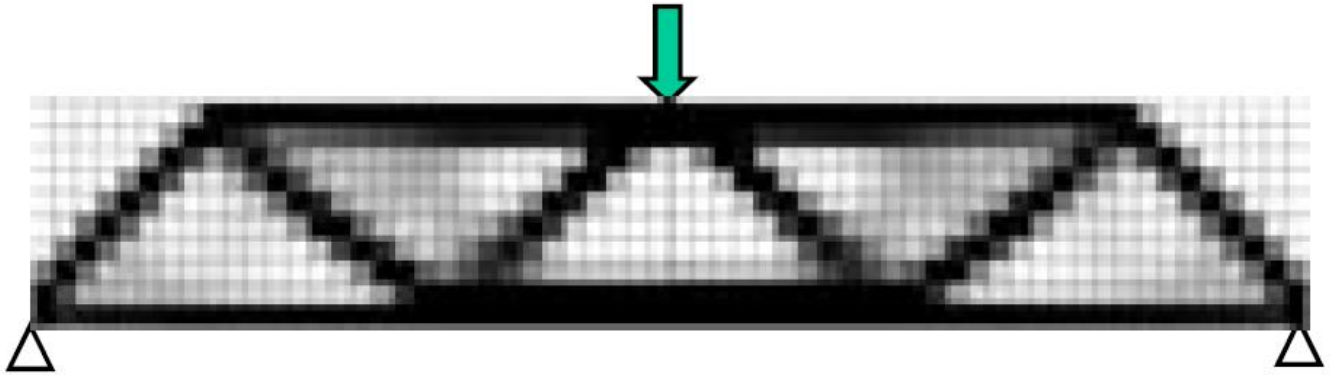
## PMR5250 – Aula 7

Docentes: Prof. Dr. Emílio C. N. Silva
Dr. Renato Picelli

# Métodos baseados em densidade

- Método SIMP.
- Dúvidas sobre análise de sensibilidade.
- Código educacional em Matlab.
- Fechar o loop do nosso código.
- Outros problemas.

# Solid Isotropic Material with Penalization (SIMP)

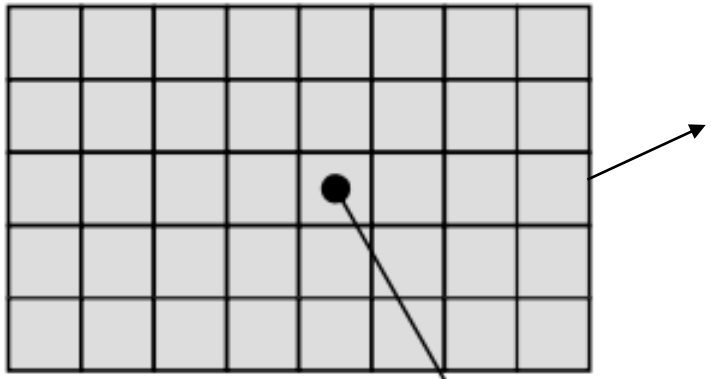


$$0 \leq x_{\min} \leq x_i \leq 1$$

$$0 \leq \rho_{\min} \leq \rho_i \leq 1$$

$x_i$ = variável de projeto ou pseudo-densidade

# Modelo de Material – “Material Law”




pseudo-densidade

SIMP

$$E_i = x_i^p E_0$$

$p$: penalidade

$E$: Módulo de elasticidade

$$\mathbf{K}_i = x_i^p \mathbf{k}_0$$

# Modelo de Material – “Material Law”

$$\mathbf{K}_i = \int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega$$

$$\mathbf{D} = \frac{x_i^p E_0}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix}$$

$$\mathbf{D} = x_i^p \mathbf{D}_0$$

SIMP

$$E_i = x_i^p E_0$$

$p$: penalidade

$E$: Módulo de elasticidade

$$\mathbf{K}_i = x_i^p \mathbf{k}_0$$

# Modelo de Material – “Material Law”

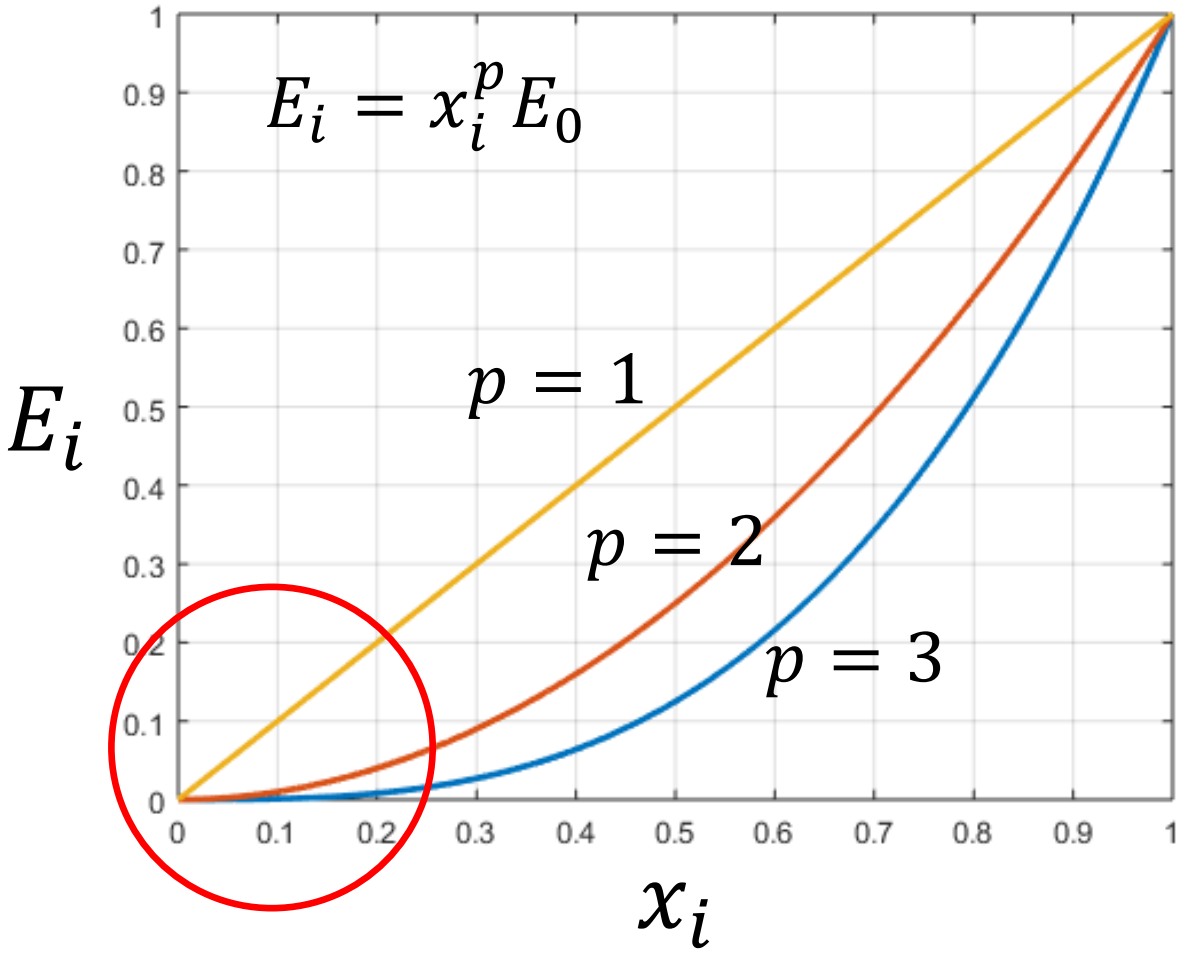
$$\mathbf{M}_i = \int_{\Omega} x_i \rho_0 \mathbf{H}^T \mathbf{H} d\Omega$$

