João Vitor Sanches 9833704

Em conjunto com Victor Chacon Codesseira 9833711

Aula 4 – Análise de vigas

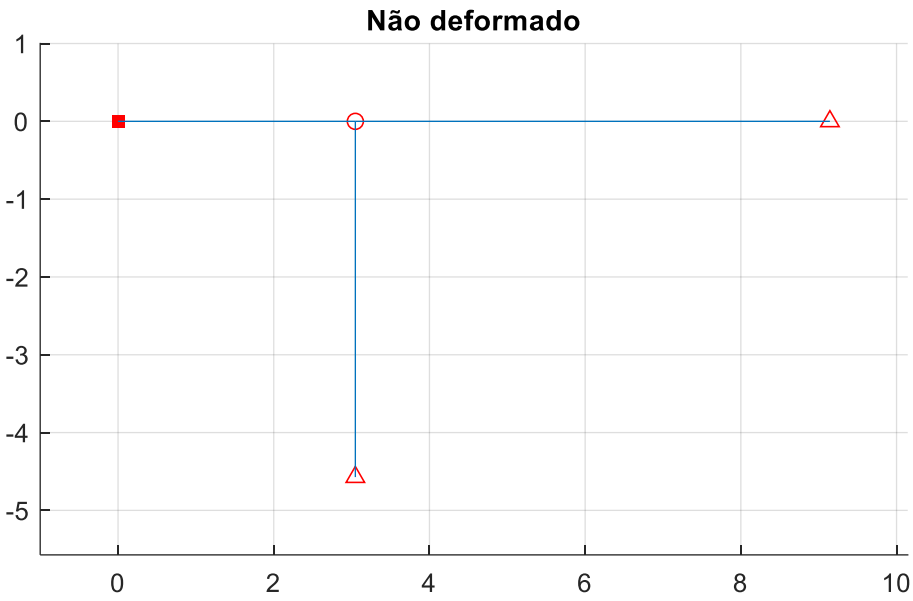
Com o elemento de viga incluído no programa, foi possível estender sua funcionalidade para o objetivo da atividade corrente:
Para o primeiro problema, o arquivo de entrada utilizado foi:

Ex_vigas_2.txt

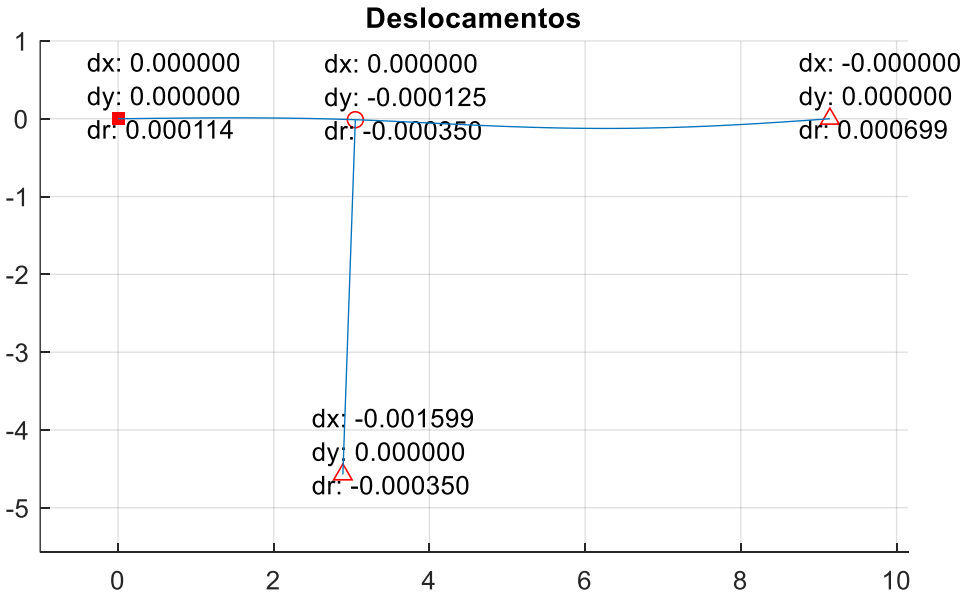
```
#HEADER
Ex 2 com vigas
Unidades SI
Cabecalho de arquivo padrao
Separar secoes com linhas vazias

#DYNAMIC
0

#NODES
0 0
3.048 0
9.144 0
3.048 -4.572

#ELEMENTS
b 1 2 0.0129032 200e9 0.00075 0
b 2 3 0.0129032 200e9 0.00075 0
b 2 4 0.0129032 200e9 0.00075 0


#LOADS

#PRESSURES
@2
0 -15700

#CONSTRAINTS
@1
0 0 u
@3
u 0 u
@4
u 0 u
```

