Trabalho2

June 30, 2020

1 Método dos Elementos Finitos - Trabalho 2

Universidade Federal Fluminense

Disciplina ministrada pelo Prof. Marco Ferro

marco.ferro@uol.com.br

Aluno Noé de Lima

noe_lima@id.uff.br

Este trabalho visa aplicar o MEF a uma estrutura de treliças.

Primeiro semestre de 2020

A célula a seguir configura o Jupyter-Notebook para exibir as equações matemáticas no formato do ambiente LaTeX e importa as bibliotecas necessárias.

[1]:
```
%display latex
from numpy import angle,array,delete,isnan,nan,zeros
from numpy.linalg import norm,solve
import json
from sage.misc.latex import MathJax,latex_extra_preamble,png
from sage.plot.line import Line
latex.add_to_preamble('\\usepackage[english,brazil]{babel}')
!uname -a
```

```
Linux DESKTOP-CR7O8A2 4.19.104-microsoft-standard #1 SMP Wed Feb 19 06:37:35 UTC
2020 x86_64 x86_64 x86_64 GNU/Linux
```

Contents

2 Introdução

Este trabalho visa aplicar o MEF a uma estrutura de treliças.

Para tanto, deverá ler um arquivo em disco com os dados da treliça para resolver via código em Python.

Após ler o arquivo, o código abaixo gera um objeto de classe própria com as propriedades do sistema contido no arquivo.

3 Introdução

4 Teoria

A partir deste sistema, temos as seguintes equações diferenciais para o deslocamento u da barra, considerando a aplicação de uma força Normal F:

[2]:
```
# Declaração das variáveis independentes
x = var('x') # Variável independente (cumprimento)
var('F,N,E,A,L,i,j') # Variáveis simbólicas de apoio
# Considerações e Restrições das variáveis de apoio
assume(N,i,j,'integer') # Número inteiro de elementos
assume(L>0) # Comprimento da barra positivo
assume(N>0) # Pelo menos 1 Elemento
assume(E>0) # Módulo de elasticidade do material positivo
assume(A>0) # Área da seção transversal positiva
print(assumptions()) # Exibe um resumo das restrições asumidas até aqui
# Variáveis dependentes
u = function('u')(x) # Função analítica desconhecida u(x)
# Equações Diferenciais de u(x)
eq1 = E*A*diff(u,x,2) == 0
eq2 = E*A*diff(u,x) == F
eq1.show() # Exibe a equação diferencial de primeira ordem de u(x)
eq2.show() # Exibe a equação diferencial de segunda ordem de u(x)
```

```
[N is integer, i is integer, j is integer, L > 0, N > 0, E > 0, A > 0]
```

```
A*E*diff(u(x), x, x) == 0
```

```
A*E*diff(u(x), x) == F
```

A solução analítica destas equações, dadas abaixo, convergem para a equação conhecida da deformação linear na barra, que é:

$$u(x) = \frac{F}{EA}x$$

[3]:
```
sol1 = u == desolve(eq1,u,ivar=x)
sol2 = u == desolve(eq2,u,ivar=x)
sol1.show() # Solução da EDO de 1 Ordem
sol2.show() # Solução da EDO de 2 Ordem
```

```
u(x) == _K2*x + _K1
```

```
u(x) == _C + F*x/(A*E)
```

A SRP - Sentença de Resíduos Ponderados, fornece a seguinte integral:

$$\int_D \phi_i R dD = 0 \tag{1}$$

Onde,

$$R = \left(EA \frac{d^2 \bar{u}}{dx^2} \right)$$

e,

$$\bar{u} = \sum_{n=1}^{N+1} u_i \phi_i$$

Assim, temos

$$\int_0^L \phi_i \left(EA \frac{d^2 \bar{u}}{dx^2} \right) dx = 0$$

[4]:
```
phi_i = function('phi_i')(x,i) # phi_i(x) da SRP
u_i = function('u_i')(i) # u_i
u_b = sum(u_i*phi_i,i,1,N+1) # û(x) estimado
R = (E*A*u_b.diff(x,2)).full_simplify() # Resíduo
u_b.show() # Exibe û
R.show() # Exibe o resíduo
SRP = (phi_i*R).integrate(x,0,L) # Sentença dos Resíduos Ponderados
SRP.show() # Exibe a SRP
```

```
sum(phi_i(x, i)*u_i(i), i, 1, N + 1)
```

```
(A*E*N + A*E)*u_i(i)*diff(phi_i(x, i), x, x)
```

```
(A*E*N + A*E)*integrate(phi_i(x, i)*diff(phi_i(x, i), x, x), x, 0, L)*u_i(i)
```