$$\mathbf{M}_i = x_i \mathbf{m}_0$$

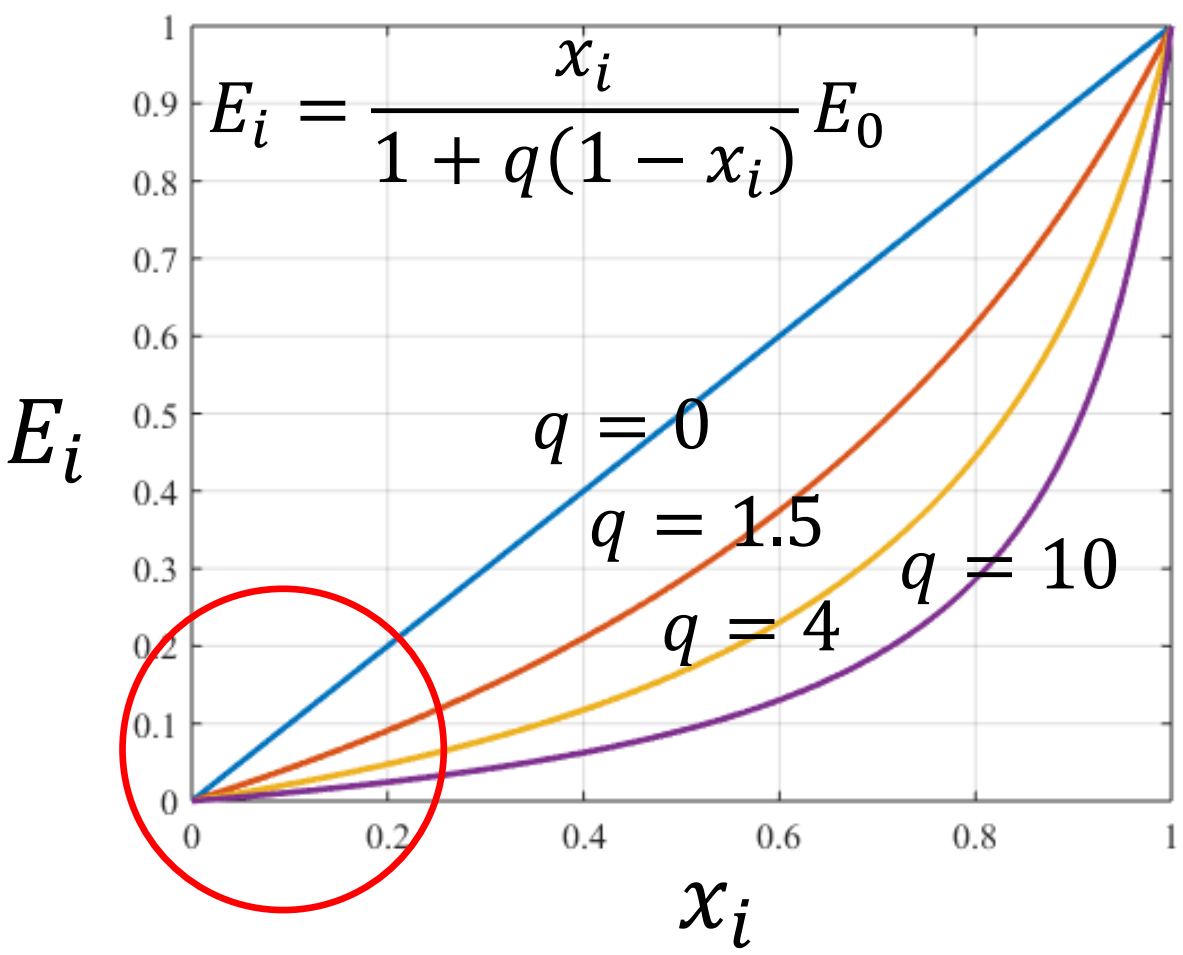
$$\rho_i = x_i \rho_0$$

Interpolação linear

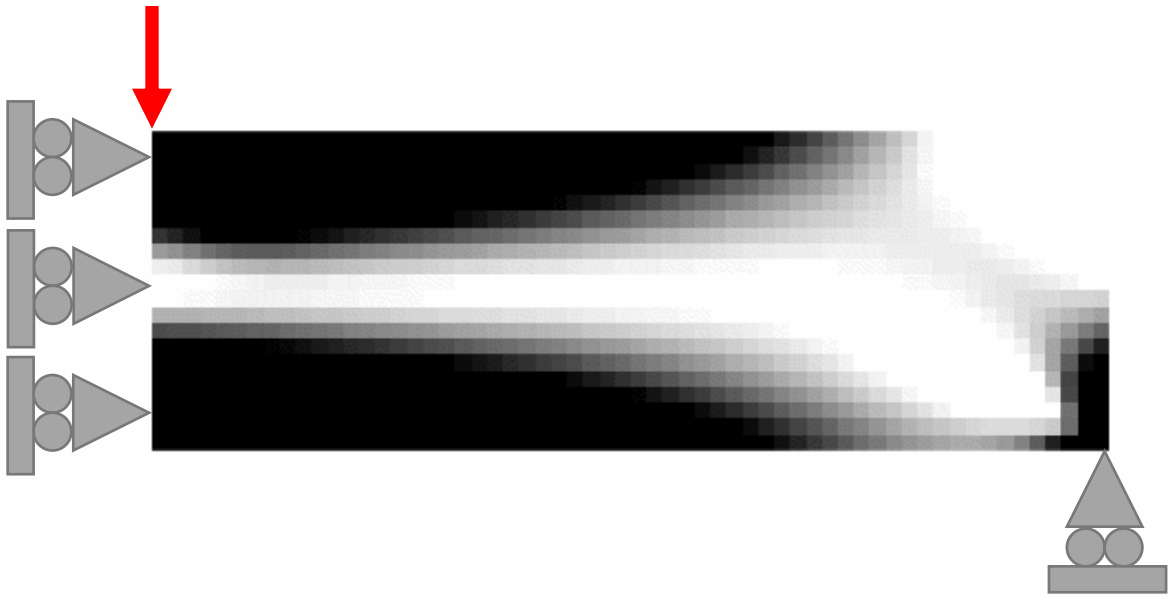
## SIMP




## RAMP



# Algoritmo



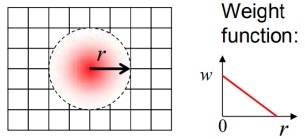
1. Definir parâmetros de material e otimização;
2. Discretizar domínio de projeto;
3. Realizar análise por elementos finitos;

$$E_i = \rho_i^p E_0 \qquad \mathbf{Ku} = \mathbf{f}$$

4. Calcular sensibilidades;

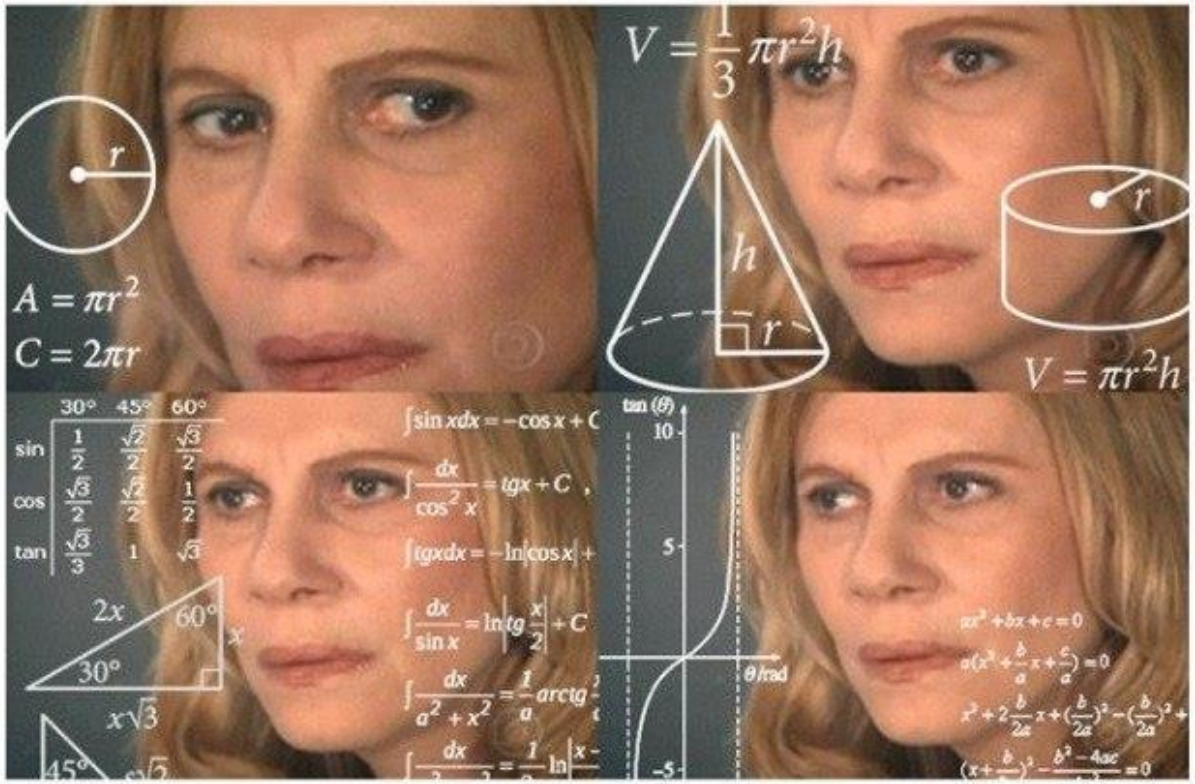
$$\frac{df(\mathbf{x})}{dx_i} = -\frac{1}{2} p x_i^{p-1} \mathbf{u}_i^T \mathbf{K}_i \mathbf{u}_i \qquad \frac{dg}{dx_i} = \frac{V_i}{V_0}$$

5. Filtro numérico;




6. Redistribuição de material (otimizador);
7. Atualização das variáveis (densidades);
8. Análise de convergência.

# Dúvidas sobre análise de sensibilidade?



# Possíveis erros da Tarefa 2

- Montagem da matriz global

$$\mathbf{K}(\mathbf{x}) = \mathbb{A} \rightarrow x_i^p \mathbf{K}_i \qquad \mathbf{M}(\mathbf{x}) = \mathbb{A} \rightarrow x_i \mathbf{M}_i$$

