UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
MODELAGEM TERMODINAMICAMENTE CONSISTENTE DE MATERIAIS

# Atividade 1
# Propagação de uma trinca em uma barra 1D de seção constante

Renan Miranda Portela

3 de dezembro de 2018

## 1 OBJETIVO

O objetivo desta atividade é verificar a propagação do dano em uma barra de seção uniforme tracionada através das equações de movimento e dano.

## 2 METODOLOGIA

A equação 2.1 é referente a equação do movimento.

$$\rho_0 \ddot{u}_0 = div_p[F(1-\varphi)^2 KE] \tag{2.1}$$

No caso de um movimento quasi-estático, consideramos a aceleração como sendo nula como mostra a equação 2.2.

$$0 = div_p[F(1-\varphi)^2 KE] \tag{2.2}$$

Fazendo a aproximação para pequenas deformações.

$$[F][E] \simeq [E] \tag{2.3}$$

Substituindo a equação 2.3 na equação 2.2

$$0 = div_p[(1-\varphi)^2 KE]$$

Aplicando resíduos ponderados

$$0 = \int_\Omega div_p[(1-\varphi)^2 KE]w(p)dp$$

Aplicando integração por partes

$$[(1-\varphi)^2 KE]w(p)|_{p=0}^{p=L} - \int_\Omega [(1-\varphi)^2 KE]\frac{dw(p)}{dp}dp = 0$$

Consideramos o termo à esquerda da equação 2.4 como sendo o vetor força $\hat{f}$.

$$[(1-\varphi)^2 K\frac{\partial u_n}{\partial p}]w(p)|_{p=0}^{p=L} - \int_\Omega [(1-\varphi)^2 K\frac{\partial u_n}{\partial p}]\frac{dw(p)}{dp}dp = 0 \tag{2.4}$$

$$\hat{f} = [(1-\varphi_{n-1})^2 K\frac{\partial u_n}{\partial p}]w(p)|_{p=0}^{p=L}$$

Forçamos o vetor força como sendo igual a força no último nó como mostrado no vetor 2.5 de tamanho $nx1$ sendo $n$ o número de nós.

$$\hat{f} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -1000 \end{bmatrix}_{nx1} \tag{2.5}$$

$$\int_\Omega [(1-\varphi_{n-1})^2 K\frac{\partial u_n}{\partial p}]\frac{dw(p)}{dp}dp = \hat{f}$$

$$\int_\Omega [(1-[N][\hat{\varphi}_{n-1}])^2 K[B][\hat{u}_n]].[B][\hat{w}]dp = \hat{f}$$

$$\int_\Omega [(1-[N][\hat{\varphi}_{n-1}])^2 K[B]^T[B]dp][\hat{u}_n] = \hat{f}$$

$$[K_g] = A_{i=1}^{nel}[K_e]$$

$$[K_g][\hat{u}] = \hat{f}$$

$$[K_e] = KA\int_L (1-[N][\hat{\varphi}_{n-1}])^2[B]^T[B]dp$$

$$[K_e] = \frac{KA}{L^e}\int_{-1}^{1}(1-[N]\hat{\varphi})^2[B]^T[B]det(J)d\xi \tag{2.6}$$

Para um elemento de dois nós:

$$[N] = [\frac{1-\xi}{2}, \frac{1+\xi}{2}]$$

$$[B] = [\frac{-1}{2}, \frac{1}{2}]$$

$$det(J) = \frac{x_{n+1} - x_n}{2}$$

Apresentamos agora a equação do dano (2.7).

$$\lambda_1 \dot{\varphi} = div_p[g_c \gamma \bigtriangledown_p \varphi_0] + (1-\varphi_0)K|E|^2 - \frac{g_c}{\gamma}\varphi_0 \tag{2.7}$$

Discretizando o termo do dano em relação ao tempo encontramos equação (2.8).

