industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

# UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

# Desarrollo y prueba de una aplicación informática para el estudio de secciones en régimen plástico

**TRABAJO FIN DE GRADO**

GRADO EN INGENIERÍA MECÁNICA

**Autor:** **Alicia Alfonso Martínez**
Director: Manuel Santiago Torrano Martínez

Cartagena, 30 de Mayo del 2018

# Resumen

El presente trabajo tiene como objetivo el diseño e implementación de una aplicación informática que permite caracterizar el comportamiento de una sección de un elemento prismático sometido a flexión pura o compuesta en diferentes regímenes de trabajo para pequeñas deformaciones.

Para ello se hace uso de las hipótesis de la Resistencia de Materiales, con excepción de la hipótesis de comportamiento del material elástico y lineal. Ya que al considerar los regímenes de trabajo elastoplástico y plástico de la sección, las tensiones y deformaciones dejan de ser proporcionales. De manera que el problema a nivel de sección tiene un carácter no lineal. De modo que se ha implementado en lenguaje Matlab las funciones necesarias para resolver los problemas de tensión - deformación a nivel de sección. Debido al carácter no lineal, ha sido necesario implementar esquemas iterativos para poder hacer cumplir las relaciones entre esfuerzos, tensiones y deformaciones que se dan en la sección. Para lo cual se han utilizado métodos numéricos conocidos de resolución de ecuaciones no lineales.

En consecuencia, el programa permite caracterizar el comportamiento de la sección en los regímenes elástico, elastoplástico y plástico mediante los diagramas de Interacción Axil - Momento y Momento - Curvatura, y también, resolver problemas de carácter práctico de estructuras isostáticas.

# Índice general

# Capítulo 1

# Introducción

El presente trabajo tiene como objetivo el diseño e implementación de una aplicación informática que permite caracterizar el comportamiento de una sección de un elemento prismático sometido a flexión pura o compuesta en diferentes regímenes de trabajo para pequeñas deformaciones.

Para cumplir este objetivo se sigue el enfoque del libro de 'Plasticidad Abreviada' de Rafael Fernández Díaz-Munío [3]. Donde se siguen considerando las hipótesis simplificativas de la Resistencia de Materiales para elementos prismáticos pero añadiéndole una complejidad al problema al considerar el comportamiento no lineal del material. De esta manera, las tensiones y deformaciones dejan de ser proporcionales y por tanto, deja de ser válido aplicar el Principio de Superposición.

Esta aplicación se ocupa de los problemas a nivel de sección. Por lo que haciendo uso de las expresiones de la Resistencia de Materiales relacionadas con las hipótesis de pequeños movimientos y deformaciones de elementos prismáticos (hipótesis de Navier), teniendo en cuenta la equivalencia estática entre tensiones y esfuerzos, y por último, considerando la relación entre tensiones y deformaciones que ofrecen las leyes constitutivas del material; se ha diseñado el programa y su implementación está escrita en el lenguaje Matlab.

El contenido de los capítulos del trabajo es el que sigue: el Capítulo 2, presenta el marco teórico que ha servido de base para la implementación del programa. En el Capítulo 3 se describe el programa de manera que quedan definidas las tareas de las que se ocupa, así como el enfoque bajo el cual se ha diseñado. En el Capítulo 4 se describe cómo han sido implementadas las funciones del programa. El Capítulo 5 muestra ejemplos de cómo puede ser utilizado el programa. En el capítulo 6 se presentan las conclusiones y se propone una serie de problemas con el que se podría avanzar en el estudio de secciones en régimen elastoplástico y plástico. El Apéndice A ofrece la información necesaria para poder utilizar el programa a nivel de usuario. Finalmente, el Apéndice B muestra el código del programa escrito en Matlab.

# Capítulo 2

# Fundamentos Teóricos

## 2.1. Planteamiento del problema

Se tiene una pieza prismática que está caracterizada por tener la dimensión longitudinal significativamente mayor a las dimensiones de la sección transversal.