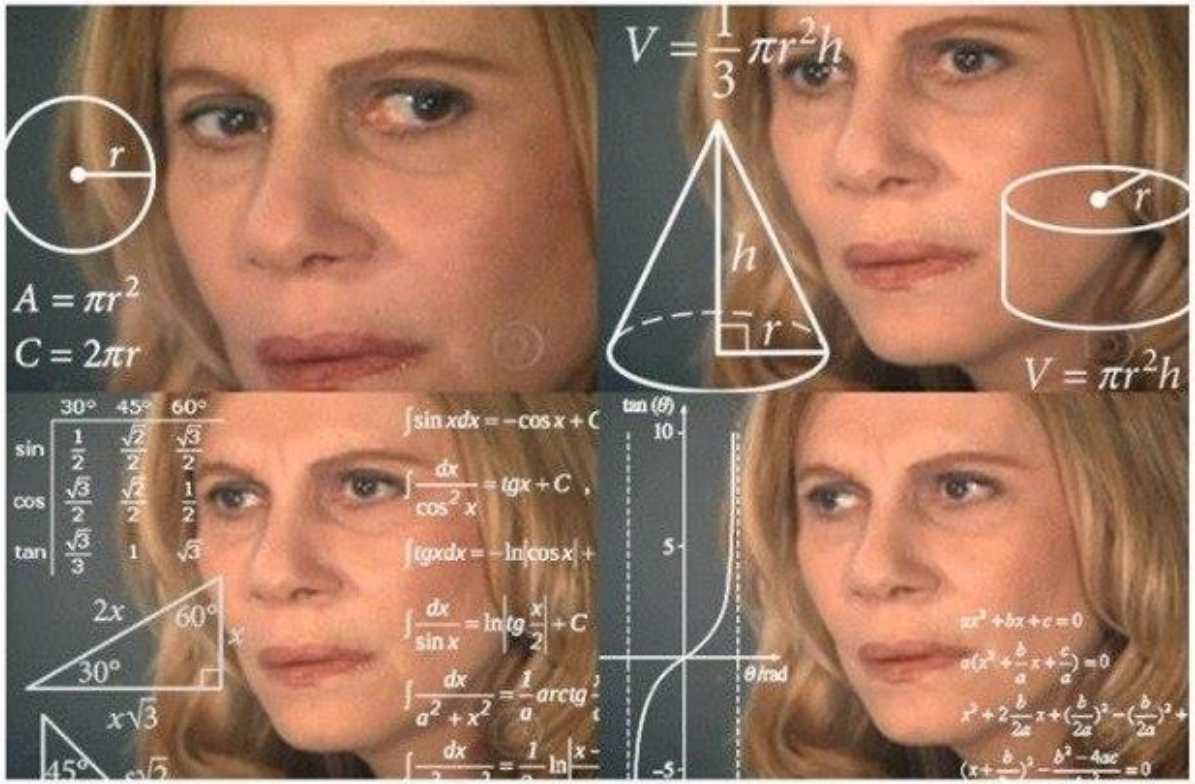
- Erro na expressão analítica

$$\frac{d\mu}{dx_i} = \frac{\mathbf{u}^{\mathrm{T}} \left( \frac{d\mathbf{K}}{dx_i} - \mu \frac{d\mathbf{M}}{dx_i} \right) \mathbf{u}}{\mathbf{u}^{\mathrm{T}} \mathbf{M} \mathbf{u}}$$

$$f(x) = \mu = \omega^2$$

$$\frac{d\mathbf{K}}{dx_i} = p x_i^{p-1} \mathbf{K}_i \qquad \frac{d\mathbf{M}}{dx_i} = \mathbf{M}_i$$

# Dúvidas sobre análise de sensibilidade?



Educational article

Struct Multidisc Optim 21, 120–127 © Springer-Verlag 2001

## A 99 line topology optimization code written in Matlab

O. Sigmund

# Exercício 1

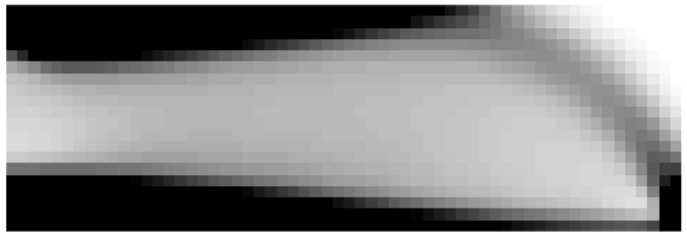
Primeiro exercício: entrar no site http://www.topopt.mek.dtu.dk/
Entrar na aba de *softwares* e baixar os códigos “99 line” e “88 line”.

Analisar a influência da penalização, raio do filtro e discretização.