$$\dot{\varphi} = \frac{\varphi_n - \varphi_{n-1}}{\Delta t} \tag{2.8}$$

Substituindo (2.8) em (2.7), encontramos:

$$\lambda_1 \frac{\varphi_n - \varphi_{n-1}}{\Delta t} = div_p[g_c \gamma \bigtriangledown_p \varphi_n] + (1-\varphi_{n-1})K|E|^2 - \frac{g_c}{\gamma}\varphi_n$$

$$\varphi_n - \varphi_{n-1} = \frac{\Delta t}{\lambda_1} div_p[g_c \gamma \bigtriangledown_p \varphi_n] + \frac{\Delta t K|E|^2}{\lambda_1}(1-\varphi_{n-1}) - \frac{g_c \Delta t}{\gamma \lambda_1}\varphi_n$$

$$\varphi_n + \frac{g_c \Delta t}{\gamma \lambda_1}\varphi_n - \frac{g_c \gamma \Delta t}{\lambda_1} \triangle_p \varphi_n = \varphi_{n-1} + \frac{\Delta t K|E|^2}{\lambda_1}(1-\varphi_{n-1})$$

Aplicando o método de resíduos ponderados:

$$\begin{aligned} \int_\Omega \varphi_n w dp + \int_\Omega \frac{g_c \Delta t}{\gamma \lambda_1} \varphi_n w dp - \int_\Omega \frac{g_c \gamma \Delta t}{\lambda_1} \triangle_p \varphi_n w dp = \\ \int_\Omega \varphi_{n-1} w dp + \int_\Omega \frac{\Delta t K|E|^2}{\lambda_1}(1-\varphi_{n-1}) w dp \end{aligned}$$

$$\begin{aligned} \int_\Omega ([N][\hat{\varphi}_n]).([N][\hat{w}])dp + \frac{g_c \Delta t}{\gamma \lambda_1} \int_\Omega ([N][\hat{\varphi}_n]).([N][\hat{w}])dp - \frac{g_c \gamma \Delta t}{\lambda_1} \int_\Omega \bigtriangledown_p \varphi_n . \bigtriangledown_p w dp = \\ \int_\Omega ([N][\hat{\varphi}_{n-1}]).([N][\hat{w}])dp + \int_\Omega \frac{\Delta t K|E|^2}{\lambda_1}(1-[N][\hat{\varphi}_{n-1}]) w dp \end{aligned}$$

$$\begin{aligned} \int_\Omega ([N]^T[N][\hat{\varphi}_n^k])dp + \frac{g_c \Delta t}{\gamma \lambda_1} \int_\Omega ([N]^T[N][\hat{\varphi}_n^k])dp - \frac{g_c \gamma \Delta t}{\lambda_1} \int_\Omega ([B]^T[B][\hat{\varphi}_n^k])dp = \\ \int_\Omega ([N]^T[N][\hat{\varphi}_{n-1}])dp + \int_\Omega \frac{\Delta t K(\frac{\partial u_n}{\partial p})^2}{\lambda_1}(1-[N][\hat{\varphi}_{n-1}])[N][\hat{w}(p)]dp \end{aligned}$$

$$\int_\Omega ([N]^T[N][\hat{\varphi}_n^k])dp + \frac{g_c\Delta t}{\gamma\lambda_1}\int_\Omega ([N]^T[N][\hat{\varphi}_n^k])dp - \frac{g_c\gamma\Delta t}{\lambda_1}\int_\Omega ([B]^T[B][\hat{\varphi}_n^k])dp = \int_\Omega ([N]^T[N][\hat{\varphi}_{n-1}])dp + \int_\Omega \frac{\Delta tK}{\lambda_1}(1-[N][\hat{\varphi}_{n-1}])[[B][\hat{u}_n(p)]]^2[N][\hat{w}(p)]dp$$

Fazendo a formulação do problema unidimensional:

$$A\int_L ([N]^T[N][\hat{\varphi}_n^k])dp + \frac{g_cA\Delta t}{\gamma\lambda_1}\int_L ([N]^T[N][\hat{\varphi}_n^k])dp - \frac{g_cA\gamma\Delta t}{\lambda_1}\int_L ([B]^T[B][\hat{\varphi}_n^k])dp = A\int_L ([N]^T[N][\hat{\varphi}_{n-1}])dp + A\int_L \frac{\Delta tK}{\lambda_1}(1-[N][\hat{\varphi}_{n-1}])[[B][\hat{u}_n(p)]]^2[N][\hat{w}(p)]dp$$