Deixando a solução analítica de lado, vamos à implementação

5 Barra horizontal

5.1 Matriz de Rigidez do Elemento

A matriz de Rigidez de uma barra dentro da treliça é dada por:

$$K_{ij} = EA \int_0^L \left(\frac{dN_i}{dx} \frac{dN_j}{dx} \right) dx \tag{2}$$

5.2 Vetor de Cargas Nodais do Elemento

$$f_i = EA \left[N_i \frac{d\bar{u}}{dx} \right]_0^L \tag{3}$$

Tem-se que:

$$\begin{cases} \phi_1(x) & = & \frac{L-x}{x} \\ \\ \phi_2(x) & = & \frac{x}{L} \end{cases} \tag{4}$$

Logo, Simplificando tudo,

$$K_{11} = K_{22} = \frac{EA}{L}$$

$$K_{12} = K_{21} = -\frac{EA}{L}$$

Ou,

$$\mathbf{K}_{local} = \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix}$$

Ou, ainda,

$$\mathbf{K}_{local} = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{5}$$

E,

$$\vec{f}_{local} = F \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

6 Caso Geral - Barra em Qualquer Direção

$$N_1 = \frac{h_e - x_e}{h_e}$$

$$N_2 = \frac{x_e}{h_e}$$

$$N_G = \begin{bmatrix} N_1 cos\alpha & N_1 sin\alpha & N_2 cos\alpha & N_2 sin\alpha \end{bmatrix}$$

Logo,

$$\frac{\partial N}{\partial x} = \begin{bmatrix} \frac{\partial N_1}{\partial x} cos\alpha & \frac{\partial N_1}{\partial x} sin\alpha & \frac{\partial N_2}{\partial x} cos\alpha & \frac{\partial N_2}{\partial x} sin\alpha \end{bmatrix}$$

$$\frac{\partial N}{\partial x} = \begin{bmatrix} -\frac{1}{h_e} cos\alpha & -\frac{1}{h_e} sin\alpha & \frac{1}{h_e} cos\alpha & \frac{1}{h_e} sin\alpha \end{bmatrix}$$

$$\frac{\partial N}{\partial x} = \frac{1}{h_e} \begin{bmatrix} -cos\alpha & -sin\alpha & cos\alpha & sin\alpha \end{bmatrix} = [\mathbf{B}]$$

Bem como,

$$[k_e] = \int_0^{h_e} EA \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx$$

Ou seja,

$$[k_e] = \int_0^{h_e} \left[\mathbf{B^T}\right] EA \, [\mathbf{B}] dx$$

Portanto,

$$[k_e] = \frac{EA}{h_e^2} \begin{bmatrix} cos^2(\alpha) & sin(\alpha)cos(\alpha) & -cos^2(\alpha) & -sin(\alpha)cos(\alpha) \\ sin(\alpha)cos(\alpha) & sin^2(\alpha) & -sin(\alpha)cos(\alpha) & -sin^2(\alpha) \\ -cos^2(\alpha) & -sin(\alpha)cos(\alpha) & cos^2(\alpha) & sin(\alpha)cos(\alpha) \\ -sin(\alpha)cos(\alpha) & -sin^2(\alpha) & sin(\alpha)cos(\alpha) & sin^2(\alpha) \end{bmatrix} \int_0^{h_e} dx = \frac{EA}{h_e^2} [\mathbf{T}] \int_0^{h_e} dx$$

Onde **T** é a Matriz de Transformação ou de Rotação.

Logo,

$$[k_e] = \frac{EA}{h_e^2} [\mathbf{T}] \int_0^{h_e} dx = \left[\frac{EA}{h_e^2} [\mathbf{T}] \, x \right]_0^{h_e} = \frac{EA}{h_e^2} [\mathbf{T}] \, h_e$$

Ou,

$$[k_e] = \frac{EA}{h_e} [\mathbf{T}]$$

7 Implementação do Código

A função abaixo lê um arquivo JSON, cujo endereço relativo está na variável *path*, e retorna o conteúdo como um dicionário Python na variável *file*.