O tipo de elemento passou a ser descrito pelas letras t ou b, no início de cada linha. Além disso, cargas distribuídas são especificadas na seção PRESSURES, tendo distribuição uniforme e em todo o elemento.

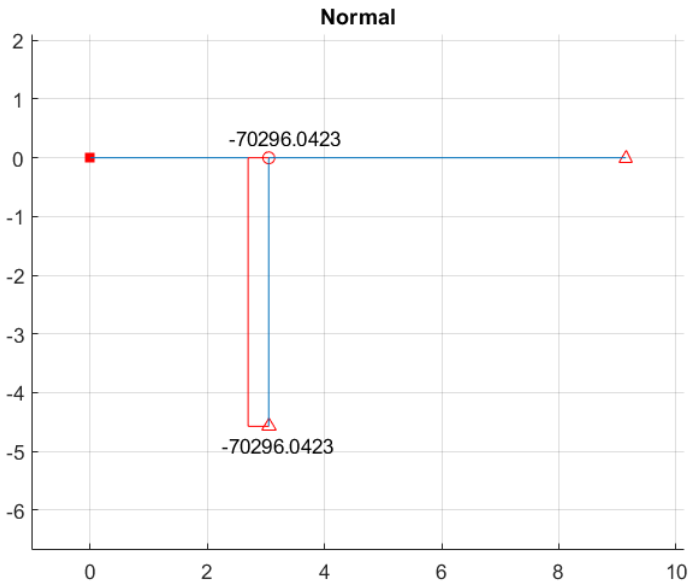
Algumas pequenas modificações foram feitas dos demais arquivos do programa de modo a tratar os 3 graus de liberdade por nó.
Os resultados obtidos com a execução do programa foram:



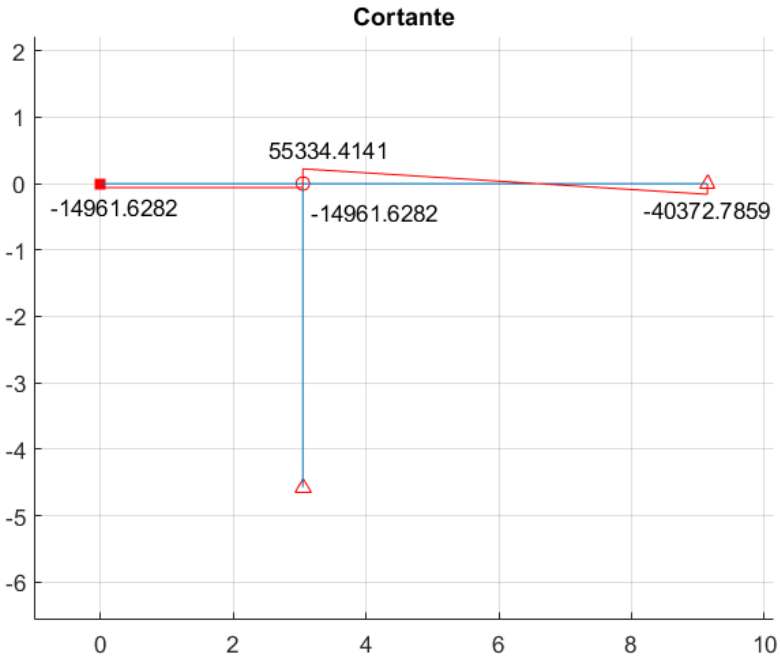
(Visualização da entrada, não deformada)



