UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Aerospace Structures

2D bars structure: Bicycle

Sergi Marsol

Nicolás Cortines

Greta- Group 11

PROFESSOR

Juan Carlos Cante

Sunday 3rd March, 2024

Contents

1 Primer Análisis **2**
1.1 Algoritmo 2
1.2 Pregunta A 4
1.3 Pregunta B 5

2 Segundo Análisis **6**
2.1 Pregunta A 6
2.2 Pregunta B 10
2.3 Pregunta C 13
2.4 Pregunta D 13

3 Conclusiones **16**

A Código empleado **17**
A.1 Funciones 17

El objetivo de esta tarea es estudiar la respuesta estructural de una bicicleta. Para hacerlo, el análisis se dividirá en dos partes: (1) el cuadro y (2) las ruedas. Cabe mencionar que todos los resultados se han obtenido con una $g = 9.8\ \frac{m}{s^2}$.

1 Primer Análisis

El objetivo de este primer análisis es transferir las cargas aplicadas por el ciclista a las ruedas. Para ello, se estudia la estructura del cuadro, hecha de un material con un módulo de Young $E = 71\,\text{GPa}$, como se muestra en la Figura 1. La masa del ciclista es de 75 kg y se aplica una fuerza neta en la dirección horizontal F para producir una aceleración hacia adelante de $a = 2.5\,\text{m/s}^2$. Suponiendo que la masa del cuadro es despreciable, se solicita lo siguiente:

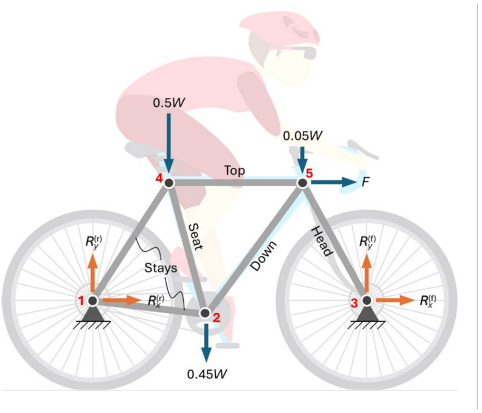

Figure 1: Bicycle case analysis

En el modelo estructural del cuadro con barras, el peso del ciclista W está distribuido entre los nodos 2, 4 y 5, y la fuerza horizontal neta F se aplica en el nodo 5. El desplazamiento en los nodos 1 y 3 está fijo. Las características geométricas de la sección transversal de los tubos se proporcionan en la siguiente tabla:

Tube	Ø (mm)	*t* (mm)
Head	36	1.5
Down		
Top	30	1.2
Seat		
Stays	20	1.0

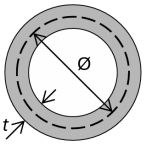

Figure 2: Sección transversal

Por ende, en esta primera parte se nos plantean dos cuestiones, por un lado, realizar el gráfico de la estructura del cuadro deformado y estado de esfuerzos de cada barra. Por otro lado, se han obtenido las reacciones en cada uno de los nodos.

1.1 Algoritmo

Para realizar este primer apartado, se han tenido que completar todas las funciones del código proporcionado. Cabe destacar que dichas funciones deben de estar configuradas **de forma genérica**, para poder reutilizarlas en otros casos de análisis.

Por consiguiente, el proceso en la construcción del algoritmo para resolver problemas estructurales numéricamente se detalla a continuación. Cabe destacar que, si bien el algoritmo es el mismo para

todo tipo de problemas que implican la evaluación de la deformación y el estado de esfuerzos de una estructura (cuasi) estática, los pasos en cada bloque pueden diferir según la tipología del problema.

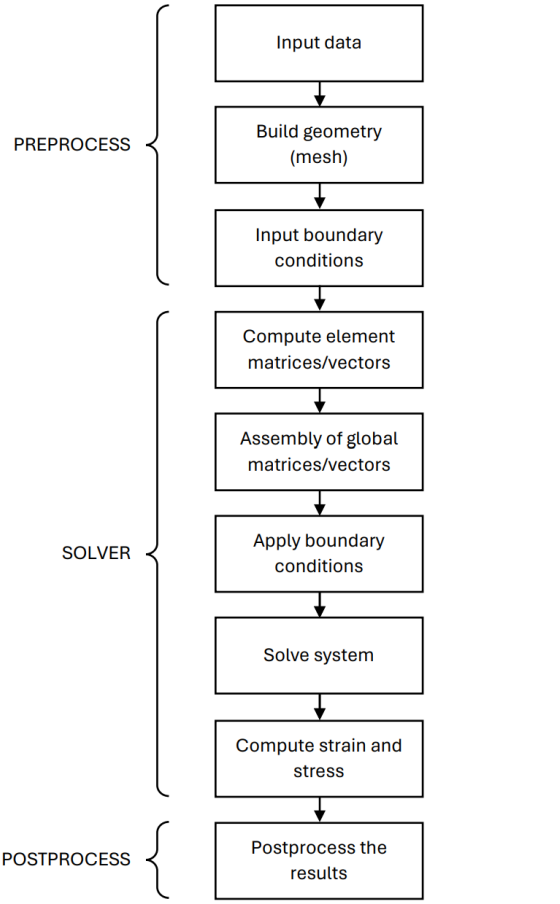

Figure 3: Algoritmo general de resolución de problemas estructurales.

A continuación, se detalla el proceso de *Preprocess*, en el cual se define la geometría del problema y se generan los datos requeridos por el solucionador en un formato completo y compatible. Según la tipología de dicho problema:

En primer lugar, se disponen de los siguientes datos de *Geometry and Boundary conditions*:

- **Elementos:** $\{(1), (2), (3), (4), (5), (6)\}$
- **Nodos:** $\{1, 2, 3, 4, 5\}$
- **Desplazamientos impuestos:** $\{u_{x1}, u_{y1}, u_{x3}, u_{y3}\}$
- **Incógnitas:** Todos los desplazamientos restantes y las cuatro reacciones, $\{R_{x1}, R_{y1}, R_{x3}, R_{y3}\}$
- **Fuerzas externas:** producidas por el peso y la fuerza ejercida en el manillar.

Una vez planteado las condiciones iniciales del problema, es posible obtener cada una de las matrices de los procesos: de *Input data*, *mesh*, y *boundary conditions*.

- **Input data:** matriz $[X]$, la cual expresa cada una de las posiciones de los nodos y matriz $[T_n]$, la cual relaciona cada uno de los elementos con sus nodos.
- **Mesh:** matriz $[m]$, que define el material y matriz $[T_m]$, la cual asigna a cada elemento el material empleado.
- **Boundary conditions:** matriz $[p]$, define los desplazamientos impuestos, y matriz $[F]$, la cual define las fuerzas externas aplicadas.

Con respecto el apartado del *Solver*, el proceso ha consistido en analizar cada una de las barras de forma local, y por consiguiente, pasar a partir de la matriz de rotación a los ejes globales para poder completar el ensamblaje de la *global stiffness matrix* $[K]$.

Una vez completados todos los pasos, ya es posible contestar las preguntas planteadas y utilizar dicho código, cambiando los datos de *Input data*, según cada ejercicio.

1.2 Pregunta A

A continuación, se muestra el gráfico de la estructura del cuadro deformada y estado de esfuerzos de cada barra.