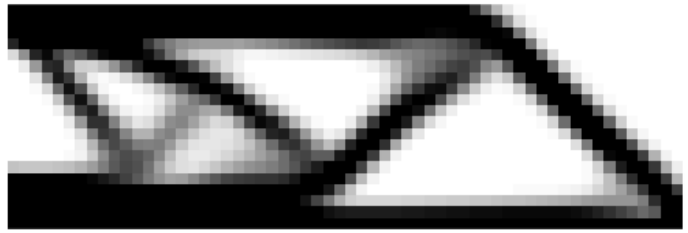
# Penalização

$$E_i = x_i^p E_0$$

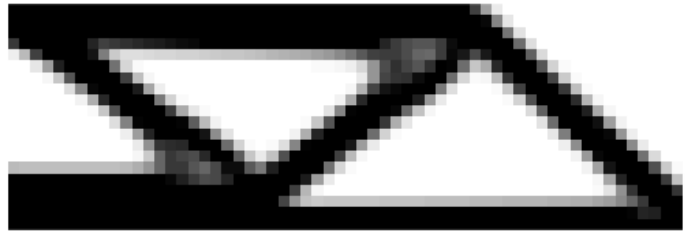
$p = 1.0$

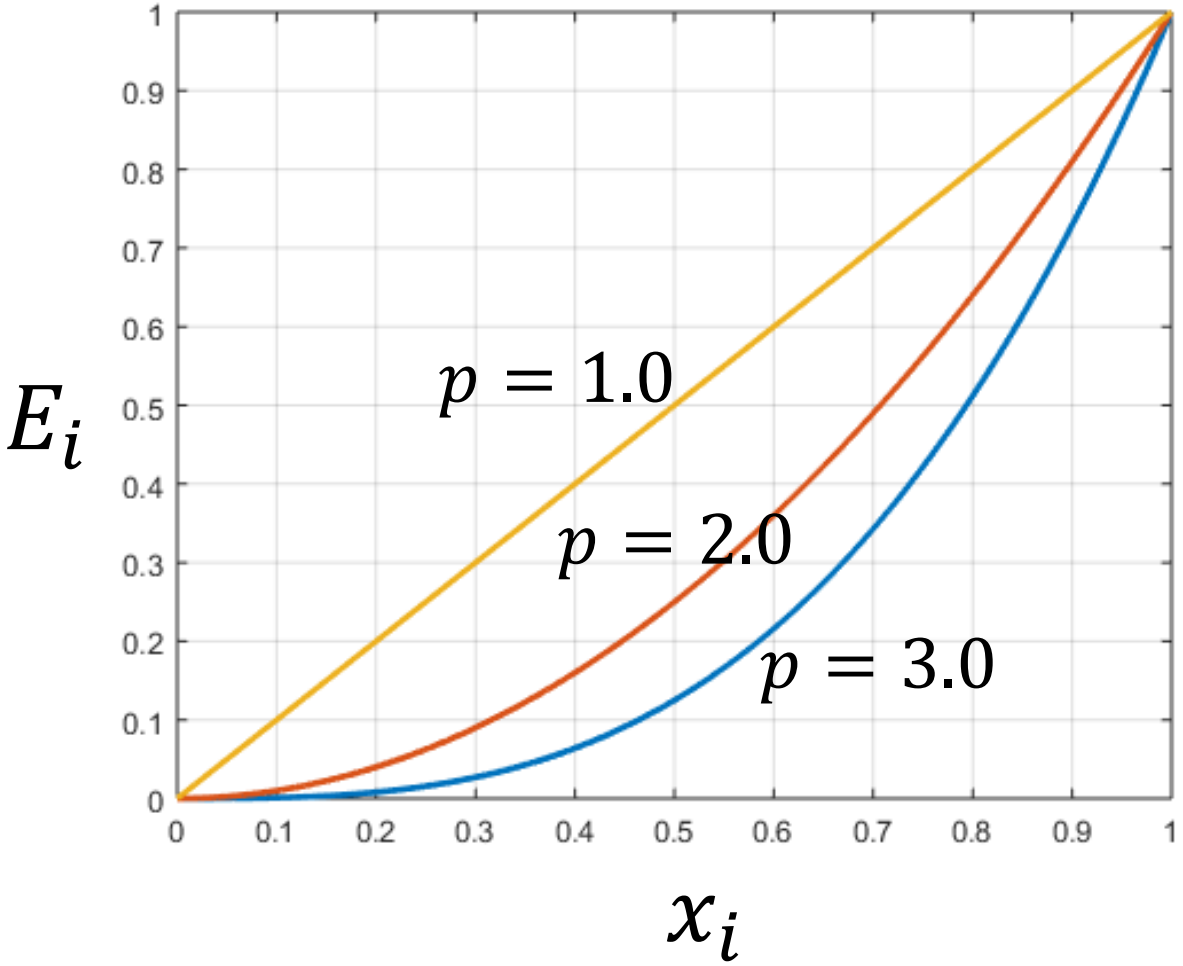


$p = 2.0$

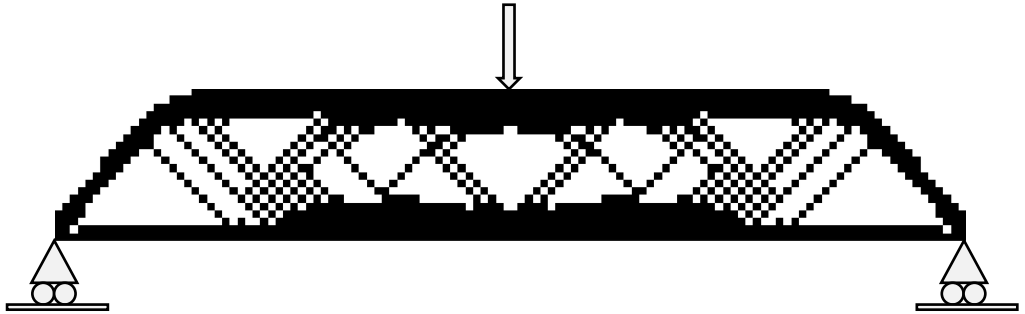


$p = 3.0$

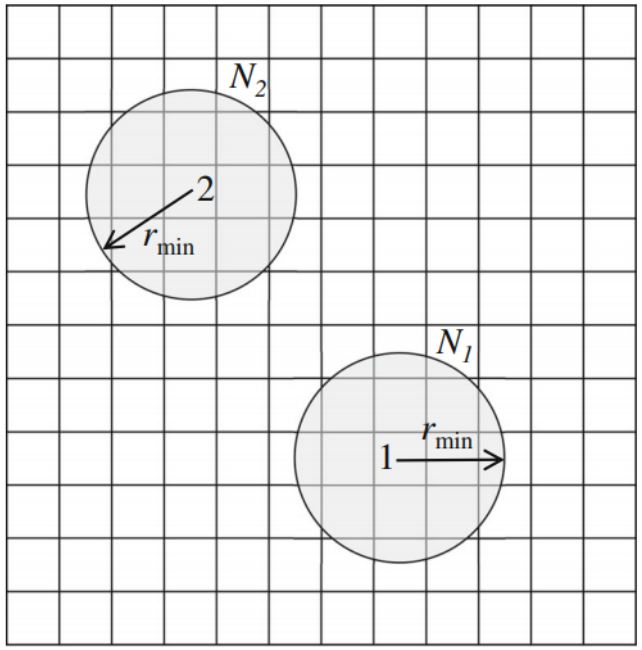




# Tabuleiro de Xadrez



- Filtro de sensibilidade:




$$\widehat{\frac{\partial c}{\partial x_e}} = \frac{1}{\max(\gamma, x_e) \sum_{i \in N_e} H_{ei}} \sum_{i \in N_e} H_{ei} x_i \frac{\partial c}{\partial x_i}$$

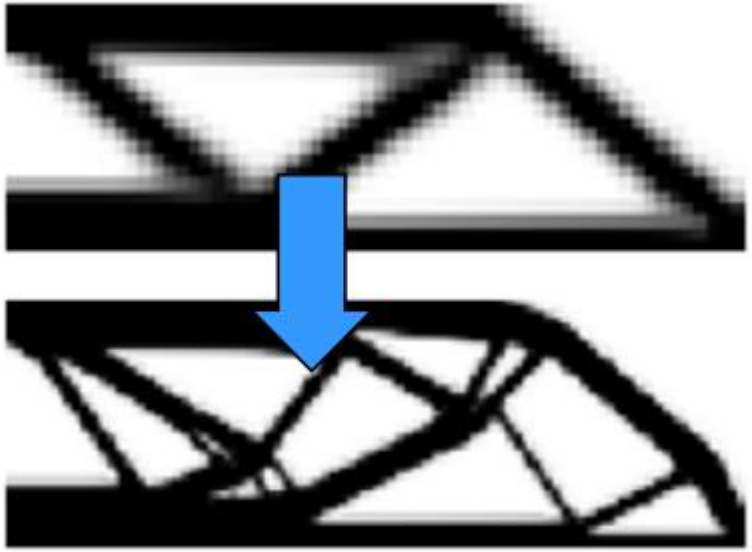
$$H_{ei} = \max(0, r_{\min} - \Delta(e, i))$$

fea.H

# Problemas numéricos

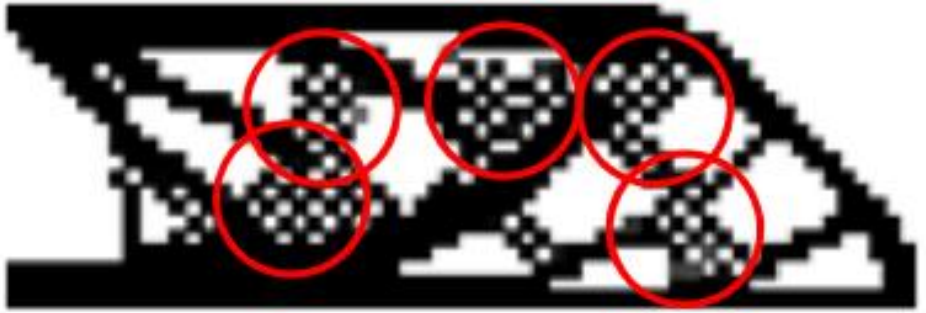
- a. dependência da malha e b. rigidez numérica artificial

a.



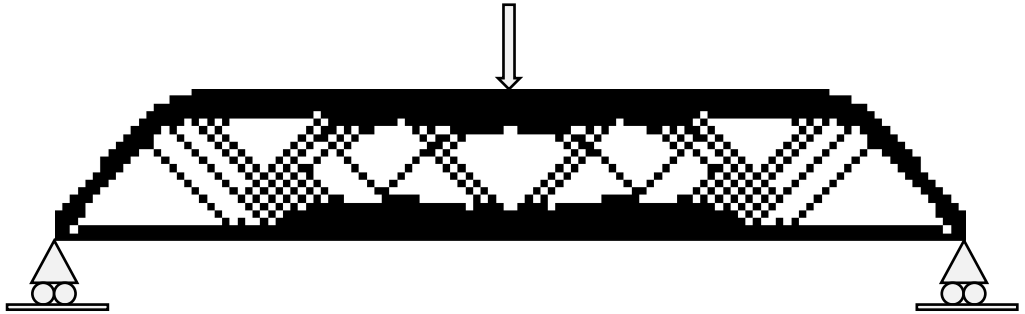
Refinamento da malha

b.

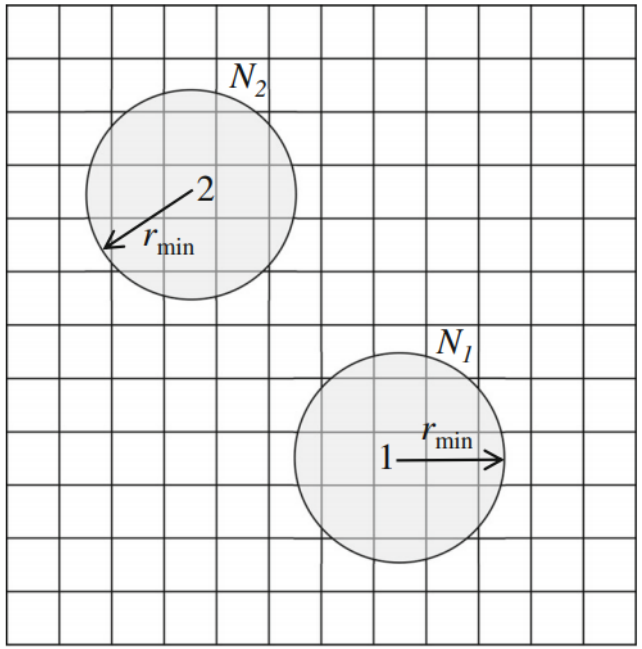


“checkerboarding”

# Tabuleiro de Xadrez



- Filtro de densidade:



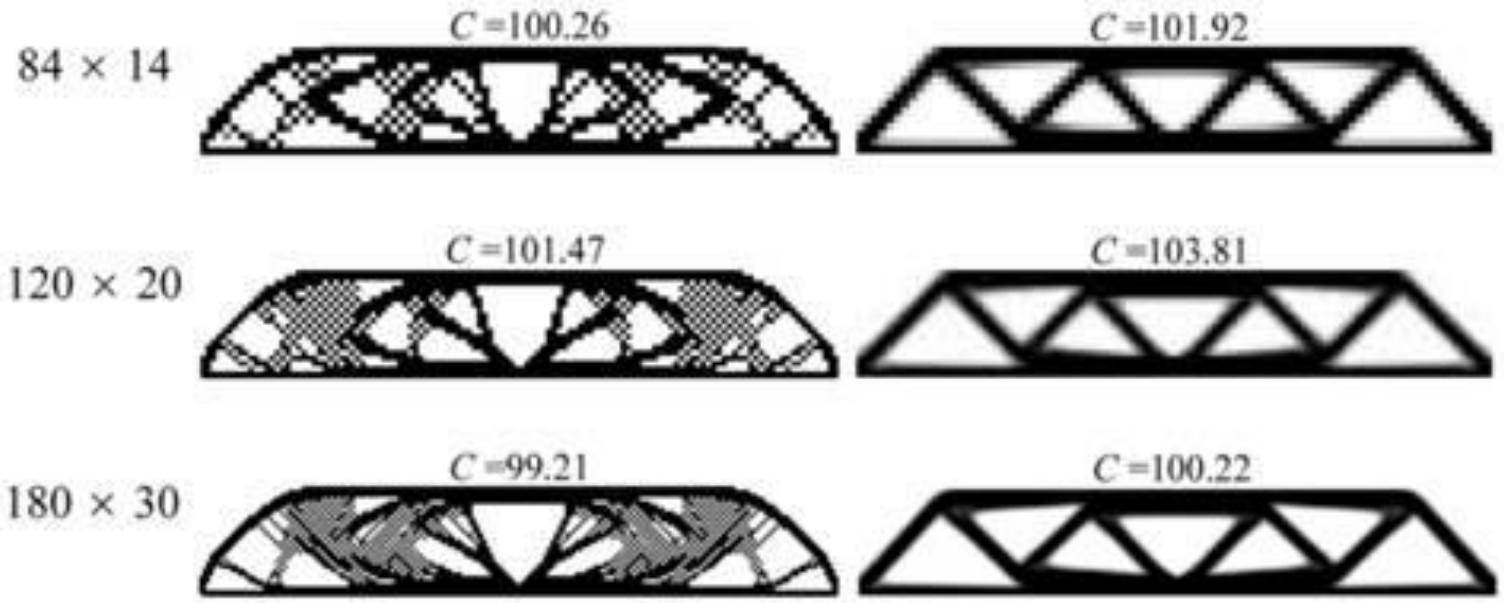

$$\tilde{x}_e = \frac{1}{\sum_{i \in N_e} H_{ei}} \sum_{i \in N_e} H_{ei} x_i$$

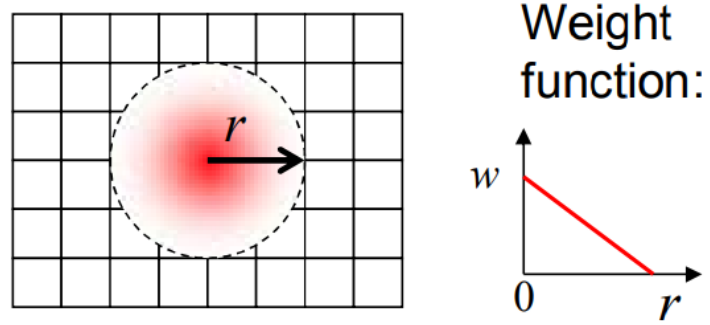
$$\frac{\partial \psi}{\partial x_j} = \sum_{e \in N_j} \frac{\partial \psi}{\partial \tilde{x}_e} \frac{\partial \tilde{x}_e}{\partial x_j} = \sum_{e \in N_j} \frac{1}{\sum_{i \in N_e} H_{ei}} H_{je} \frac{\partial \psi}{\partial \tilde{x}_e}$$

Ver código SIMP 88 linhas (lá é matriz!!!).

# Problemas numéricos

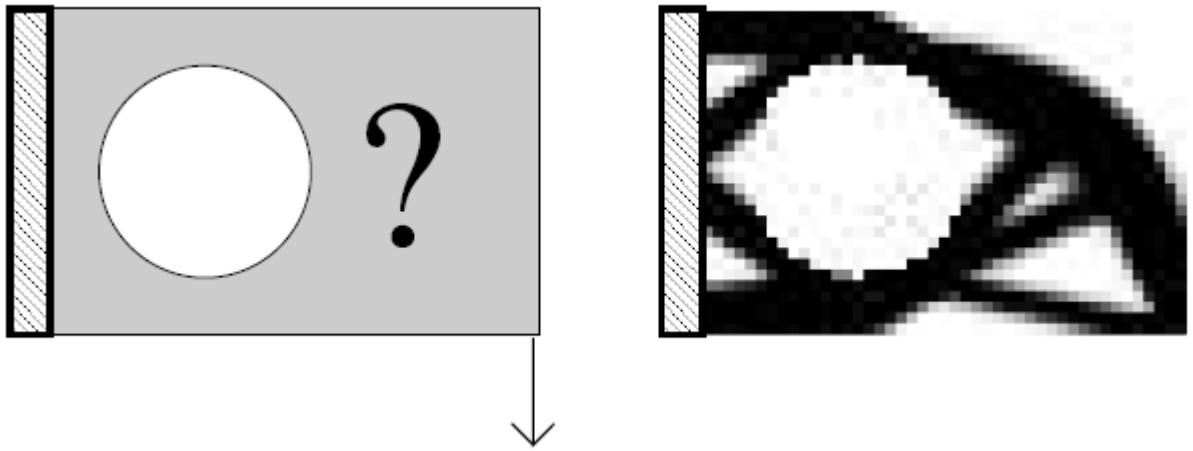
- Solução: filtros numéricos.



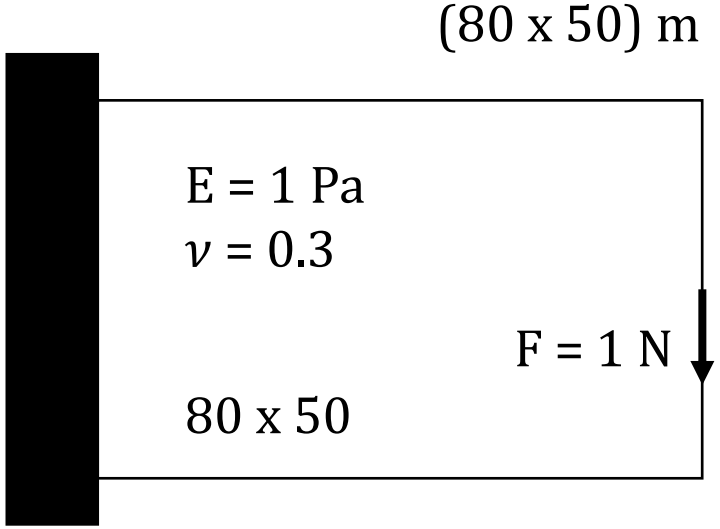

Média ponderada

# Domínio não-otimizável (non-design domain)




Seguir as instruções do artigo do 99 linhas e adaptar o código para considerar domínio passivo (não-otimizável) – do it at home for fun!

# Exercício 2




Estado plano de tensões

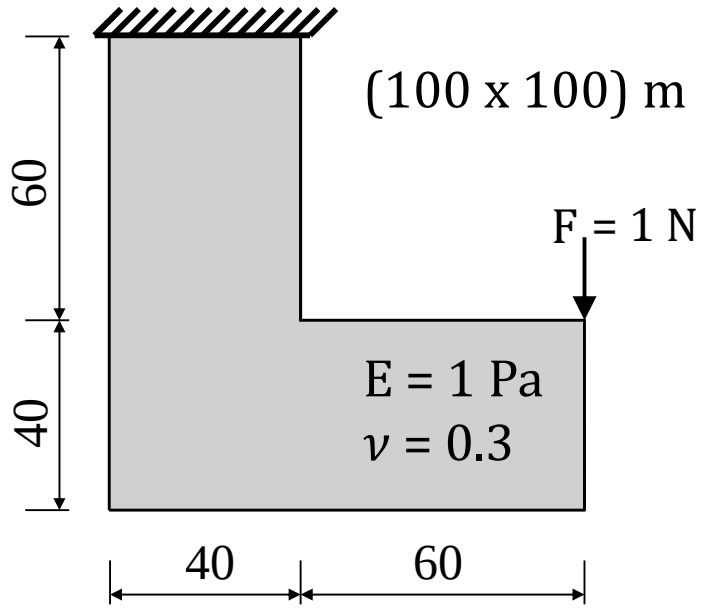
```
% Optimization parameters
rmin = 3;
penalty = 3;
final_volume_fraction = 0.5;
```

$$\begin{aligned} \text{minimizar} \quad & C(x_i) = \mathbf{F}^T\mathbf{U} \\ \text{tal que} \quad & V(x_i) - \bar{V} \leq 0 \\ & 0 \leq x_i \leq 1 \end{aligned}$$

Roteiro:

1. Atualizar a função do filtro no código com a rotina fornecida (Atenção! Esse filtro é implementado em função da unidade de comprimento, e.g., rmin = 3 m).
2. Extrair (copiar e colar) a função OC do código SIMP 99 linhas e salvar em seu diretório.
3. Realizar o loop de otimização utilizando o seu código.
   - Resolver equação de equilíbrio.
   - Calcular a sensibilidade da $C(x_i)$. Filtrar.
   - Chamar o otimizador OC (atualizar as variáveis).
   - Avaliar a convergência ($x_i^k - x_i^{k-1}$).
   - Plotar as variáveis de projeto.