$$\frac{A}{L^e}\int_{-1}^{1}([N]^T[N][\hat{\varphi}_n^k])det(J)d\xi + \frac{g_cA\Delta t}{\gamma\lambda_1L^e}\int_{-1}^{1}([N]^T[N][\hat{\varphi}_n^k])det(J)d\xi - \frac{g_cA\gamma\Delta t}{\lambda_1L^e}\int_{-1}^{1}([B]^T[B][\hat{\varphi}_n^k])det(J)d\xi = \frac{A}{L^e}\int_{-1}^{1}([N]^T[N][\hat{\varphi}_{n-1}])det(J)d\xi + \frac{A\Delta tK}{L^e\lambda_1}\int_{-1}^{1}[N]^T(1-[N][\hat{\varphi}_{n-1}])[[B][\hat{u}_n]]^2det(J)d\xi \tag{2.9}$$

# 3 Resultados

A imagem 3.1 mostra o progresso do dano ao longo das iterações. Verifica-se que o dano precisa de dois terços do tempo para alcançar 10% do dano prescrito, a partir desse ponto o dano se propaga mais rapidamente.

As imagens 3.2 mostram o progresso da trinca em uma barra de comprimento $L = 1\ m$ com 1000 elementos. O valor da área foi considerada como 1 $m^2$, a constante $K$ corresponde ao módulo de *Young* e foi atribuído valor $200.10^9\ N/m^2$, $L^e$ é o comprimento do elemento dado pela divisão do comprimento da geometria pelo número de elementos, $g_c$ corresponde a energia de *Griffth* e vale 2700, $\lambda$ é a sensibilidade de propagação do dano e $\gamma$ a espessura do dano, valendo, aproximadamente o comprimento do elemento. O código foi baseado nas equações 2.6 e 2.9.

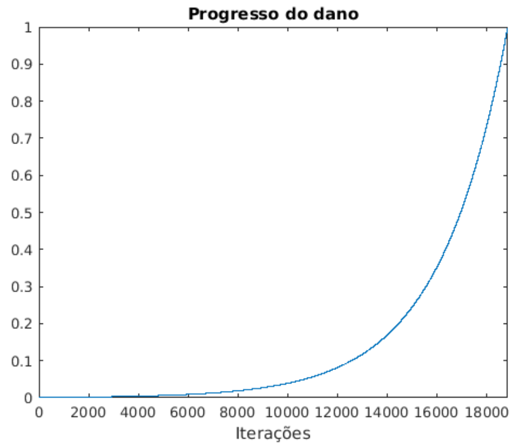
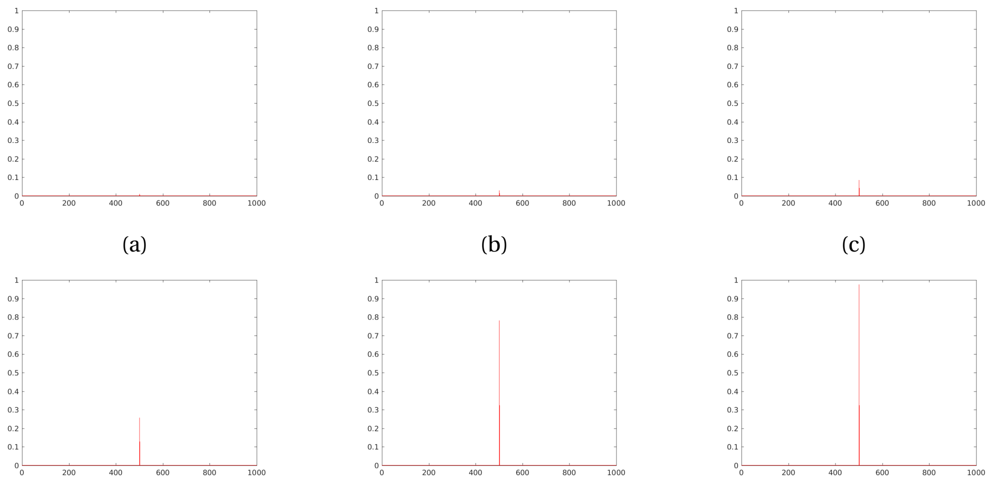



Figura 3.1: Progresso do dano ao longo das iterações



(a) (b) (c)

(d) (e) (f)

Figura 3.2: Evolução da trinca

# 4 Conclusão

No estudo de um caso *quasi*-estático de uma trinca se propagando em uma barra tracionada, e considerando o caso isotérmico, verificamos que a propagação da trinca se dá exponencialmente.