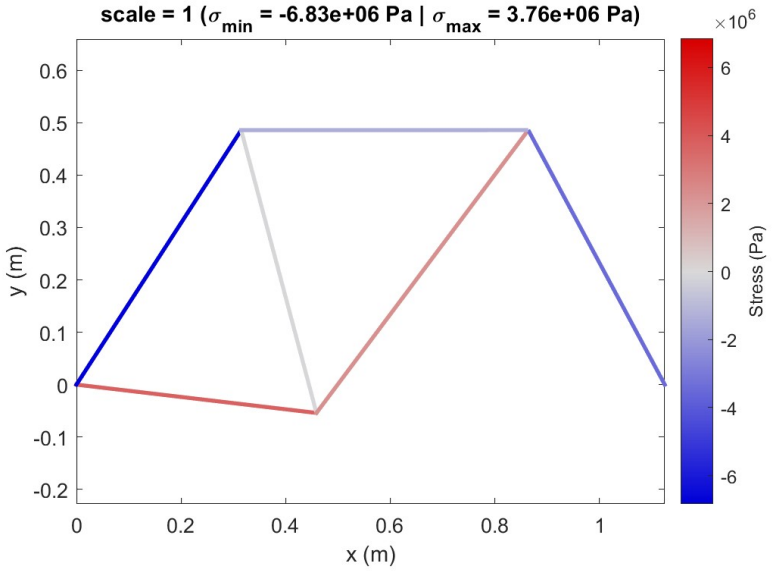

Figure 4: Estado de esfuerzos del cuadro de la bicicleta

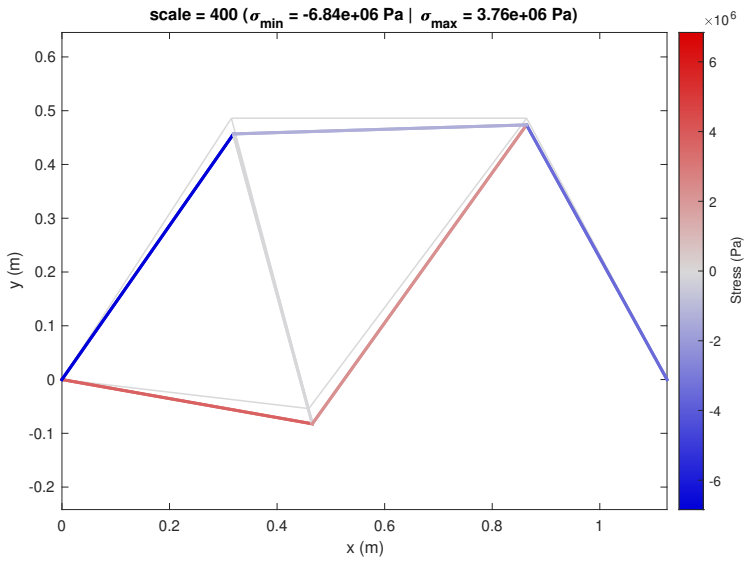

Figure 5: Estado de esfuerzos del cuadro de la bicicleta, escala 400

$\sigma_{\min}$ (MPa)	$\sigma_{\max}$ (MPa)
-6.83	3.76

Se observa que las barras inferiores en contacto con los pedales, es decir, los elementos 1 y 4, están sometidas a un esfuerzo de tracción. Este análisis estructural es coherente, ya que en el punto de unión de estos dos elementos se encuentran los pedales, que se modelizan como una fuerza debido al peso. Específicamente, el esfuerzo de tracción máximo se observa en la barra que une los pedales con la rueda trasera, el elemento 1, dado que en cada uno de los nodos de esta barra se encuentran, por un lado, las reacciones y, por otro lado, la fuerza ejercida en los pedales, como se mencionó anteriormente.

Por otro lado, las barras que unen las ruedas con el sillín y el manillar se encuentran sometidas a compresión, elementos 2 y 5. Al mismo tiempo, la barra que une el sillín con el manillar, elemento 6, también experimenta un esfuerzo de compresión, aunque ligeramente menor que los otros dos elementos. Este análisis estructural es coherente, ya que en las barras que conectan las ruedas con el sillín y el manillar se aplican fuerzas externas de carácter compresivo en los nodos del sillín y el manillar, nodos 4 y 5, respectivamente. En concreto, el elemento 2 experimenta el mayor esfuerzo compresivo, dado que en el nodo 4, el peso, como fuerza externa, tiene una mayor influencia que en otros puntos del cuadro de la bicicleta.

1.3 Pregunta B

A continuación, se presentan las fuerzas de reacción en los nodos prescritos, es decir, los nodos 1 y 3. La matriz R de reacciones es:

$$R = \begin{bmatrix} -1.1106 \\ 387.9300 \\ -186.3894 \\ 347.0700 \end{bmatrix} \text{N}$$

donde las dos primeras filas hacen referencia al nodo 1, en las direcciones x e y, y las dos siguientes al nodo 3, en las mismas direcciones. Cabe mencionar que los valores de reacciones negativas implican que el sentido es opuesto al establecido inicialmente.

2 Segundo Análisis

Para modelar la respuesta estructural de la rueda, se considera la estructura de barras simplificada representada en la Figura 6. En este caso, las cargas de entrada corresponden a las reacciones de la estructura del marco obtenidas en la Parte 1.

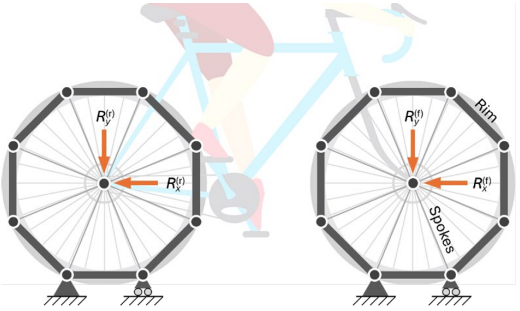

Figure 6: Modelo estructural de las ruedas con barras

El radio de cada rueda es de 35 cm. La siguiente tabla proporciona las propiedades del material y la sección transversal de las llantas y los radios.

Part	*E* (GPa)	A (mm²)	I (mm⁴)
Rim	70	140	1470
Spokes	210	3.8	1.15

Por ende, en esta segunda parte se nos plantean varias cuestiones. Por un lado, obtener las matrices de coordenadas nodales y conectividad para el modelo de las ruedas, graficar la estructura deformada y el estado de tensión de cada barra, y por último, evaluar el riesgo de pandeo de los radios y determinar el valor de pretensión necesario para evitar el fenómeno.

Cabe mencionar que las ruedas, es decir, los elementos que estamos estudiando, se modelizan como octágonos. Con el objetivo de probar distintas geometrías, el código se ha diseñado de tal manera que se puede aproximar la rueda al polígono de n lados deseado.

2.1 Pregunta A

A continuación, se han obtenido las matrices de coordenadas nodales y conectividades para el modelo de barras. Como se ha comentado, se ha programado el código que compone el *input data*, *mesh* y *boundary conditions* de tal forma que es aplicable para cualquier polígono de n lados. Para el caso de la aproximación de la rueda como octágono:

La matriz de **Coordenadas Nodales** queda:

$$x = \begin{bmatrix} -0.133939201327781 & -0.323357836378950 \\ 0.133939201327781 & -0.323357836378950 \\ 0.323357836378950 & -0.133939201327781 \\ 0.323357836378950 & 0.133939201327781 \\ 0.133939201327781 & 0.323357836378950 \\ -0.133939201327781 & 0.323357836378950 \\ -0.323357836378950 & 0.133939201327781 \\ -0.323357836378950 & -0.133939201327781 \\ 0 & 0 \\ 0.991060798672219 & -0.323357836378950 \\ 1.258939201327781 & -0.323357836378950 \\ 1.448357836378950 & -0.133939201327781 \\ 1.448357836378950 & 0.133939201327781 \\ 1.258939201327781 & 0.323357836378950 \\ 0.991060798672219 & 0.323357836378950 \\ 0.801642163621050 & 0.133939201327781 \\ 0.801642163621050 & -0.133939201327781 \\ 1.125000000000000 & 0 \end{bmatrix} \text{m}$$

Donde el origen de coordenadas se encuentra en el centro de la rueda trasera, correspondiente al nodo 9. Por otro lado, para obtener las coordenadas de la rueda delantera se han tomado las posiciones de la rueda delantera y se ha sumado la distancia horizontal entre ambos centros, la cual es $d = 1.125\,\text{m}$.