# Exercício 3




Estado plano de tensões

```
% Optimization parameters
rmin = 3;
penalty = 3;
final_volume_fraction = 0.4;
```

$$\begin{aligned} \text{minimizar} \quad & C(x_i) = \mathbf{F}^T\mathbf{U} \\ \text{tal que} \quad & V(x_i) - \bar{V} \leq 0 \\ & 0 \leq x_i \leq 1 \end{aligned}$$

Ponto de partida, código do slide anterior.
Roteiro:

1. Identificar a região não otimizável (passiva) antes do loop de otimização e zerar as variáveis de projeto dessa região para formar a viga em L dentro do domínio quadrado 100 x 100.
2. Antes de chamar o otimizador OC, zerar as sensibilidades da região passiva.

Dica para o futuro: um engano comum é considerar a restrição de volume para o domínio inteiro 100 x 100, enquanto a estrutura cheia na verdade ocupa um espaço menor inicial.

# Method of Moving Asymptotes (MMA)

Throughout this paper, optimization problems on the following form are considered, where the variables are $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, $y = (y_i, \ldots, y_m)^T \in \mathbb{R}^m$ and $z \in \mathbb{R}$.

$$
\begin{aligned}
\text{minimize} \quad & f_0(x) + z + \sum_{i=1}^{m} (c_i y_i + \tfrac{1}{2} d_i y_i^2) \\
\text{subject to} \quad & f_i(x) - a_i z - y_i \leq 0, && i = 1, \ldots, m \\
& x_j^{min} \leq x_j \leq x_j^{max}, && j = 1, \ldots, n \\
& y_i \geq 0, && i = 1, \ldots, m \\
& z \geq 0,
\end{aligned}
\tag{1.1}
$$

Here, $f_0, f_1, \ldots, f_m$ are given, continuously differentiable, real-valued functions.

$x_j^{min}$ and $x_j^{max}$ are given real numbers such that, for each $j$, $x_j^{min} < x_j^{max}$.

$a_i$, $c_i$ and $d_i$ are given non-negative real numbers such that, for each $i$, $c_i + d_i > 0$.

Svanberg (1998) – Lecture notes

http://www.ingveh.ulg.ac.be/uploads/education/meca-0027-1/MMA_DCAMM_1998.pdf

# Method of Moving Asymptotes (MMA)

Assume that one wants to solve a problem on the following "standard" form for non-linear programming.

$$\begin{aligned} &\text{minimize} && f_0(x) && \\ &\text{subject to} && f_i(x) \le 0, && i = 1, \ldots, m \\ &&& x_j^{min} \le x_j \le x_j^{max}, && j = 1, \ldots, n \end{aligned} \tag{2.1}$$

The parameters $a_i$, $c_i$ and $d_i$ should then be chosen as follows in problem (1.1).

For each $i$, let $a_i = 0$. Then $z = 0$ in any optimal solution of (1.1). Further, for each $i$, let $d_i = 0$ and $c_i =$ "a large number", so that the variables $y_i$ become "expensive". Then typically $y_i = 0$ in any optimal solution of (1.1), and the corresponding $x$ is an optimal solution of (2.1).

Svanberg (1998) – Lecture notes

http://www.ingveh.ulg.ac.be/uploads/education/meca-0027-1/MMA_DCAMM_1998.pdf

# Method of Moving Asymptotes (MMA)

```
% MMA parameters
m     = 1;                              % The number of general constraints.
n     = length(fea.mesh.incidence);     % The number of design variables x_j.
xold1 = densities;                      % xval, one iteration ago (provided that iter>1).
xold2 = densities;                      % xval, two iterations ago (provided that iter>2).
xmin  = rho_min*ones(n,1);              % Column vector with the lower bounds for the variables x_j.
xmax  = ones(n,1);                      % Column vector with the upper bounds for the variables x_j.
low   = xmin;                           % Column vector with the lower asymptotes from the previous iteration.
upp   = xmax;                           % Column vector with the upper asymptotes from the previous iteration.
a0    = 1;                              % The constants a_0 in the term a_0*z.
a_i   = zeros(m,1);                     % Column vector with the constants a_i in the terms a_i*z.
cMMA  = 1000*ones(m,1);                 % Column vector with the constants c_i in the terms c_i*y_i.
d_i   = zeros(m,1);                     % Column vector with the constants d_i in the terms 0.5*d_i*(y_i)^2.
```

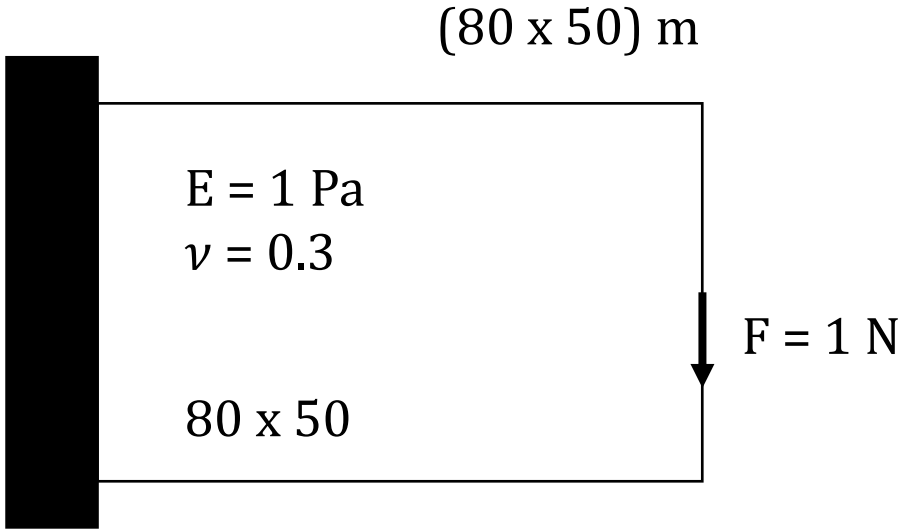
```
% Calling MMA subfunction
[densities_new, ~, ~, ~, ~, ~, ~, ~, ~, low,upp] = ...
mmasub(m, n, loop, densities, xmin, xmax, xold1, xold2, ...
objective,objective_sensitivities,0*objective_sensitivities,...
constraints,constraints_sensitivities',0*constraints_sensitivities',...
low,upp,a0,a_i,cMMA,d_i);

% Updating densities
xold2     = xold1;
xold1     = densities;
densities = densities_new;
```

```
% Calling MMA subfunction
[densities_new, ~, ~, ~, ~, ~, ~, ~, ~, low,upp] = ...
mma_movelim(m, n, loop, densities, xmin, xmax, xold1, xold2, ...
objective,objective_sensitivities,0*objective_sensitivities,...
constraints,constraints_sensitivities',0*constraints_sensitivities',...
low,upp,a0,a_i,cMMA,d_i,move_limit);

% Updating densities
xold2     = xold1;
xold1     = densities;
densities = densities_new;
```

# Exercício 4




Estado plano de tensões

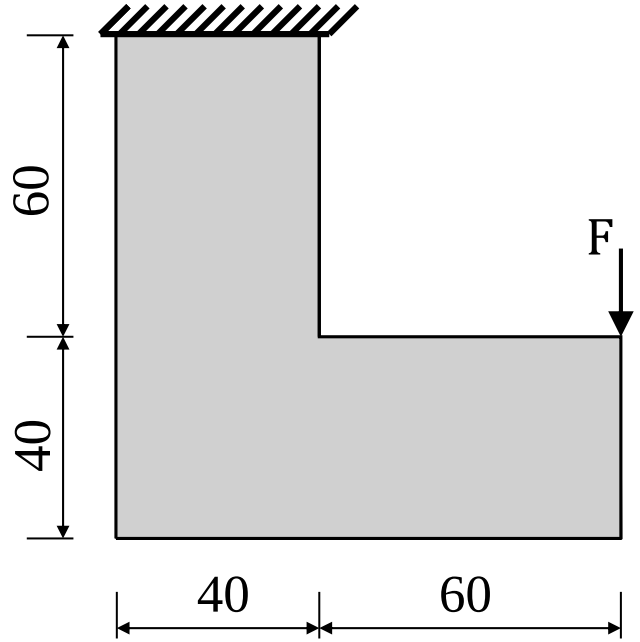
```
% Optimization parameters
rmin = 3;
penalty = 3;
final_volume_fraction = 0.5;
```

$$\begin{aligned} \text{minimizar} \quad & C(x_\mathrm{i}) = \mathbf{F}^T\mathbf{U} \\ \text{tal que} \quad & V(x_\mathrm{i}) - \bar{V} \leq 0 \\ & 0 < x_\mathrm{min} \leq x_\mathrm{i} \leq 1 \end{aligned}$$

Roteiro:

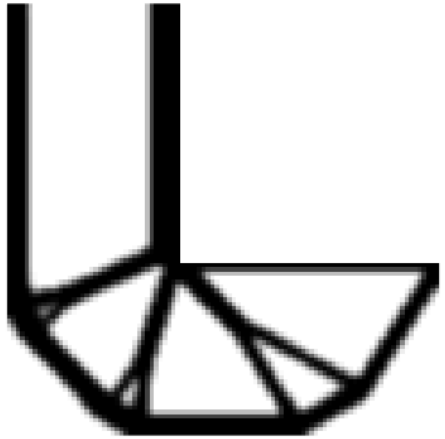
1. Inserir o cálculo da sensibilidade da restrição de volume.
2. Usar MMA no lugar do OC.
   - Preparar parâmetros do MMA antes do loop de otimização.
   - Chamar o MMA como solver.
   - Atualizar o histórico de variáveis de projeto.
3. Utilizar filtro de densidade.