# ANEXO

```
% @university: UNIVERSIDADE ESTADUAL DE CAMPINAS
% @school: FACULDADE DE ENGENHARIA MECANICA
% @module: MODELAGEM DE MATERIAIS TERMODINAMICAMENTE CONSISTENTES - IM437 L
% @activity: ATIVIDADE 1
%
% @author: DIPL. -ENG RENAN MIRANDA PORTELA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clean memory and close windows

clear all
close all
clc

%% Coordinate matrix

nel = 100; % number of elements
coord = zeros(nel,2); % coordinate matrix pre-location
L = 1; % length in meters [m]
A = 1; % area in meters square [m2]

%% Equation parameters

delta_t = 1e-2;
gamma = L/nel;
g_c = 2700;
lambda = 1e-2;
L_e = L/nel;

for i = 1:nel+1
coord(i,1) = i; % number of nodes
coord(i,2) = (i-1)*L/nel; % x coordinates
end

%% incidence matrix

inci = zeros(nel,5);

for i = 1:nel
inci(i,1) = i; % number of elements
inci(i,2) = 1; % material
```

```
inci(i,3) = 1; % geometry
inci(i,4) = i; % node 1
inci(i,5) = i +1; % node 2
end

%% material matrix
           %E     rho  nu
material = [200e9 7000 0.3;  % steel [Pa] [kg/m2] []
           70e9 2700 0.27]; % aluminium

%% geometry matrix

nnos = size(coord,1); % number of nodes
if mod(nnos,2) ~= 0

    k = (nnos+1)/2;

else

    k = nnos/2;

end

phi_damage = zeros(nnos,1);
phi_damage(k,1) = 1e-3;

%% boundary conditions matrix
%   bc = [ node | degree of freedom | value ]
%
%   Grau de liberdade 1 --> x
%   Grau de liberdade 2 --> y
%   Grau de liberdade 3 --> z
%   Grau de liberdade 4 --> ox
%   Grau de liberdade 5 --> oy
%   Grau de liberdade 6 --> oz

bc=[1 1 0;
    1 2 0];

%% load matrix

Load=[nel+1 1 1000];

%% code
```

```
nbc = size(bc,1); % number of boundary conditions
nF = size(Load,1); % number of external loads

%% while loop

kk = 0; % figure number
conv = 0; % convergence condition
iter = 0; % number of iterations

max_value = zeros(100,1); % maximum value of damage vector

while conv == 0 % beginning of while loop

    iter = iter + 1;
    pause(1e-6)
    plot(phi_damage,'r') % re-plot of phi
    axis([0 nel 0 1])

    max_value(iter) = max(phi_damage);

    %% Assembling global stiffness matrix

    alldof = 1:nnos; % all degrees of freedom
    kg = zeros(nnos); % global stiffness matrix pre-location

    for i = 1 : nel

        node_1 = inci(i,4); % first node
        node_2 = inci(i,5); % second node

        x_1 = coord(node_1,2); % location  first node
        x_2 = coord(node_2,2); % location  second node

        E = material(inci(i,2),1); % young's module

        loc = [node_1, node_2];  % location vector

        ke = 0; % elemental stiffness matrix iniciation

        for j = 1 : 3 % gaussian quadrature integration

            if j == 1
```

```
            e = -sqrt(3/5);
            w = 5/9;

        elseif j == 2

            e = 0;
            w = 8/9;

        elseif j == 3

            e = sqrt(3/5);
            w = 5/9;

        end

        N = [(1-e)/2; (1+e)/2]; % base function
        B = 0.5*[-1 1]; % base function derivative

        phi = phi_damage(loc,1); % elemental damage
        det_J = (x_2 - x_1)/2; % Jacobian 2 nodes element

        ke = ke + E*A/L_e*((1-N'*phi)^2)*B'*B*det_J;
        % elemental stiffness matrix

    end

    kg(loc,loc) = kg(loc,loc) + ke; % global stiffness matrix

end

freedof = alldof; % free degrees of freedom

for k = 1 : size(bc,1) % free degrees of freedom