Las matrices de **Conectividades**, para la rueda trasera, quedan:

En primer lugar, la matriz que relaciona el número del elemento con sus grados de libertad:

$$Td = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 6 \\ 5 & 6 & 7 & 8 \\ 7 & 8 & 9 & 10 \\ 9 & 10 & 11 & 12 \\ 11 & 12 & 13 & 14 \\ 13 & 14 & 15 & 16 \\ 15 & 16 & 1 & 2 \\ 17 & 18 & 1 & 2 \\ 17 & 18 & 3 & 4 \\ 17 & 18 & 5 & 6 \\ 17 & 18 & 7 & 8 \\ 17 & 18 & 9 & 10 \\ 17 & 18 & 11 & 12 \\ 17 & 18 & 13 & 14 \\ 17 & 18 & 15 & 16 \end{bmatrix}$$

Por otro lado, la siguiente matriz relaciona cada elemento con sus propiedades según el material:

$$Tm = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

Por último,

$$Tn = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 6 \\ 6 & 7 \\ 7 & 8 \\ 8 & 1 \\ 9 & 1 \\ 9 & 2 \\ 9 & 3 \\ 9 & 4 \\ 9 & 5 \\ 9 & 6 \\ 9 & 7 \\ 9 & 8 \end{bmatrix}$$

De esta matriz cabe mencionar que el nodo 1 se ubica en la posición de contacto de más a la izquierda de la rueda con el suelo, empezando por la rueda de la izquierda. La rueda delantera se computa, simplemente sumándole a la trasera $n + 1$, donde n es el número de lados del polígono.

A continuación, se muestran los gráficos que representan la geometría de las ruedas según su aproximación al polígono que se considere, siguiendo el recorrido de sus nodos.

Para el caso en que las ruedas son aproximadas como un octágono:

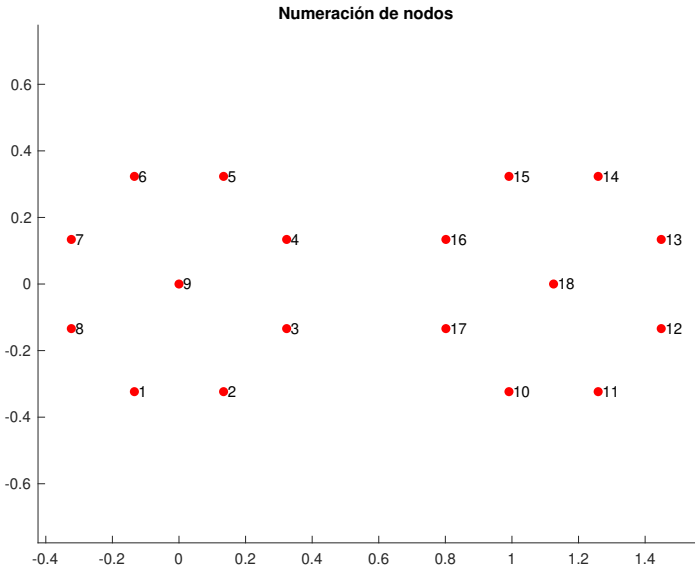

Figure 7: Numeración nodos, octágono

Para el caso en que las rueda son aproximadas como un isodecágono:

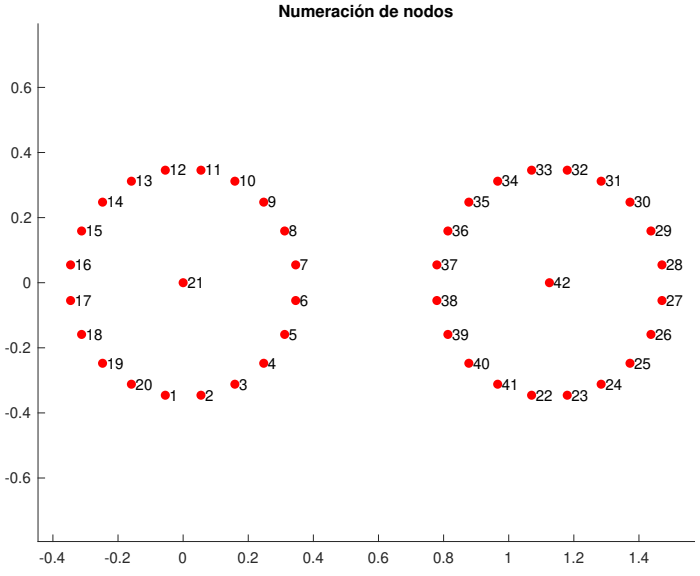

Figure 8: Numeración nodos, Polígono n lados (en este caso $n = 20$)

Como se puede observar en el código, se ha generalizado para poder calcular la estructura en función del número de radios deseado.

2.2 Pregunta B

En esta pregunta se procede a graficar la estructura deformada y el estado de esfuerzos de cada barra. Asimismo, con el objetivo de mostrar cómo varían los esfuerzos en las barras según la geometría de la rueda, se ha realizado el gráfico para dos casos: rueda octagonal ($n = 8$) y rueda con forma de polígono de 20 lados ($n = 20$).

Caso **rueda octagonal** ($n = 8$):