# Problema da tensão mecânica

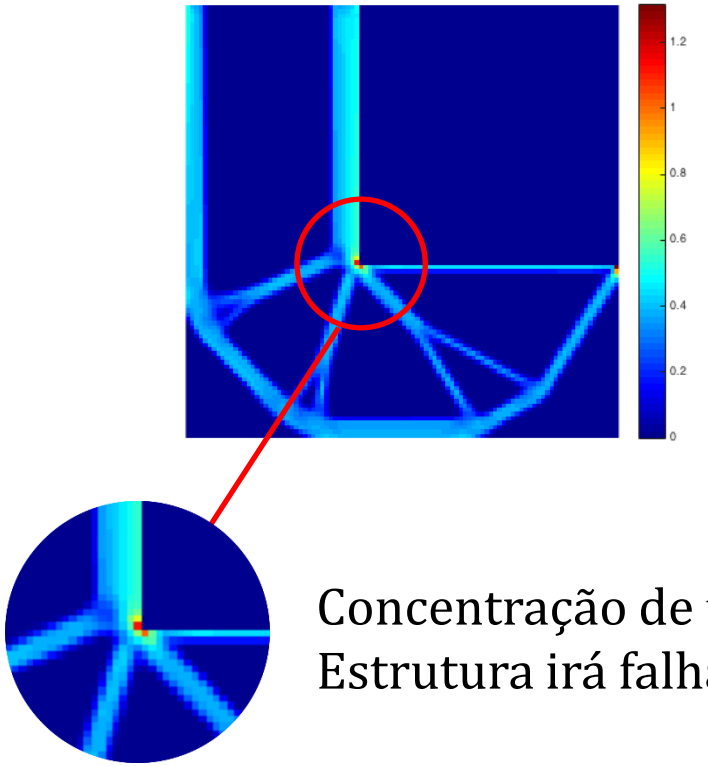



Máxima rigidez;
$V \leq 40\%$

Topologia otimizada



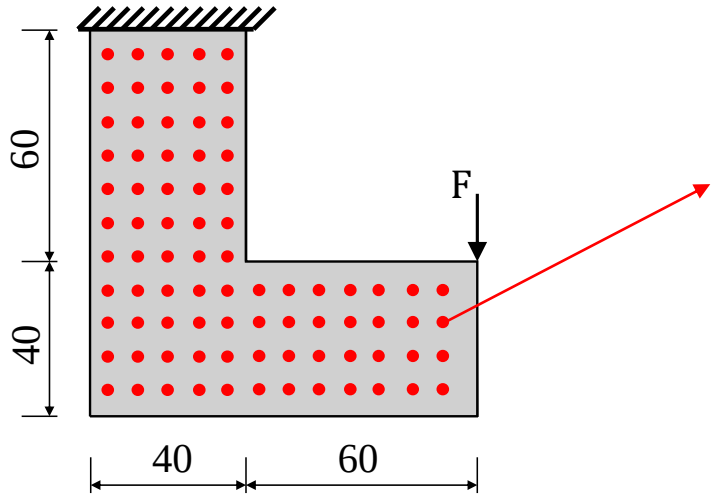
Tensão de von Mises




Concentração de tensão.
Estrutura irá falhar!

# Problema da tensão mecânica

- Objetivo muito frequente em projeto estrutural.
- Evitar o re-projeto devido a falha mecânica.
- Projetar estruturas leves mas evitando concentração de tensões.
- Controlar o limite de tensão em todos os pontos do domínio:

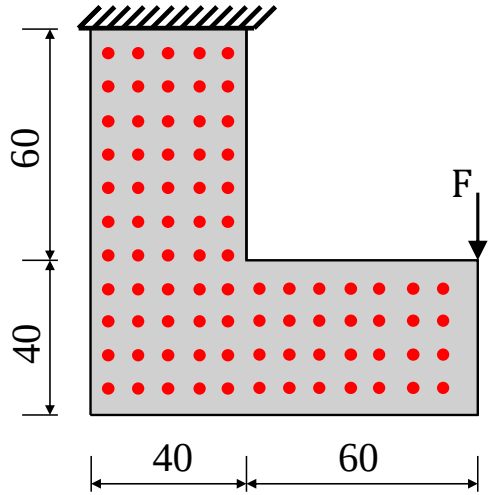



Tantas restrições quantos pontos de controle tiver.
Muitas restrições e difícil de se resolver!

$$\sigma_i \leq \bar{\sigma}$$

# Problema da tensão mecânica

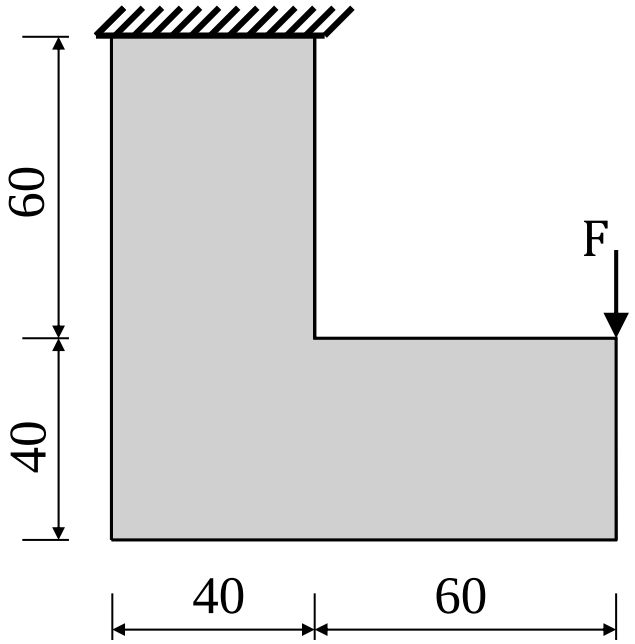
- Medida de tensão global: a função p-norm.




$$\sigma_{PN} = \left( \sum_{e=1}^{N} v_e \sigma_e^P \right)^{1/P} \leq \bar{\sigma}_{PN}$$

- Agrega todos os pontos de controle em uma só função.
- A função se aproxima da tensão máxima quando $P$ aumenta.