[5]:
```
def readjson(path):
    file = None
    try:
        with open(path,'r') as f:
            file = json.load(f)
    except IOError as err:
        print('File Error: ' + str(err))
    except JSONDecodeError as err:
        print('JSON Error: ' + str(err))
    finally:
        return file
```

Será criada, a seguir, uma classe chamada *node* para armazenar os elementos do tipo Nó, contendo as informações necessárias para a definição da localização e tipo de apoio existente.

[6]:
```
class node:
    def __init__(self,x=0.0,y=0.0,z=0.0,tag=''):
        self.dot = array([x,y,z])
        self.l = zeros([3])
        self.u = zeros([3])
        self.s = array([False,False,False])
        self.tag = tag

    def support(self,rx,ry,rz):
        self.s = array([rx,ry,rz])

    def load(self,fx,fy,fz):
        self.l = array([fx,fy,fz])

    def uloc(self, ul):
        self.u = ul
```

Após a definição dos Nós, agora será criada uma classe para armazenar as barras (colunas, vigas) que compõem a treliça.

[7]:
```
class bar:
    def __init__(self,no1,no2,EA,tag=''):
        self.at = no1.dot # Origem da barra
        self.vec = no2.dot - no1.dot # Vetor (x,y) da barra
        self.EA = EA # Módulo de Elasticidade x área da seção transversal
        self.tag = tag # Nome de referência para a barra
    def K(self):
        L = norm(self.vec)
```

```python
        dx = self.vec[0]
        dy = self.vec[1]
        B = array([[-dx,-dy,dx,dy]])/L
        return B.T*(self.EA/L)*B # Matriz de Rigidez Local

    def K11(self):
        L = norm(self.vec)
        B = array([[self.vec[0],self.vec[1]]])/L
        return B.T*(self.EA/L)*B

    def K12(self):
        return -K11(self)

    def K21(self):
        return -K11(self)

    def K22(self):
        return K11(self)
```

Por último, uma classe geral contendo a treliça em si, com os nós e suas respectivas barras.

Dentro da classe *trelica* também estára o método para calcular a matriz de rigidez K e a solução do sistema pelo método dos deslocamentos.