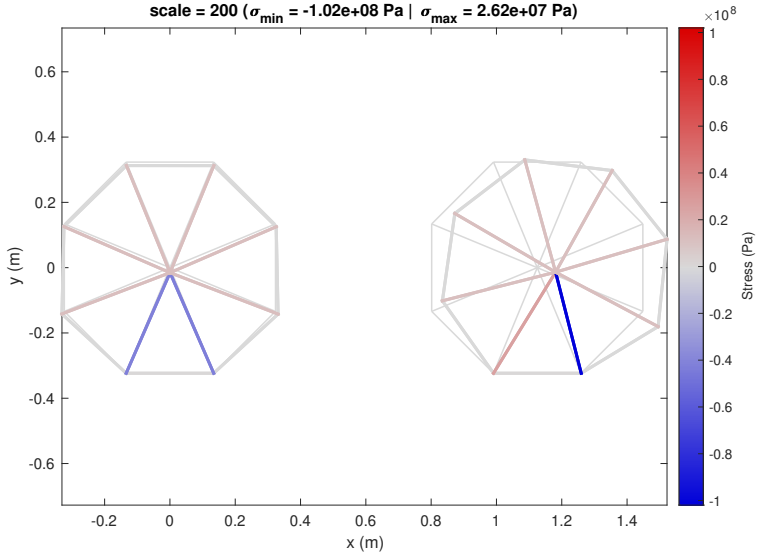

Figure 9: Diagrama, esfuerzos y desplazamientos, ruedas

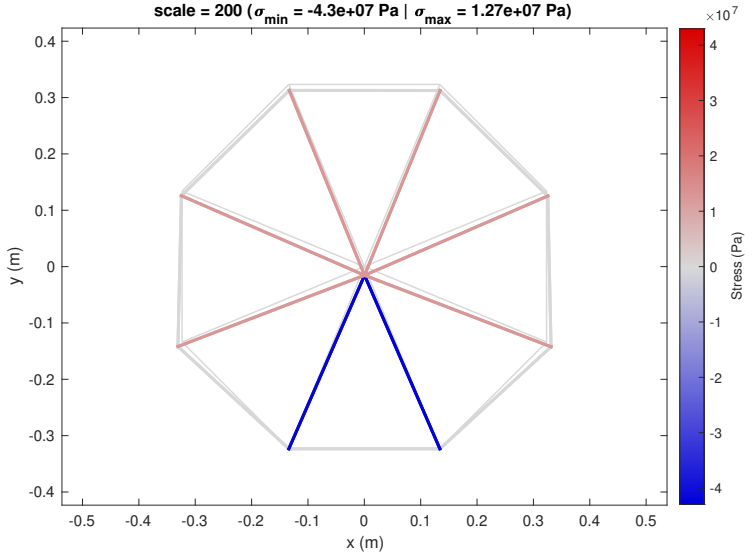

Figure 10: Diagrama, esfuerzos y desplazamientos, rueda trasera

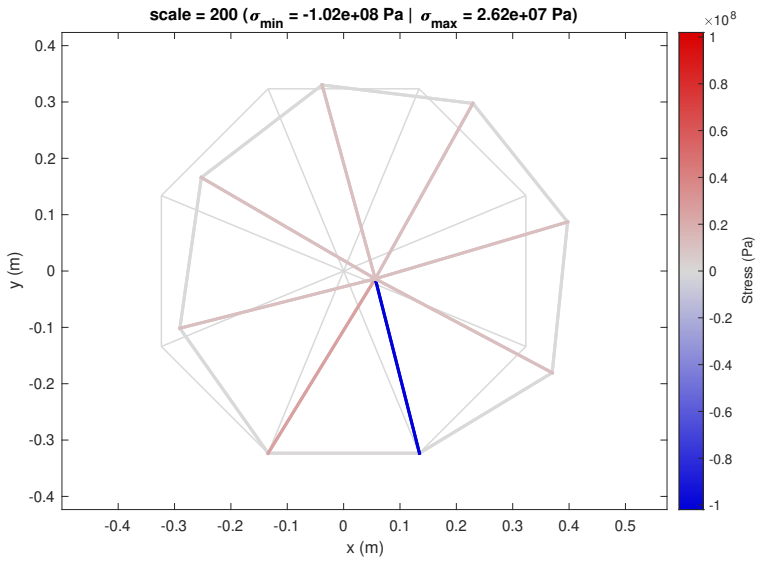

Figure 11: Diagrama, esfuerzos y desplazamientos, rueda delantera

Se puede observar en el gráfico la comparativa de los esfuerzos y desplazamientos experimentados por ambas ruedas en cada uno de los radios. La rueda trasera muestra cargas menores en comparación con la rueda delantera, así como un desplazamiento reducido. En la rueda trasera, los radios que hacen contacto con el suelo están comprimidos. En contraste, los radios de la rueda delantera en contacto con el suelo soportan cargas y deformaciones más significativas y no trabajan ambos a compresión, solamente el de la derecha.

Caso rueda con forma de **polígono de 20 lados** ($n = 20$):

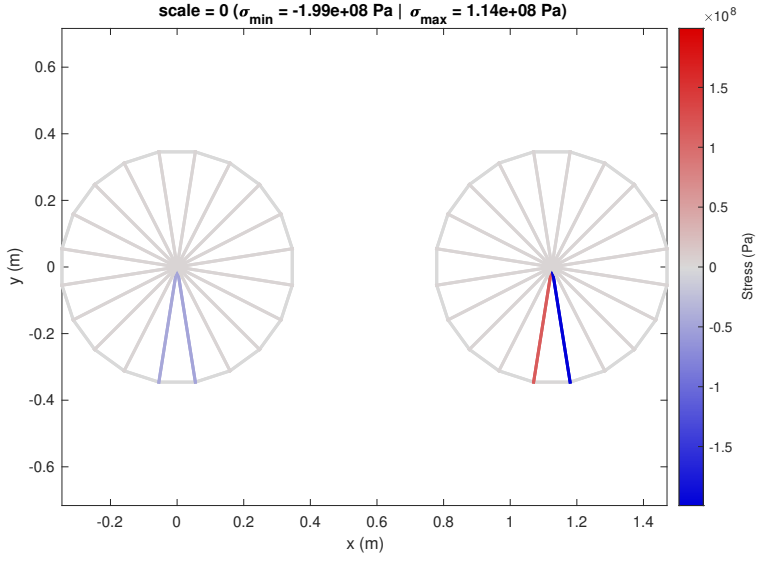

Figure 12: Diagrama, esfuerzos y desplazamientos, ruedas

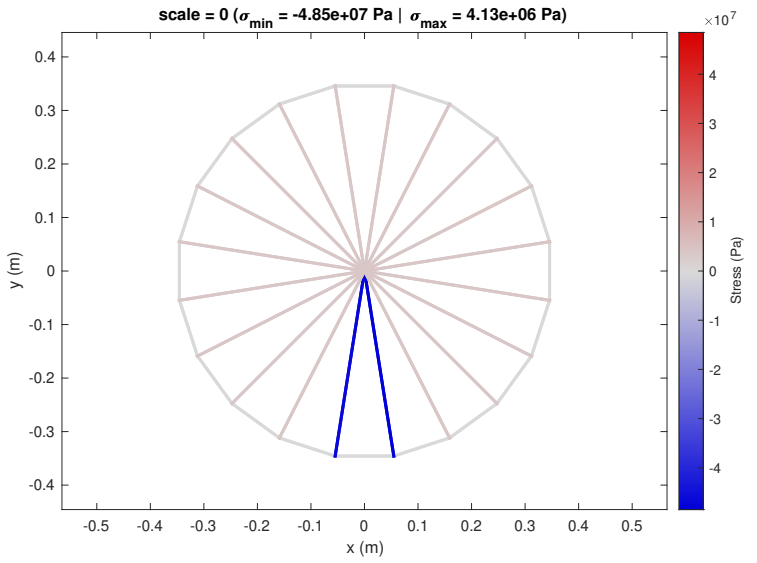

Figure 13: Diagrama, esfuerzos y desplazamientos, rueda trasera

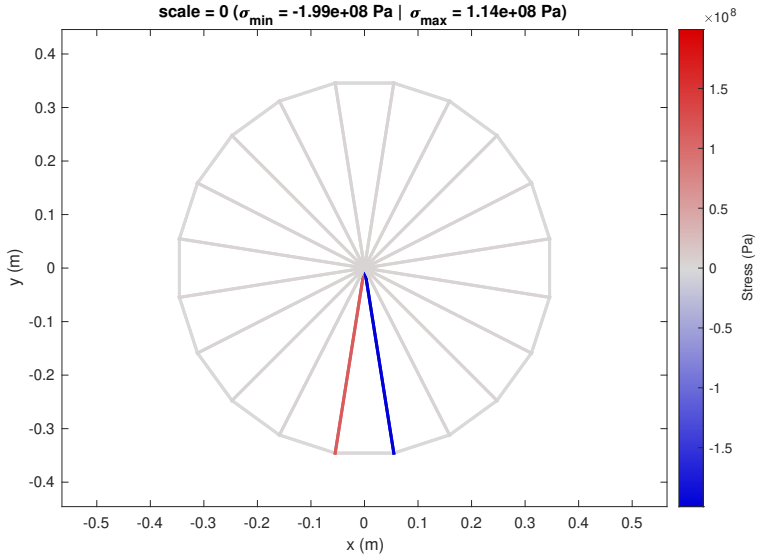

Figure 14: Diagrama, esfuerzos y desplazamientos, rueda delantera

En este caso, resulta interesante analizar el efecto de aumentar el número de radios, es decir, el número de lados n del polígono que representa el cuadro de la bicicleta. Es importante tener en cuenta que el análisis para un mayor número de radios no es completamente representativo, ya que, a pesar de aumentar el número de lados del polígono y, por ende, el número de barras que soportan el cuadro de la bicicleta, según esta modelización, se mantiene el mismo número de radios en contacto con el suelo, es decir, 2. Esto no refleja la realidad de manera precisa, ya que el arco de rueda en contacto con el suelo siempre es constante y teóricamente, al aumentar el número de radios, la carga se distribuiría de manera más uniforme, lo cual no se ve reflejado en este análisis. Para tener esto en cuenta, sería necesario establecer

en el programa unas condiciones de apoyo que incluyeran a todos los radios comprendidos entre el arco (ángulo) establecido para el modelado.