    freedof(bc(k,1)) = 0;
end

%% Assembling force vector

F = zeros(nnos,1);

for  i = 1 : size(Load,1)

    F(Load(i,1)) = Load(i,3);
```

```
end

u_n = zeros(size(F,1),1); % deformation vector pre−location

kg_aux = sparse(kg(logical(freedof),logical(freedof)));
% column & rows elimination
F_aux = sparse(F(logical(freedof),1)); % column & rows elimination

u_n(logical(freedof),1) = kg_aux\F_aux; % deformation vector

%% damage (N−1)

F_2 = zeros(nnos,1); % global force vector pre−location

for i = 1 : nel

    node_1 = inci(i,4); % first node
    node_2 = inci(i,5); % second node

    x_1 = coord(node_1,2); % location first node
    x_2 = coord(node_2,2); % location second node

    E = material(inci(i,2),1); % young's module

    loc = [node_1, node_2]; % location vector

    fe = 0; % force vector iniciation

    for j = 1 : 3 % gaussian quadrature integration

        if j == 1

            e = −sqrt(3/5);
            w = 5/9;

        elseif j == 2

            e = 0;
            w = 8/9;

        elseif j == 3

            e = sqrt(3/5);
            w = 5/9;
```

```
        end

        N = [(1-e)/2; (1+e)/2]; % base function
        B = 0.5*[-1 1]; % base function derivative

        phi = phi_damage(loc,1); % elemental damage
        u_ele = u_n(loc,1); % elemental deformation

        det_J = (x_2 - x_1)/2; % Jacobian 2 nodes element


        fe = fe + A/L_e*N*N'*phi*det_J
        + A*delta_t*E/(L_e*lambda)*N*(1-N'*phi)*det_J*(B*u_ele)^2;
        % elemental force vector

    end

    F_2(loc,1) = F_2(loc,1) + fe; % assembling global force vector

end

%% damage N

kg_2 = zeros(nnos);

for i = 1 : nel

    node_1 = inci(i,4); % first node
    node_2 = inci(i,5); % second node

    x_1 = coord(node_1,2); % location first node
    x_2 = coord(node_2,2); % location second node

    E = material(inci(i,2),1); % young's module

    loc = [node_1, node_2]; % location vector

    ke_2 = 0; % elemental stiffness matrix iniciation

    for j = 1 : 3 % gaussian quadrature integration

        if j == 1
```

```
            e = -sqrt(3/5);
            w = 5/9;

        elseif j == 2

            e = 0;
            w = 8/9;

        elseif j == 3

            e = sqrt(3/5);
            w = 5/9;

        end

        N = [(1-e)/2; (1+e)/2]; % base function
        B = 0.5*[-1 1]; % base function derivative

        det_J = (x_2 - x_1)/2; % Jacobian 2 nodes element

        ke_2 = ke_2 + A/L_e*(N*N'*det_J) +
        g_c*A*delta_t/(gamma*lambda*L_e)*(N*N'*det_J) -
        g_c*A*delta_t*gamma/(lambda*L_e)*(B'*B*det_J);

    end

    kg_2(loc,loc) = kg_2(loc,loc) + ke_2; % assembling global stiffness matrix

end

phi_n = zeros(size(F_2,1),1); % phi damage step

kg_2_aux = sparse(kg_2(logical(freedof),logical(freedof))); % column & rows
elimination
F_2_aux = sparse(F_2(logical(freedof),1)); % column & rows elimination

phi_n(logical(freedof),1) = kg_2_aux\F_2_aux; % displacement vector

phi_damage = phi_damage + phi_n; % new phi damage vector


if mod(iter, 3000) == 0 % saving figures
    kk = kk + 1;
    filename = sprintf('%d_%d', nel, kk);
```

```
        saveas(gcf,filename,'png')
    end

    if max(phi_damage) >= 1 % convergence condition
        break
    end

end

figure() % new figure
plot(max_value) % plot damage progress
axis([0 iter 0 1])
```