[8]:
```python
class trelica:
    def __init__(self, file):
        self.n = file['n']
        self.m = 0 # Número de barras
        self.E = array(file['E'])
        self.A = array(file['A'])
        cargas = array(file['loads'])
        self.nos = []
        self.barras = []
        self.K = zeros([2*self.n,2*self.n])
        self.f = zeros([2*self.n])
        self.u = zeros([2*self.n])
        for name, value in file['nodes'].items():
            no = node(value['x'],
                      value['y'],
                      value['z'],
                      name)
            no.support(value['Tx'],
                      value['Ty'],
                      value['Tz'],)
            self.nos.append(no)
        for i in range(self.n):
            self.nos[i].load(cargas[i][0],
                      cargas[i][1],
```

```
                    cargas[i][2])
            # Cálculo dos Vetores u e f
            ff = array([self.nos[i].l[0],self.nos[i].l[1]])
            uu = array([nan,nan])
            if self.nos[i].s[0]:
                uu[0] = 0
                ff[0] = nan
            if self.nos[i].s[1]:
                uu[1] = 0
                ff[1] = nan
            self.f[2*i:2*i+2] = ff
            self.u[2*i:2*i+2] = uu
            for j in range(i,self.n):
                EA = self.E[i,j]*self.A[i,j]
                if EA:
                    barra = bar(self.nos[i],self.nos[j],EA)
                    self.m += 1 # Conta as barras
                    self.barras.append(barra)
                    # Cálculo da Matriz k
                    k = barra.K11()
                    self.K[2*i:2*i+2,2*i:2*i+2] += k
                    self.K[2*i:2*i+2,2*j:2*j+2] -= k
                    self.K[2*j:2*j+2,2*i:2*i+2] -= k
                    self.K[2*j:2*j+2,2*j:2*j+2] += k

    def deslocamentos(self):
        K,u,f = self.K,self.u,self.f
        change = True
        m = 2*self.n
        while change:
            change = False
            for i in range(m):
                if isnan(f[i]):
                    f -= u[i]*K[:,i]
                    K[:,i] = zeros(m)
                    K[i,i] = -1
                    K = delete(K,i,0)
                    K = delete(K,i,1)
                    u = delete(u,i)
                    f = delete(f,i)
                    change = True
                    m -= 1
                    break
        u = solve(K,f)
        k = 0
        desloc = self.u.copy()
        for i in range(2*self.n):
```

```
            if isnan(desloc[i]):
                desloc[i] = u[k]
                k += 1
        forces = self.K.dot(desloc) # Recalcula as forças nodais a partir dos deslocamentos
        for i in range(self.n):
            self.nos[i].uloc(array([desloc[2*i],desloc[2*i+1],0]))
            print('Nó: (',self.nos[i].dot[0],',',self.nos[i].dot[1],'):')
            print('Fx =', forces[2*i], 'kN, Fy =', forces[2*i+1],'kN')
            print('dx =', 1000*desloc[2*i], 'mm, dy =', 1000*desloc[2*i+1],'mm\n\n')
        return

    def tensoes(self):
        self.deslocamentos()
        for i in range(self.n):
            for j in range(i,self.n):
                EA = self.E[i,j]*self.A[i,j]
                if EA:
                    c = self.nos[j].dot[0] - self.nos[i].dot[0] # delta x da barra
                    s = self.nos[j].dot[1] - self.nos[i].dot[1] # delta y da barra
                    L = norm(array([c,s]))
                    Bl = array([-c,-s,c,s])/L
                    ul = array([self.nos[i].u[0],self.nos[i].u[1],self.nos[j].u[0],self.nos[j].u[1]])
                    N = (EA/L)*Bl.dot(ul)
                    if N > 0:
                        print('A Barra do Nó (', self.nos[i].dot[0], ',', self.nos[i].dot[1], ') para o Nó (', self.nos[j].dot[0], ',', self.nos[j].dot[1], ') está sujeita a uma tração de', N, 'kN\n\n')
                    elif N < 0:
                        print('A Barra do Nó (', self.nos[i].dot[0], ',', self.nos[i].dot[1], ') para o Nó (', self.nos[j].dot[0], ',', self.nos[j].dot[1], ') está sujeita a uma compressão de', N, 'kN\n\n')
                    else:
                        print('A Barra do Nó (', self.nos[i].dot[0], ',', self.nos[i].dot[1], ') para o Nó (', self.nos[j].dot[0], ',', self.nos[j].dot[1], ') está sem carregamento\n\n')
        return

    def desenha(self):
        p = line([])
        for i in range(self.m):
```

```
            p += line([self.barras[i].at[0:-1], self.barras[i].at[0:-1]+self.
↪barras[i].vec[0:-1]])
        return p
```

8 Exercício 1 - Sistema Com 3 Barras

O arquivo a ser lido está no formato JSON e será armazenado na variável *parser*.

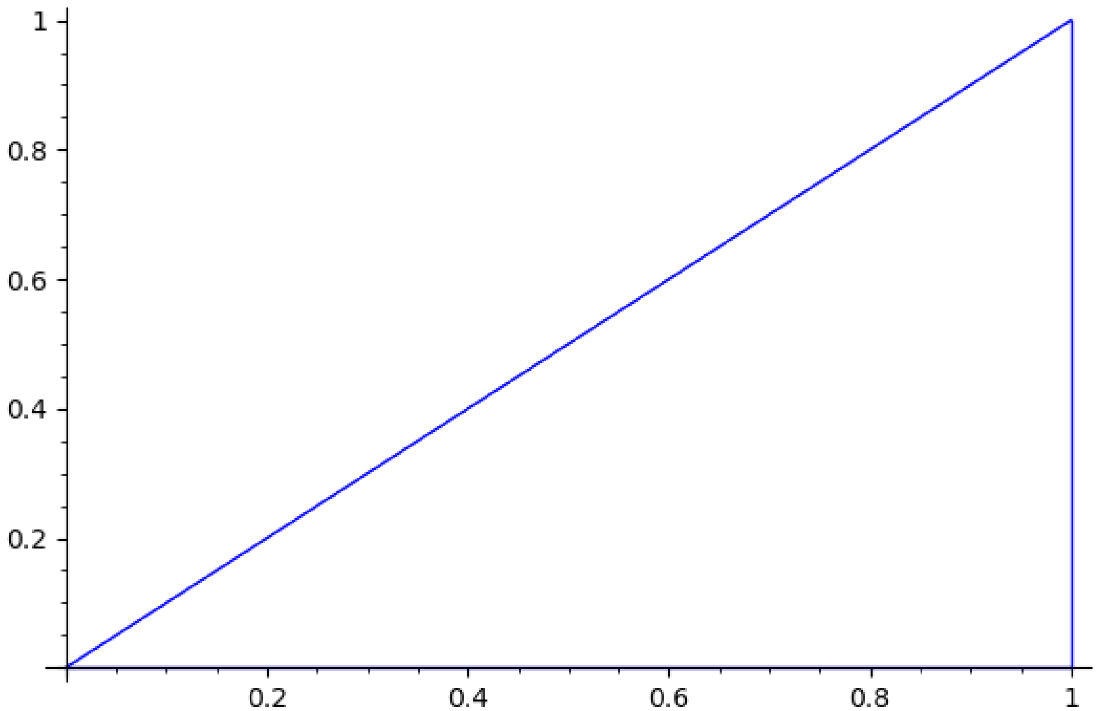
A partir do valor no arquivo JSON, em *parser*, será criada a treliça e armazenada em *tr*