2.3 Pregunta C

Una vez realizado el diagrama y el análisis de esfuerzos en las distintas partes de la rueda, es necesario considerar el posible criterio de falla por pandeo, el cual está determinado por el esfuerzo crítico de Euler.

El pandeo es un fenómeno que ocurre en estructuras esbeltas sometidas a compresión axial, como pueden ser las barras de los radios de una rueda. Cuando una barra está comprimida axialmente, puede experimentar un desplazamiento lateral, conocido como pandeo, que puede llevar al colapso de la estructura si no se tiene en cuenta correctamente.

La ecuación proporcionada para el esfuerzo crítico (σ_{cr}) está relacionada con el pandeo porque representa el máximo esfuerzo que una barra puede soportar antes de pandearse. Si el esfuerzo aplicado a la barra supera este valor crítico, la barra puede pandearse y perder su capacidad de soportar la carga axial.

$$\sigma_{cr} = \frac{\pi^2 E}{l^2} \frac{I}{A}$$

Esta ecuación tiene en cuenta varios parámetros importantes para el pandeo. La longitud de la barra (l) y el área de la sección transversal (A) influyen en la estabilidad de la barra, ya que una barra más larga o con una sección transversal más pequeña será más propensa al pandeo. El segundo momento de área (I) es una medida de la distribución de masa alrededor del eje neutro de la sección transversal y afecta la rigidez de la barra frente al pandeo. En resumen, la ecuación proporciona una manera de evaluar el riesgo de pandeo de los radios de una rueda en función de sus propiedades geométricas y materiales.

En nuestro caso, al modelizar una rueda como un octágono, se tienen 8 radios, y se observa que el valor obtenido de, $\sigma_{cr} = 5.12\,\text{MPa}$ mientras que el valor de tensión de compresión máximo de los radios es de $42.95\,\text{MPa}$, el cual es superior al valor crítico. Por lo tanto, se puede afirmar que sufrirá pandeo.

2.4 Pregunta D

Para la obtención del valor de la pretensión inicial que garantiza un factor de seguridad de 2.5, hemos programado una función cuya imagen es cero cuando se satisface la condición mencionada. Dicha función se adjunta en el apéndice. El gráfico de la misma puede observarse en la figura 15.

Los siguientes valores se corresponden al valor de la **pretensión necesaria** para cada rueda con tal de garantizar un **factor de seguridad de exactamente 2.5** en cada una.

$$\sigma_{\text{trasera}} = 45.25\,\text{MPa} \qquad \sigma_{\text{delantera}} = 110.53\,\text{MPa}$$

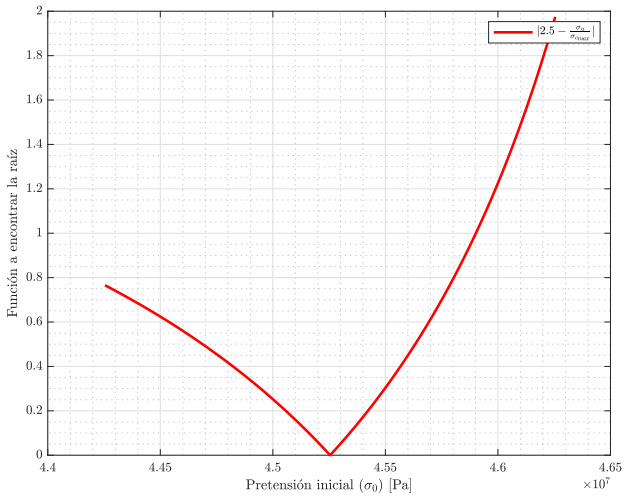

Figure 15: Diagrama, proceso iterativo, rueda trasera

Cuando la función vale cero se tiene el valor de la pretensión mínima necesaria para garantizar el factor de seguridad de al menos 2.5 ante la falla por pandeo.

El mismo procedimiento se ha seguido para determinar el valor de dicha tensión para la rueda delantera.

La siguiente figura muestra el estado de tensiones en la rueda trasera tras la aplicación de la pretensión en los radios. Como era de esperar, la tensión de los radios inferiores ya no se muestra azul. La tensión del radio sometido a mayor compresión, el cual es el radio inferior derecho, o el asociado al nodo 2, termina por ser de 2.05 MPa.