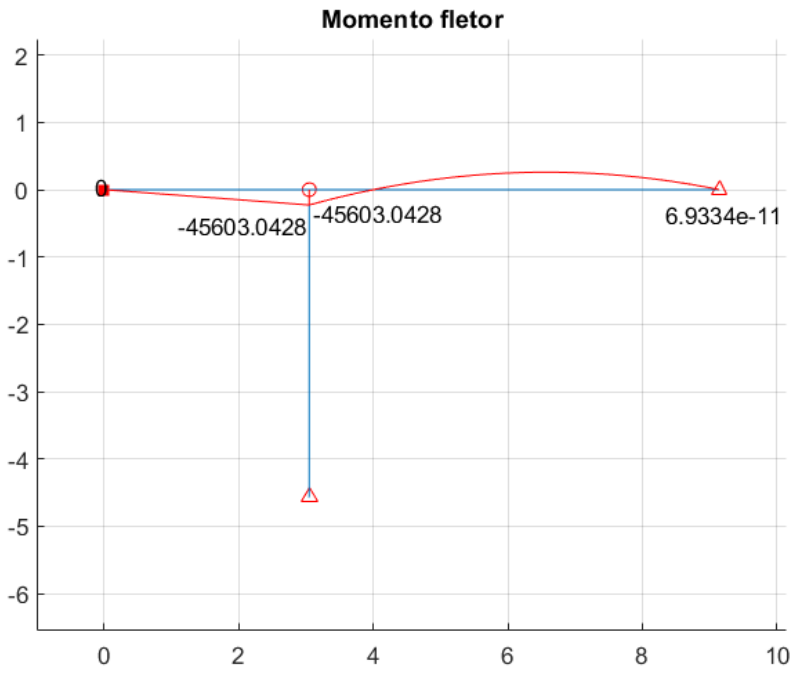
(Estrutura deformada e valores anotados nos pontos)



(Diagrama de força normal, em vermelho, nos membros da estrutura, em azul)



(Diagrama de força cortante, em vermelho, nos membros da estrutura, em azul)

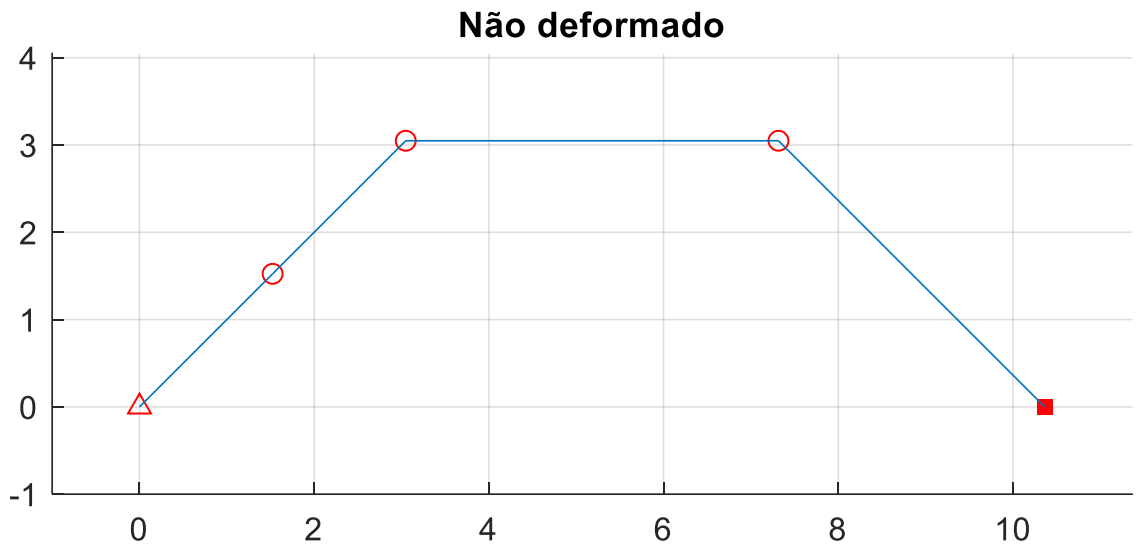


(Diagrama de momento fletor, em vermelho, nos membros da estrutura, em azul)