[9]:
```
parser3n = readjson("trelica3nos.json")
print(json.dumps(parser3n, indent=4, sort_keys=True)) # Exibir conteúdo do␣
↪arquivo lido
trel3n = trelica(parser3n) # Treliça de 3 nós para teste
trel3n.desenha().show() # Desenha as barras da estrutura
trel3n.tensoes() # apresenta os deslocamentos nos nós, reações de apoio e␣
↪carregamento nas barras
```


```
{
    "A": [
        [
            0.0,
            1.0,
            1.0
        ],
        [
            1.0,
            0.0,
            1.0
        ],
        [
            1.0,
            1.0,
            0.0
        ]
    ],
    "E": [
        [
            0.0,
            1000.0,
            1000.0
        ],
        [
            1000.0,
            0.0,
            1000.0
        ],
```

```
        [
            1000.0,
            1000.0,
            0.0
        ]
    ],
    "dim": 2,
    "loads": [
        [
            0.0,
            0.0,
            0.0
        ],
        [
            0.0,
            0.0,
            0.0
        ],
        [
            1.0,
            -1.0,
            0.0
        ]
    ],
    "n": 3,
    "nodes": {
        "No 1": {
            "Mx": false,
            "My": false,
            "Mz": false,
            "Tx": true,
            "Ty": true,
            "Tz": false,
            "x": 0.0,
            "y": 0.0,
            "z": 0.0
        },
        "No 2": {
            "Mx": false,
            "My": false,
            "Mz": false,
            "Tx": false,
            "Ty": true,
            "Tz": false,
            "x": 1.0,
            "y": 0.0,
            "z": 0.0
        },
```

```
        "No 3": {
            "Mx": false,
            "My": false,
            "Mz": false,
            "Tx": false,
            "Ty": false,
            "Tz": false,
            "x": 1.0,
            "y": 1.0,
            "z": 0.0
        }
    }
}
```