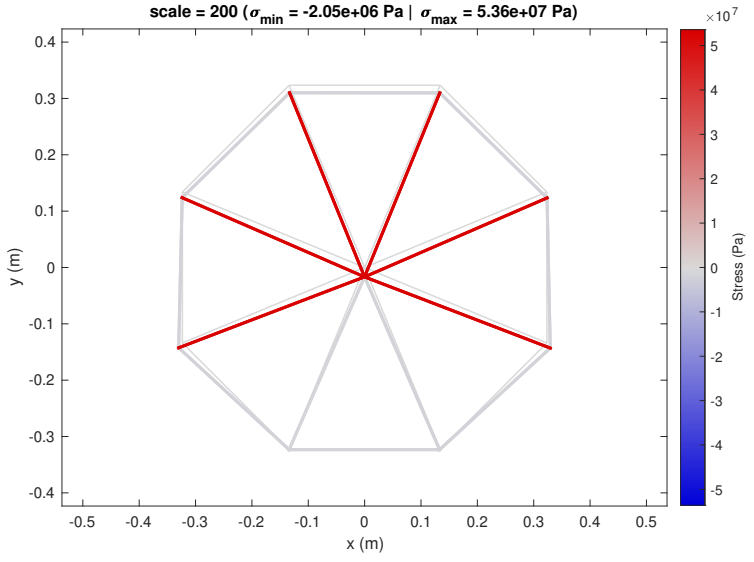

Figure 16: Diagrama, esfuerzos y desplazamiento, rueda trasera

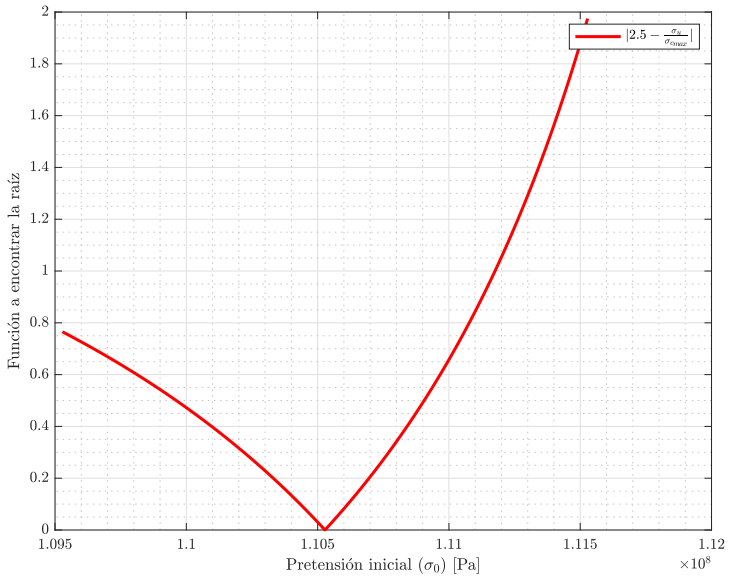

Figure 17: Diagrama, proceso iterativo, rueda delantera

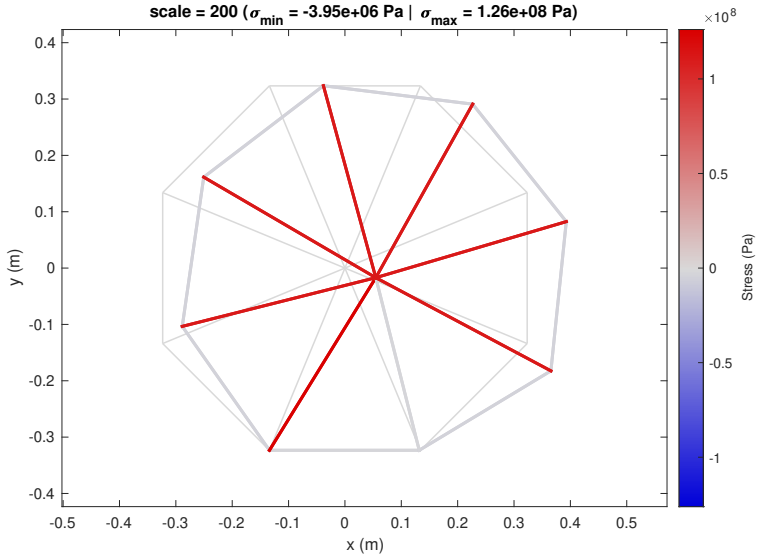

Figure 18: Diagrama, esfuerzos y desplazamiento, rueda delantera

Se puede observar como para la rueda delantera se ha seguido el mismo proceso, y se observa el mismo fenómeno.

3 Conclusiones

En primer lugar, se ha realizado el análisis estructural, obteniendo las tensiones y deformaciones, de una bicicleta, conocida su geometría y siendo esta modelada mediante vigas articuladas. La realización de este primer análisis ha tenido como objetivo determinar las reacciones que el cuerpo de la bicicleta transmitirá a los anclajes de las ruedas para, posteriormente, en el segundo análisis, poder realizar el estudio de los esfuerzos a los cuales estarán sometidos los radios de las mismas.

Con tal de determinar el estado tensional de los radios de las ruedas se ha implementado en MATLAB el método matricial. En última instancia, dicho análisis se ha realizado para poder establecer un nivel de pretensión necesario que garantice un factor de seguridad concreto ante la falla por pandeo de los radios de las ruedas de la bicicleta.

El resultado de dicho análisis ha permitido establecer, mediante el código programado, el valor de pretensión necesario para garantizar la no-falla por pandeo con el factor de seguridad deseado.

Concluimos, por lo tanto, que la práctica se ha realizado de forma satisfactoria, cumpliendo con los objetivos de la misma.

Más allá de los puntos solicitados a desarrollar, se ha implementado en el código la generalización del número de radios de las ruedas, ofreciendo así más versatilidad en el modelado de la geometría.

A Código empleado

A.1 Funciones

```
%% STRUCTURAL PROBLEM CODE STRUCTURE

clear
close all
format long

%% 1) PREPROCESS

% 1.1 Input data (define your input parameters here)
data.ni = 2;  % Degrees of freedom per node

% 1.2 Build geometry (mesh)
% Nodal coordinates matrix
x = [% column_1 = x-coord , column_2 = y-coord , ...
        0        0
    0.459   -0.054
    1.125        0
    0.315    0.486
    0.864    0.486
];
data.nnod = size(x,1); % Number of nodes
data.nd = size(x,2);   % Problem dimension
data.ndof = data.nnod*data.ni;  % Total number of degrees of freedom

% Nodal connectivities matrix
Tn = [% column_1 = element node 1 , column_2 = element node 2, ...
    1   2
    1   4
    2   4
    2   5
    3   5
    4   5
];
data.nel = size(Tn,1); % Number of elements
data.nne = size(Tn,2); % Number of nodes in a bar

% Create degrees of freedom connectivities matrix
Td = connectDOF(data,Tn);

% Material properties matrix
E=71e9;
A1=pi*(37.5^2-34.5^2)*1e-6/4;
A2=pi*(31.2^2-28.8^2)*1e-6/4;
A3=pi*(21^2-19^2)*1e-6/4;
m = [% Each column corresponds to a material property (area, Young's
   modulus, etc.)
    E    A1    0
    E    A2    0
    E    A3    0
];

```

```
% Material connectivities matrix
Tm = [% Each row is the material (row number in 'm') associated to each element
    3
    3
    2
    1
    2
    1
];


% 1.3 Input boundary conditions
% Fixed nodes matrix
p = [% Each row is a prescribed degree of freedom | column_1 = node, column_2 = direction, column_3 = value of prescribed displacement
    1   1   0
    1   2   0
    3   1   0
    3   2   0
];

% Point loads matrix
W = 75*9.8;
force=75*2.5; % N
F = [% Each row is a point force component | column_1 = node, column_2 = direction (1 = x-direction, 2 = y-direction), column_3 = force magnitude
    2   2   -.45*W
    4   2   -.5*W
    5   1   force
    5   2   -.05*W
];

%% 2) SOLVER

% 2.1.1 Compute element stiffness matrices
Kel = stiffnessFunction(data,x,Tn,m,Tm);

% 2.1.2 Compute element force vectors
fel = forceFunction(data,x,Tn,m,Tm);

% 2.2 Assemble global stiffness matrix
[K,f] = assemblyFunction(data,Td,Kel,fel);

% 2.3.1 Apply prescribed DOFs
[up,vp] = applyBC(data,p);

% 2.3.2 Apply point loads
f = pointLoads(data,f,F); %pointLoads(data,Td,f,F) Td???

% 2.4 Solve system
[u,r] = solveSystem(data,K,f,up,vp);

% 2.5 Compute stress
```

```
sig = stressFunction(data,x,Tn,m,Tm,Td,u);

%% 3) POSTPROCESS

scale = 400; % Set a number to visualize deformed structure properly
units = 'Pa'; % Define in which units you're providing the stress vector

plot2DBars(data,x,Tn,u,sig,scale,units);
```

```
%% STRUCTURAL PROBLEM CODE STRUCTURE

clear
close all
format long

%% 1) PREPROCESS

% 1.1 Input data (define your input parameters here)
data.ni = 2;  % Degrees of freedom per node

% 1.2 Build geometry (mesh)
% Nodal coordinates matrix
% column_1 = x-coord , column_2 = y-coord , ...
n = 8; % rodes com polígon regular de n costats
R = 35e-2; % [m]
d = 1.125; % [m]
x1 = polygonN(n,R);
x = [x1
    0 0];
%    x1(:,1)+d x1(:,2)
%    d 0];

data.nnod = size(x,1); % Number of nodes
data.nd = size(x,2);   % Problem dimension
data.ndof = data.nnod*data.ni;  % Total number of degrees of freedom

% Nodal connectivities matrix

% column_1 = element node 1 , column_2 = element node 2, ...
tn = connectPolygonN(n);
Tn = tn;
   %  tn+(n+1)];

data.nel = size(Tn,1); % Number of elements
data.nne = size(Tn,2); % Number of nodes in a bar

% Create degrees of freedom connectivities matrix
Td = connectDOF(data,Tn);

% Material properties matrix
E1=70e9; % [Pa]
E2=210e9; % [Pa]
A1=140e-6; % [m^2]
A2=3.8e-6; % [m^2]
I1 = 1420e-12; % [m^4]
I2 = 1.15e-12; % [m^4]
```

```
sigmaCrEuler = pi^2*E2*I2/(R^2*A2); % [Pa]
%sigma0=4.525452414038591e+07; %trasera pretensión
%sigma0 = 1.105280041599181e+08; %delantera pretensión
sigma0=0;
SF = 2.5;
sigmaAdm = sigmaCrEuler/SF; % [Pa]
m = [% Each column corresponds to a material property (area, Young's modulus, etc.) and inertia [m^4]
     E1     A1     0
     E2     A2     sigma0
];


% Material connectivities matrix
tm = ones(n,1);
Tm = [% Each row is the material (row number in 'm') associated to each element
 tm*1
 tm*2
% tm*1
% tm*2
];

% 1.3 Input boundary conditions
% Fixed nodes matrix
p = [% Each row is a prescribed degree of freedom | column_1 = node, column_2 = direction, column_3 = value of prescribed displacement
     1    1    0
     1    2    0
     2    2    0
    % (n+1)+1    1    0
    % (n+1)+1    2    0
    % (n+1)+2    2    0
];

% Point loads matrix
Rr = 1e2*[-0.011105555555555 3.879300000000002]; % [N]
Rf = 1e2*[-1.863894444444445 3.470700000000001]; % [N]
F = [% Each row is a point force component | column_1 = node, column_2 = direction (1 = x-direction, 2 = y-direction), column_3 = force magnitude
     n+1 1    -1*Rr(1)
     n+1 2    -1*Rr(2)
    % 2*(n+1) 1    -1*Rf(1)
    % 2*(n+1) 2    -1*Rf(2)
];

%% 2) SOLVER

% 2.1.1 Compute element stiffness matrices
Kel = stiffnessFunction(data,x,Tn,m,Tm);

% 2.1.2 Compute element force vectors
fel = forceFunction(data,x,Tn,m,Tm);