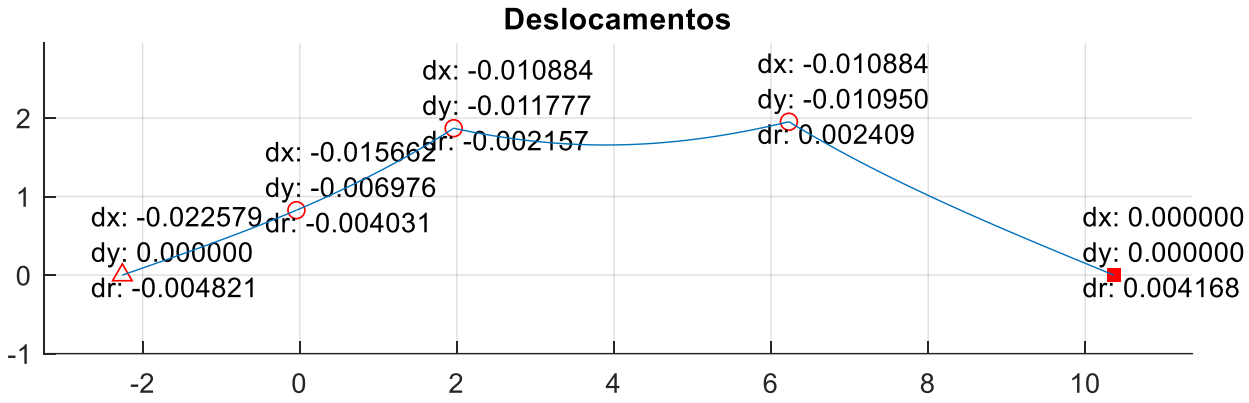
Para o segundo problema:
Ex_vigas_3.txt

```
#HEADER
Ex 3 com vigas
Cabecalho de arquivo padrao
Separar secoes com linhas vazias

#DYNAMIC
0

#NODES
0 0
1.524 1.524
3.048 3.048
7.3152 3.048
10.3632 0

#ELEMENTS
b 1 2 0.013 200e9 0.00075 0
b 2 3 0.013 200e9 0.00075 0
b 3 4 0.013 200e9 0.00075 0
b 4 5 0.013 200e9 0.00075 0


#LOADS
@2
0 -45359.24 0

#PRESSURES
@3
0 -15700

#CONSTRAINTS
@1
u 0 u
@5
0 0 u
```

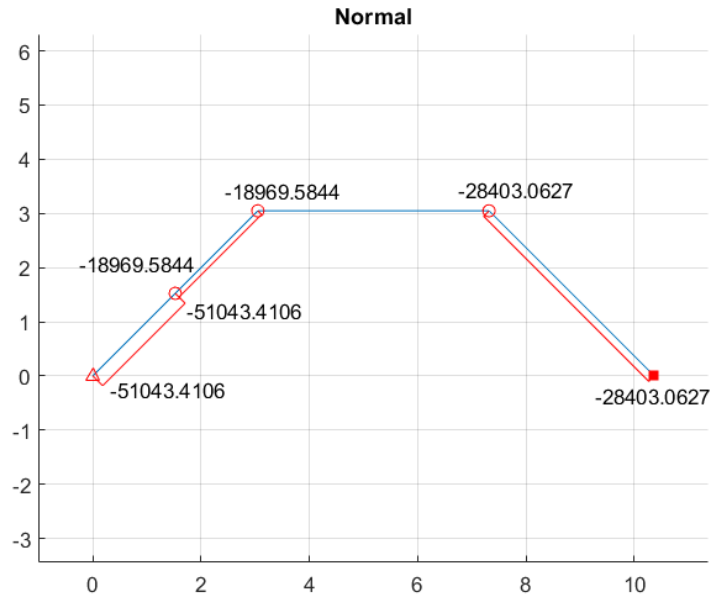
As saídas do programa foram:



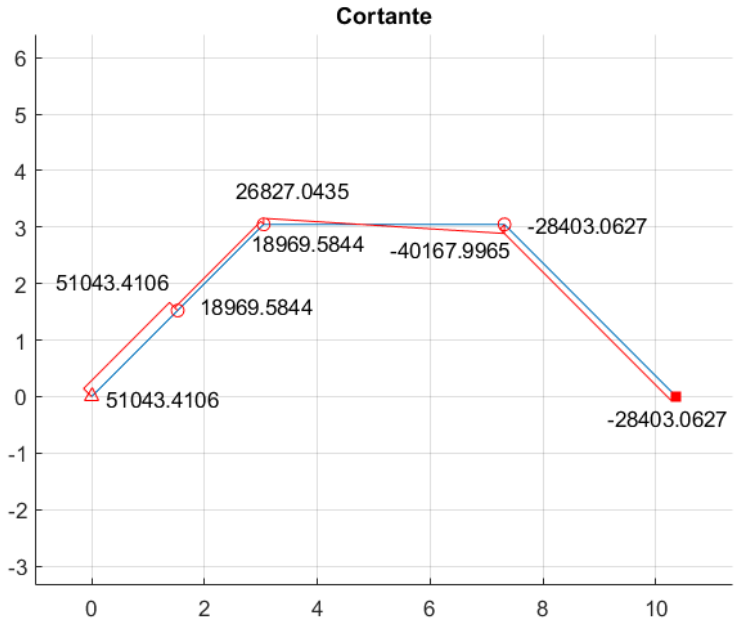
(Visualização da entrada, não deformada)



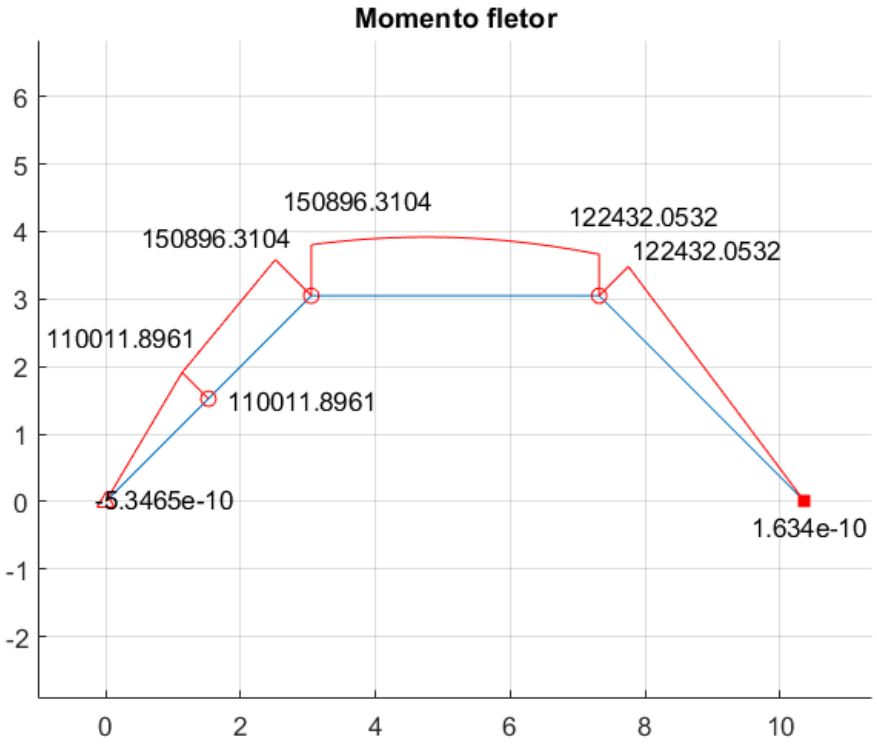
(Estrutura deformada e valores anotados nos pontos)



(Diagrama de força normal, em vermelho, nos membros da estrutura, em azul)



(Diagrama de força cortante, em vermelho, nos membros da estrutura, em azul)



(Diagrama de momento fletor, em vermelho, nos membros da estrutura, em azul)