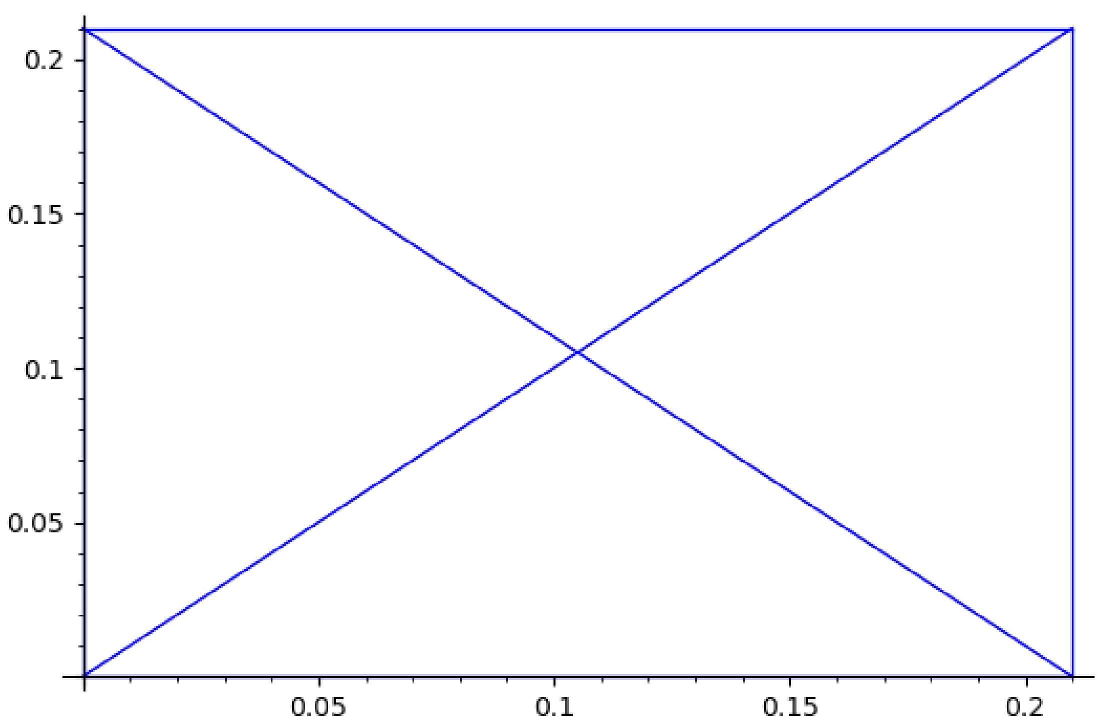

```
Nó: ( 0.0 , 0.0 ):
Fx = -1.0 kN, Fy = -1.0 kN
dx = 0.0 mm, dy = 0.0 mm


Nó: ( 1.0 , 0.0 ):
Fx = 0.0 kN, Fy = 2.0 kN
dx = 0.0 mm, dy = 0.0 mm
```

```
Nó: ( 1.0 , 1.0 ):
Fx = 1.0 kN, Fy = -1.0 kN
dx = 4.828427124746192 mm, dy = -2.0 mm


A Barra do Nó ( 0.0 , 0.0 ) para o Nó ( 1.0 , 0.0 ) está sem carregamento


A Barra do Nó ( 0.0 , 0.0 ) para o Nó ( 1.0 , 1.0 ) está sujeita a uma tração de
1.4142135623730956 kN


A Barra do Nó ( 1.0 , 0.0 ) para o Nó ( 1.0 , 1.0 ) está sujeita a uma
compressão de -2.0 kN
```

9 Exercício 1 - Sistema Com 6 Barras

O exemplo a seguir contém 4 nós e 6 barras.

[10]:
```
trel4n = trelica(readjson("trelica4nos.json")) # Treliça de 4 nós para teste
trel4n.desenha().show() # Desenha as barras da estrutura
trel4n.tensoes() # apresenta os deslocamentos nos nós, reações de apoio e
 ↪carregamento nas barras
```



Nó: (0.0 , 0.0):
Fx = 0.9999999999999996 kN, Fy = 0.5714285714285712 kN
dx = 0.0 mm, dy = 0.0 mm

Nó: (0.0 , 0.21):
Fx = -0.9999999999999996 kN, Fy = 0.4285714285714284 kN
dx = 0.0 mm, dy = 0.0 mm

Nó: (0.21 , 0.21):
Fx = 1.1102230246251565e-16 kN, Fy = -1.0 kN
dx = 0.5714285714285712 mm, dy = -1.7142857142857137 mm

Nó: (0.21 , 0.0):
Fx = 0.0 kN, Fy = 2.220446049250313e-16 kN
dx = -0.4285714285714283 mm, dy = -1.2857142857142851 mm

A Barra do Nó (0.0 , 0.0) para o Nó (0.0 , 0.21) está sem carregamento

A Barra do Nó (0.0 , 0.0) para o Nó (0.21 , 0.21) está sujeita a uma compressão de -0.8081220356417683 kN

A Barra do Nó (0.0 , 0.0) para o Nó (0.21 , 0.0) está sujeita a uma compressão de -0.4285714285714283 kN

A Barra do Nó (0.0 , 0.21) para o Nó (0.21 , 0.21) está sujeita a uma tração de 0.5714285714285712 kN

A Barra do Nó (0.0 , 0.21) para o Nó (0.21 , 0.0) está sujeita a uma tração de 0.6060915267313262 kN

A Barra do Nó (0.21 , 0.21) para o Nó (0.21 , 0.0) está sujeita a uma compressão de -0.42857142857142855 kN