```

```
% 2.2 Assemble global stiffness matrix
[K,f] = assemblyFunction(data,Td,Kel,fel);

% 2.3.1 Apply prescribed DOFs
[up,vp] = applyBC(data,p);

% 2.3.2 Apply point loads
f = pointLoads(data,f,F); %pointLoads(data,Td,f,F) Td???

% 2.4 Solve system
[u,r] = solveSystem(data,K,f,up,vp);

% 2.5 Compute stress
sig = stressFunction(data,x,Tn,m,Tm,Td,u);

sigRad = sig((n+1):2*n);
%               (3*n+1):4*n]);

%% 3) POSTPROCESS

scale = 200; % Set a number to visualize deformed structure properly
units = 'Pa'; % Define in which units you're providing the stress vector

plot2DBars(data,x,Tn,u,sig,scale,units);
ylim([min(x(:,2))-0.1,max(x(:,2))+0.1]);

%%
% close all
% hold on;
% title('Numeración de nodos')
% %plot(x(:,1),x(:,2),'red');c
% scatter(x(:,1),x(:,2),'red','filled')
% % make data labels:
% xlim([min(x(:,1))-0.1,max(x(:,1))+0.1]);
% text(x(:,1),x(:,2),sprintfc(' %d',1:numel(x(:,1))))
% axis equal;
```

```
clear; close all;

E2=210e9; % [Pa]
A2=3.8e-6; % [m^2]
I2 = 1.15e-12; % [m^4]
R = 0.35;
sigmaCrEuler = pi^2*E2*I2/(R^2*A2); % [Pa]
SF = 2.5;
sigmaAdm = sigmaCrEuler/SF; % [Pa]


% Configución de las opciones de optimización
tolerance = 1e-3;
options = optimset('Display','on','TolFun', tolerance);

% Punto inicial
initial_guess = 1e7;

func = @(x) abs(2.5 - sigmaCrEuler/getSigmaCompressionMax(x));
```

```
20
21 % Encuentra la raíz con fminunc o fminsearch
22 sigma0Good = fminsearch(func, initial_guess, options)
23 SFactual = sigmaCrEuler/getSigmaCompressionMax(sigma0Good)
24
25 %% Per visulitzar la funció a trobar el mínim o l'arrel
26 %set(0, 'DefaultTextInterpreter', 'latex');
27
28 xx=linspace(sigma0Good-1e6,sigma0Good+1e6,4000);
29 plot(xx,arrayfun(func,xx),'-r','LineWidth',2)
30 xlabel('Pretensi\''on inicial ($\sigma_0)$ [Pa]','Interpreter','latex')
31 ylabel('Funci\''on a encontrar la ra\''iz','Interpreter','latex')
32 box on;
33 set(gca,'TickLabelInterpreter','latex','XLim',xlim*1,'YLim',ylim*1);
34 legend('$|2.5 - \frac{\sigma_{u}}{\sigma_{c_{max}}}|$','interpreter','
   latex')
35 grid on; grid minor;
```

```
1 function [up,vp] = applyBC(data,p)
2     node_to_dof = @(ni,node,direction) ni*(node-1) + direction;
3     up = p(:,3);
4     vp = node_to_dof(data.ni,p(:,1),p(:,2));
5 end
```

```
1 function [K,f] = assemblyFunction(data,Td,Kel,fel)
2     K = zeros(data.ni*data.nnod);
3     f = zeros(data.ni*data.nnod,1);
4     for e=1:data.nel
5         K(Td(e,:),Td(e,:)) = K(Td(e,:),Td(e,:)) + Kel(:,:,e);
6         f(Td(e,:),1) = f(Td(e,:),1) + fel(:,e);
7     end
8 end
```

```
1 function Td = connectDOF(data,Tn)
2     nrows = data.nel;
3     ncols = data.nne*data.ni;
4     Td = zeros(nrows,ncols);
5
6     for i=1:nrows
7         for j=1:ncols
8             Td(i,j) = data.ni*(Tn(i,mod(j,data.nne)+fix(j/data.nne))-1)
   +mod(j,data.ni+1)+fix(j/(data.ni+1));
9         end
10    end
11 end
```

```
1 function tn = connectPolygonN(n)
2     tn = zeros(2*n,2);
3     for i=1:(n-1)
4         tn(i,:) = [i i+1];
5         tn(i+n,:) = [n+1 i];
6     end
7     tn(n,:) = [n 1];
8     tn(2*n,:) = [n+1 n];
9 end
```

```
function fel = forceFunction(data,x,Tn,m,Tm)

    fel = zeros(data.nne*data.ni,data.nel);
    for e=1:data.nel
        xel = [x(Tn(e,:),:)];
        Ael = m(Tm(e),2);
        sigma0el = m(Tm(e),3);

        l = sqrt((xel(2,1)-xel(1,1))^2+(xel(2,2)-xel(1,2))^2);
        c = (xel(2,1)-xel(1,1))/l;
        s = (xel(2,2)-xel(1,2))/l;
        R = [
            c s 0 0
            -s c 0 0
            0 0 c s
            0 0 -s c];

        fel(:,e) = -sigma0el*Ael*R.'*[-1 0 1 0]';
    end
end
```

```
function sigmaCompressionMax = getSigmaCompressionMax(sigma0)
    %% 1) PREPROCESS

    % 1.1 Input data (define your input parameters here)
    data.ni = 2;  % Degrees of freedom per node

    % 1.2 Build geometry (mesh)
    % Nodal coordinates matrix
    % column_1 = x-coord , column_2 = y-coord , ...
    n = 8; % rodes com polígon regular de n costats
    R = 35e-2; % [m]
    d = 1.125; % [m]
    x1 = polygonN(n,R);
    x = [x1
        0 0];
    %   x1(:,1)+d x1(:,2)
    %    d 0];

    data.nnod = size(x,1); % Number of nodes
    data.nd = size(x,2);   % Problem dimension
    data.ndof = data.nnod*data.ni;  % Total number of degrees of freedom

    % Nodal connectivities matrix

    % column_1 = element node 1 , column_2 = element node 2, ...
    tn = connectPolygonN(n);
    Tn = tn;
   %      tn+(n+1)];

    data.nel = size(Tn,1); % Number of elements
    data.nne = size(Tn,2); % Number of nodes in a bar

    % Create degrees of freedom connectivities matrix
    Td = connectDOF(data,Tn);