Em https://github.com/jvSanches/PMR5026 é possível ter acesso ao programa completo...
Em adição ao anterior, o código foi incrementado com a listagem abaixo:

postProcessor.m

```
scale_deform = 100;
scale_normal = 5e-6;
scale_shear = 4e-6;
scale_moment = 5e-6;

disp("Showing results...");
scatterNodes(nodes, elements, false, true, 0);
title("Não deformado")

fig_u = scatterNodes(nodes, elements, true, false, scale_deform);
title("Deslocamentos")
for i = 1:length(nodes)
    txt = sprintf("dx: %f\ndy: %f\ndr: %f", [nodes(i).dx, nodes(i).dy,
nodes(i).dtheta]);
    text(nodes(i).x + nodes(i).dx*scale_deform - 0.4, nodes(i).y +
nodes(i).dy*scale_deform + 0.3, txt);
end

fig_N = scatterNodes(nodes, elements, false, true, 0);
title("Normal")
axis equal
fig_V = scatterNodes(nodes, elements, false, true, 0);
title("Cortante")
axis equal
fig_M = scatterNodes(nodes, elements, false, true, 0);
title("Momento fletor")
axis equal

for i=1:length(elements)
    elements(i) = elements(i).calculateStress();
    el = elements(i);
    x = linspace(0, el.L, 101);
    T = [el.l, el.m; -el.m, el.l]^-1;

    %% Display displacement
    figure(fig_u)

    u = x + scale_deform*(eval(subs(el.elasticLineX))-x);
    v = scale_deform*(eval(subs(el.elasticLineY)));
    points = T*[u;v] + [el.n1.x; el.n1.y];

    plot(points(1,:), points(2,:), 'Color', [0 0.447 0.741]);


    %% Display normal
    plotDiagram(fig_N, el, el.Normal, x, T, scale_normal);

    %% Display shear
    plotDiagram(fig_V, el, el.Shear, x, T, scale_shear);

    %% Display moment
```

```
    plotDiagram(fig_M, el, el.Moment, x, T, scale_moment);
end

%% Function for scatter of nodes
function fig = scatterNodes(nodes, elements, displaced, lines, scale)
    fig = figure();
    hold on;
    grid on;

    max_x = -inf;
    min_x = inf;
    max_y = -inf;
    min_y = inf;
    for i=1:length(nodes)
        nx = nodes(i).x;
        ny = nodes(i).y;
        if displaced
            nx = nx + scale*nodes(i).dx;
            ny = ny + scale*nodes(i).dy;
        end

        max_x = max(max_x, nx);
        max_y = max(max_y, ny);
        min_x = min(min_x, nx);
        min_y = min(min_y, ny);
        if nodes(i).xconstrained
            if nodes(i).yconstrained
                scatter(nx, ny, 's', 'filled', 'red');
            else
                scatter(nx, ny, '>', 'filled', 'red');
            end
        else
            if nodes(i).yconstrained
                scatter(nx, ny,'^' ,'red');
            else
                scatter(nx, ny, 'red');
            end
        end
    end

    offset = 1;
    axis equal
    axis([min_x-offset max_x+offset min_y-offset max_y+offset])


    if lines
        for i=1:length(elements)
            if displaced
                line_x = [elements(i).n1.x+scale*elements(i).n1.dx,
elements(i).n2.x+scale*elements(i).n2.dx];
                line_y = [elements(i).n1.y+scale*elements(i).n1.dy,
elements(i).n2.y+scale*elements(i).n2.dy];
            else
                line_x = [elements(i).n1.x, elements(i).n2.x];
                line_y = [elements(i).n1.y, elements(i).n2.y];
            end
```

```
            line(line_x,line_y)
        end
    end
end

function plotDiagram(fig, element, equation, divisions, rot, scale)
    offset_x = -0.15;
    offset_y = 0.05;
    figure(fig);
    syms x
    if (isnumeric(equation) && abs(equation) < 1e-6) || (~isnumeric(equation)
&& isnumeric(eval(equation)) && abs(eval(equation)) < 1e-6)
        return
    end
    data = eval(subs(equation, x, divisions));
    if abs(data) < 1e-6
        return
    end

    points = rot*[divisions;scale*data] + [element.n1.x; element.n1.y];

    plot(points(1,:), points(2,:), 'r');
    line([points(1,1), element.n1.x], [points(2,1), element.n1.y], 'Color',
'r');
    line([points(1,end), element.n2.x], [points(2,end), element.n2.y],
'Color', 'r');

    text(points(1,1) + offset_x, points(2,1) + offset_y, string(data(1)));
    text(points(1,end) + offset_x, points(2,end) + offset_y,
string(data(end)));
end
```