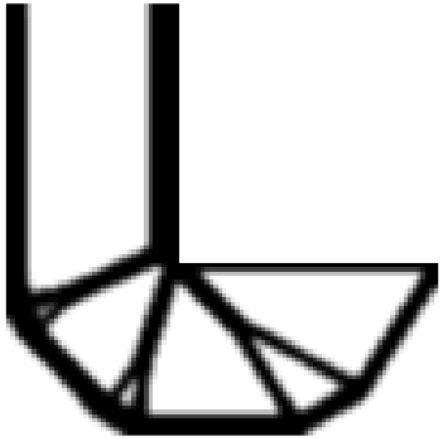
# Problema da tensão

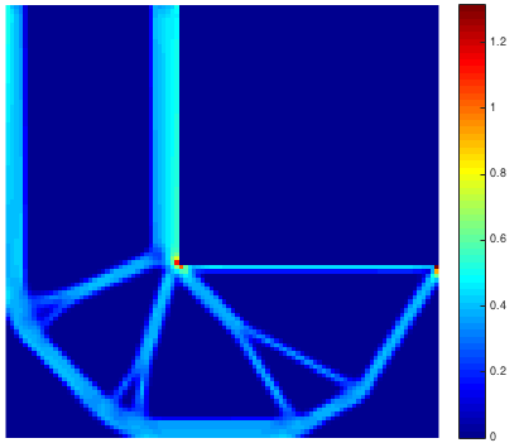



Solução

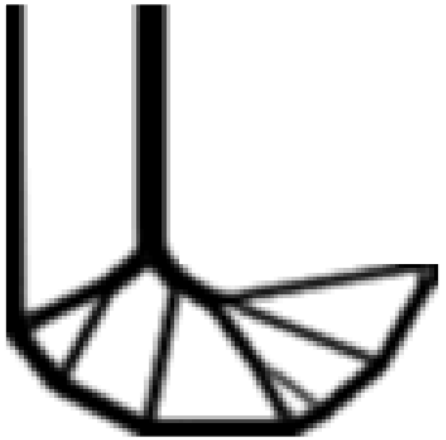
tensão de von Mises

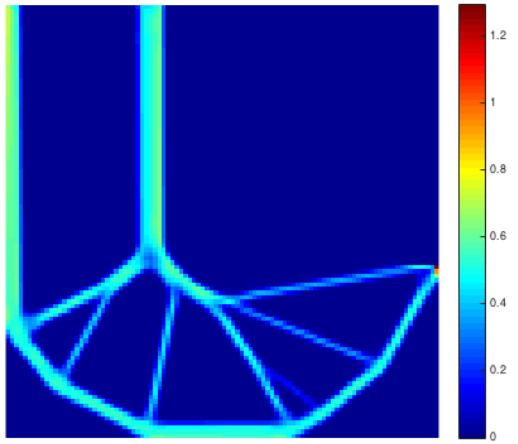
Máxima rigidez



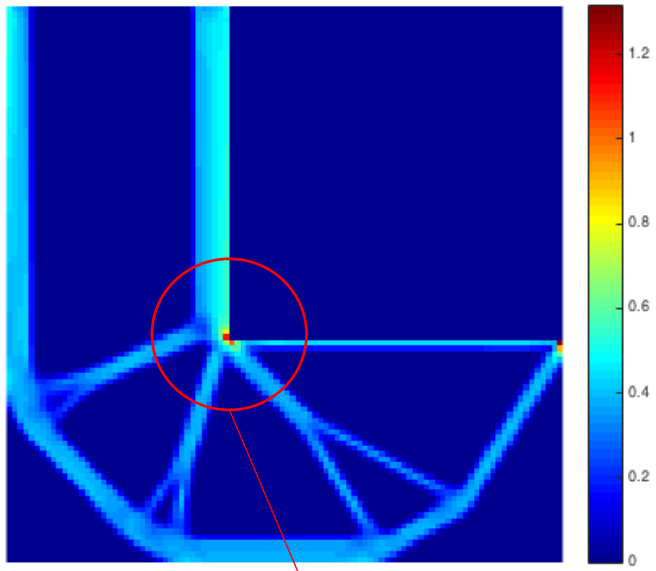



Mínima tensão p-norm

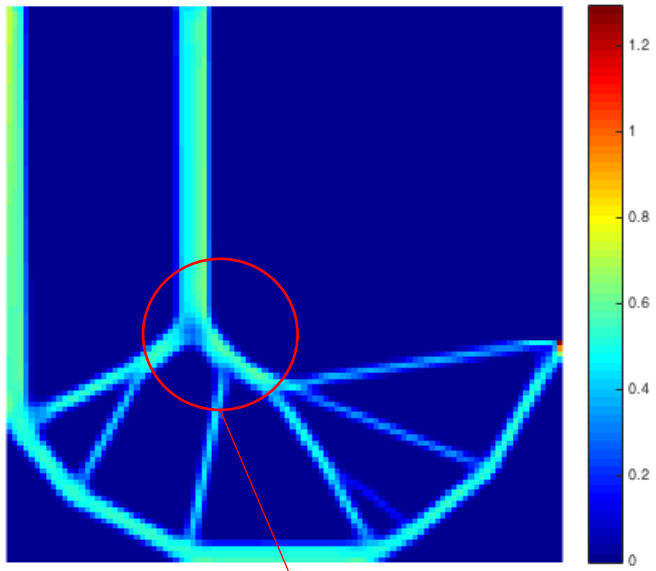




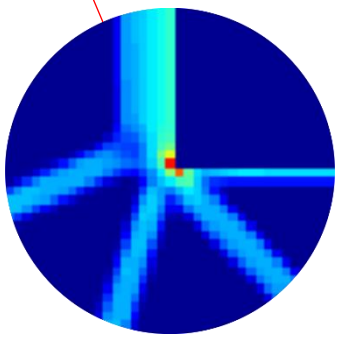
Solução para *compliance*

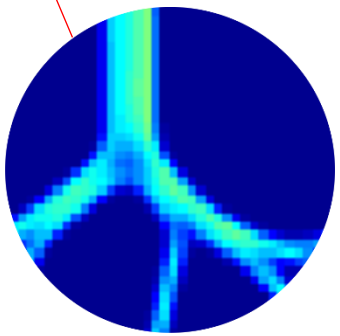



Solução para tensão




concentração de tensões





# Minimização da tensão mecânica via função p-norm

$$\text{minimizar} \quad \sigma_{PN} = \left( \sum_{i=1}^{N} V_i \sigma_i^P \right)^{1/P}$$

$$\text{tal que} \quad V(x_\mathrm{i}) - \bar{V} \leq 0$$

$$0 \leq x_\mathrm{i} \leq 1$$

# Exercício resolvido – Condução de calor

Com poucas modificações no algoritmo disponibilizado no Moodle (99 lines) podemos alterar a física do código de um problema de equilíbrio para um problema de elasticidade para a solução de um problema de condução de calor.

De forma análoga ao problema de "compliance estrutural", temos o "compliance de condução". Além disso, consideramos que os "vazios" são ocupados por um outro matéria com condutividade $k_2$ e a região com variável de projeto igual a 1, com condutividade $k_1$. De tal forma que podemos escrever nosso problema de otimização:

$$\begin{aligned} \text{minimizar} \quad & C(x_\mathrm{i}) = \mathbf{U}^\mathrm{T}\mathbf{K}_{\boldsymbol{e}}(\mathbf{x})\mathbf{U} \\ \text{tal que} \quad & V(x_\mathrm{i}) - \bar{V} \leq 0 \\ & 0 < x_{\min} \leq x_\mathrm{i} \leq 1 \end{aligned}$$

$$\mathbf{K}_{\boldsymbol{e}} = \boldsymbol{x}^\mathrm{p}\mathbf{K}_1 + (1 - \boldsymbol{x}^\mathrm{p})\,\mathbf{K}_2$$

$$\mathbf{K}_{\boldsymbol{e}} = \mathbf{K}_2 + \boldsymbol{x}^\mathrm{p}(\mathbf{K}_1 - \mathbf{K}_2)$$

# Exercício resolvido – Condução de calor

As primeiras mudanças no código 99 lines são na extração das matrizes de resposta locais (Ue) já que agora temos um grau de liberdade a menos por nó, na função objetivo e na sua sensibilidade.

Linhas 20 a 22:

De:

```
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;
```

Para:

```
Ue = U([n1;n2;n2+1;n1+1],1);
c = c + (kappa2+(kappa1-kappa2)*x(ely,elx)^penal)*Ue'*KE*Ue;
dc(ely,elx) = -(kappa1-kappa2)*penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
```

# Exercício resolvido – Condução de calor

Observe que como só metade parte do material presente no domínio é dado em função de **x**, a sensibilidade só leva em conta parte da distribuição. De tal forma que é assumido que onde não há o material governado por $k_1$, há o segundo material com condutividade $k_2$.

A segunda mudança ocorre na definição das matrizes globais.

Linhas 68-69

De:

```
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);
```

Para:

```
K = sparse((nelx+1)*(nely+1), (nelx+1)*(nely+1));
F = sparse((nely+1)*(nelx+1),1); U = sparse((nely+1)*(nelx+1),1);
```

# Exercício resolvido – Condução de calor

Em seguida, mudamos a contagem dos graus de liberdade para a montagem da matriz global já que os reduzimos pela metade.

Linhas 74-75

De:

```
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;
```

Para:

```
edof = [n1; n2; n2+1; n1+1];
K(edof,edof) = K(edof,edof) + ((kappa1-kappa2)*x(ely,elx)^penal)*KE;
```

Note a facilidade em construir a matriz global. Isso de deve ao fato de que estamos usando uma malha estruturada.

# Exercício resolvido – Condução de calor

Agora mudamos as condições de contorno

Linhas 78-81

De:

```
% 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);
```

Para:

```
% DEFINE LOADS AND SUPPORTS (SQUARE PLATE WITH HEAT SINK)
F(:,1) = 0.01;
fixeddofs   = [nely/2+1-(nely/20):2:nely/2+1+(nely/20)];
alldofs     = [1:(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
```

Aqui existe um problema quanto a forma que o termo fonte é imposto. Você saberia qual é esse problema?

# Exercício resolvido – Condução de calor

E por fim, mudamos a definição de matriz de rigidez:

Linhas 88-99

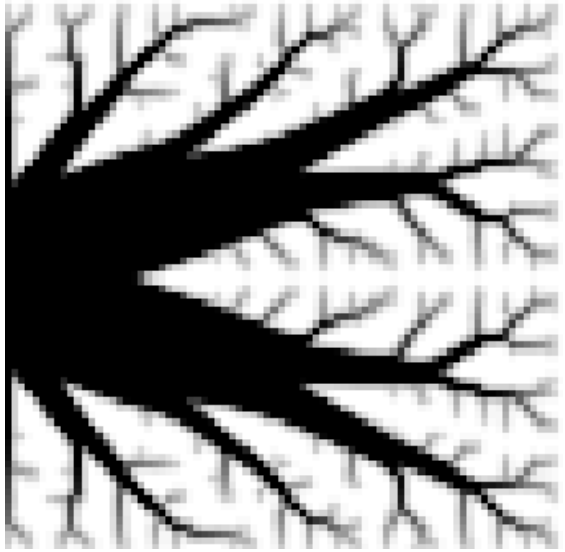
De:

```
E = 1.;
nu = 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)];
```

# Exercício resolvido – Condução de calor

Para:

```
KE = [2/3 -1/6 -1/3 -1/6
      -1/6 2/3 -1/6 -1/3
      -1/3 -1/6 2/3 -1/6
      -1/6 -1/3 -1/6 2/3];
```



Ao rodar com kappa1 = 1 e kappa2 = 0.001, o seguinte plot deve ser visualizado:

Digite no terminal:

```
toph(80,80,0.4,3.0,1.2, 0.1, 0.0001)
```

Teste outros valores de kappa1 e kappa2, cuidado que eles são definidos dentro de duas funções (linhas 6 e 7 e nas linhas 73 e 74)