```

```
% Material properties matrix
E1=70e9; % [Pa]
E2=210e9; % [Pa]
A1=140e-6; % [m^2]
A2=3.8e-6; % [m^2]
I1 = 1420e-12; % [m^4]
I2 = 1.15e-12; % [m^4]
sigmaCrEuler = pi^2*E2*I2/(R^2*A2); % [Pa]
SF = 2.5;
sigmaAdm = sigmaCrEuler/SF; % [Pa]
m = [% Each column corresponds to a material property (area, Young's
 modulus, etc.) and inertia [m^4]
    E1    A1    0
    E2    A2    sigma0
];


% Material connectivities matrix
tm = ones(n,1);
Tm = [% Each row is the material (row number in 'm') associated to
each element
  tm*1
  tm*2
%  tm*1
%  tm*2
];

% 1.3 Input boundary conditions
% Fixed nodes matrix
p = [% Each row is a prescribed degree of freedom | column_1 = node,
 column_2 = direction, column_3 = value of prescribed displacement
    1   1   0
    1   2   0
    2   2   0
  %  (n+1)+1   1   0
  %  (n+1)+1   2   0
  %  (n+1)+2   2   0
];

% Point loads matrix
Rr = 1e2*[-0.011105555555555 3.879300000000002]; % [N]
Rf = 1e2*[-1.863894444444445 3.470700000000001]; % [N]
F = [% Each row is a point force component | column_1 = node,
column_2 = direction (1 = x-direction, 2 = y-direction), column_3 =
force magnitude
    n+1 1   -1*Rr(1)
    n+1 2   -1*Rr(2)
   % 2*(n+1) 1   -1*Rf(1)
   % 2*(n+1) 2   -1*Rf(2)
];

%% 2) SOLVER

% 2.1.1 Compute element stiffness matrices
Kel = stiffnessFunction(data,x,Tn,m,Tm);
```

```
86
87      % 2.1.2 Compute element force vectors
88      fel = forceFunction(data,x,Tn,m,Tm);
89
90      % 2.2 Assemble global stiffness matrix
91      [K,f] = assemblyFunction(data,Td,Kel,fel);
92
93      % 2.3.1 Apply prescribed DOFs
94      [up,vp] = applyBC(data,p);
95
96      % 2.3.2 Apply point loads
97      f = pointLoads(data,f,F); %pointLoads(data,Td,f,F) Td???
98
99      % 2.4 Solve system
100     [u,r] = solveSystem(data,K,f,up,vp);
101
102     % 2.5 Compute stress
103     sig = stressFunction(data,x,Tn,m,Tm,Td,u);
104
105     sigRad = sig((n+1):2*n)
106                 %%(3*n+1):4*n]);
107     sigmaCompressionMax = abs(min(sigRad));
```

```
1  function plot2DBars(data,x,Tn,u,sig,scale,units)
2
3  % Precomputations
4  X = x(:,1);
5  Y = x(:,2);
6  Ux = scale*u(1:data.ni:end,1);
7  Uy = scale*u(2:data.ni:end,1);
8  smin = min(sig);
9  smax = max(sig);
10
11 % Open plot window
12 figure; box on; hold on; axis equal;
13 % Plot undeformed structure
14 plot(X(Tn'),Y(Tn'),'-', ...
15     'Color',0.85*[1,1,1],'LineWidth',1);
16 % Plot deformed structure with colorbar for stresses
17 patch(X(Tn')+Ux(Tn'),Y(Tn')+Uy(Tn'),[sig';sig'], ...
18     'EdgeColor','interp','LineWidth',2);
19 % Colorbar settings
20 clims = get(gca,'clim');
21 clim(max(abs(clims))*[-1,1]);
22 n = 128; % Number of rows
23 c1 = 2/3; % Blue
24 c2 = 0; % Red
25 s = 0.85; % Saturation
26 c = hsv2rgb([c1*ones(1,n),c2*ones(1,n);1:-1/(n-1):0,1/n:1/n:1;s*ones
       (1,2*n)]');
27 colormap(c);
28 cb = colorbar;
29
30 % Add labels
31 title(sprintf('scale = %g (\\sigma_{min} = %.3g %s | \\sigma_{max} = %.3
       g %s)',scale,smin,units,smax,units));
```

```
xlabel('x (m)');
ylabel('y (m)');
cb.Label.String = sprintf('Stress (%s)',units);

end
```

```
function f = pointLoads(data,f,F)
    node_to_dof = @(ni,node,direction) ni*(node-1) + direction;
    Fext = zeros(size(f));
    Fext(node_to_dof(data.ni,F(:,1),F(:,2))) = F(:,3);
    f = f + Fext;
end
```

```
function x = polygonN(n,R)
    x = zeros(n,2);
    a = (360/n);
    for i=1:n
        cordx = R*cosd(a*(i-1) + a/2 - (90 + a));
        cordy = R*sind(a*(i-1) + a/2 - (90 + a));
        x(i,:) = [cordx cordy];
    end
end
```

```
function [u,r] = solveSystem(data,K,f,up,vp)
    u = zeros(data.ndof,1);
    r = zeros(data.ndof,1);

    u(vp) = up;
    vf = setdiff((1:data.ndof)',vp);

    u(vf) = K(vf,vf)\(f(vf)-K(vf,vp)*up);
    r(vp) = K(vp,:)*(u) - f(vp);

end
```

```
function Kel = stiffnessFunction(data,x,Tn,m,Tm)

    Kel = zeros(4,4,data.nel);

    for e=1:data.nel
    xel = [x(Tn(e,:),:)];
    Eel = m(Tm(e),1);
    Ael = m(Tm(e),2);

    l = sqrt((xel(2,1)-xel(1,1))^2+(xel(2,2)-xel(1,2))^2);
    c = (xel(2,1)-xel(1,1))/l;
    s = (xel(2,2)-xel(1,2))/l;

    Kel(:,:,e) = (Eel*Ael/l)*[
        c^2 c*s -c^2 -c*s
        c*s s^2 -c*s -s^2
        -c^2 -c*s c^2 c*s
        -c*s -s^2 c*s s^2
        ];
    end

```

```
end
```

```
function sig = stressFunction(data,x,Tn,m,Tm,Td,u)
sig = zeros(data.nel,1);

for e=1:data.nel
    xel = [x(Tn(e,:),:)];
    Eel = m(Tm(e),1);
    sigmael0 = m(Tm(e),3);
    l = sqrt((xel(2,1)-xel(1,1))^2+(xel(2,2)-xel(1,2))^2);
    c = (xel(2,1)-xel(1,1))/l;
    s = (xel(2,2)-xel(1,2))/l;
    R = [c s 0 0
         -s c 0 0
         0 0 c s
         0 0 -s c];
    uel = u(Td(e,:));
    sig(e) = (Eel/l)*[-1 0 1 0]*R*uel + sigmael0;
end
```