Por el Principio de Saint Venant sabemos que *en una pieza prismática las tensiones (y deformaciones) que actúan sobre una sección recta alejada de los puntos de aplicación de un sistema de cargas, sólo dependen de la fuerza* $\overrightarrow{R}$ *y del momento resultante* $\overrightarrow{M}$ *de las fuerzas situadas a un lado de la sección.*

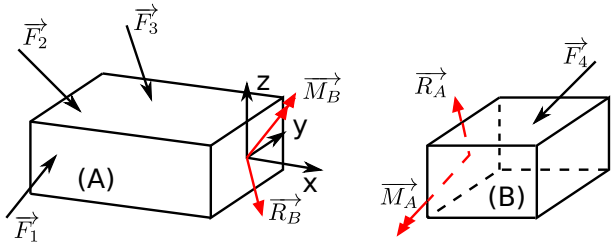



Figura 2.1: Elemento prismático solicitado

$\overrightarrow{R}$ y $\overrightarrow{M}$ son las componentes de fuerza aplicadas en el centro de gravedad de una sección. En el caso de la Figura 2.1, $\overrightarrow{R_A}$ y $\overrightarrow{M_A}$ representan las resultantes de las fuerzas que la parte B ejerce sobre A. Además, para que exista equilibrio en la pieza, estos esfuerzos han de cumplir: $\overrightarrow{R_A} = -\overrightarrow{R}_B$ y $\overrightarrow{M}_A = -\overrightarrow{M}_B$.

La pieza prismática es un modelo de material continuo en el que se desarrolla un sistema de fuerzas internas continuas, $d\overrightarrow{f}$, que contrarrestan las fuerzas externas. Se define como tensión ($\overrightarrow{t}$) en un punto a la fuerza por unidad de área de la forma: $\overrightarrow{t} = \frac{d\overrightarrow{f}}{dS}$. El vector tensión en un punto del sólido, cuando éste se corta imaginariamente con una superficie $\Omega$, es el vector de fuerzas por unidad de superficie que el resto del cuerpo realiza sobre este punto y superficie. Por tanto, los esfuerzos se relacionan con las tensiones por medio de las expresiones:

$\overrightarrow{R} = \int\int_\Omega \overrightarrow{t}\, dS$

$\overrightarrow{M} = \int\int_\Omega \overrightarrow{r} \times \overrightarrow{t}\, dS$

Se considera el sistema de ejes cartesiano XYZ con origen en el centro de gravedad de la sección. Los ejes $y$ y $z$ son ejes principales de la sección, y el eje $x$ es tangente a la directriz de la pieza. Las componentes de los esfuerzos respecto al sistema de referencia son:

$\overrightarrow{R} = N\,\hat{i} + V_y\,\hat{j} + V_z\,\hat{k}$

$\overrightarrow{M} = M_x\,\hat{i} + M_y\,\hat{j} + M_z\,\hat{k}$

las componentes de tensión respecto al sistema de referencia son:

$\overrightarrow{t} = \sigma_x\,\hat{i} + \tau_{xy}\,\hat{j} + \tau_{xz}\,\hat{k}$

y el vector posición es de la forma: $\overrightarrow{r} = (0, y, z)$

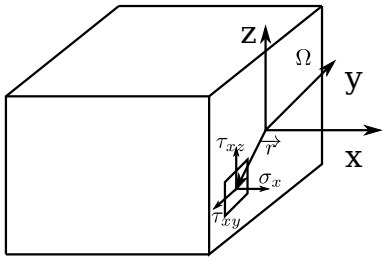



Figura 2.2: Componentes de tensión en un prisma mecánico

De este modo, las relaciones entre las diferentes componentes de tensión y esfuerzo son:

$N = \int\int_\Omega \sigma_x\,dS$; $V_y = \int\int_\Omega \tau_{xy}\,dS$; $V_z = \int\int_\Omega \tau_{xz}\,dS$;

$M_x = \int\int_\Omega (\tau_{xz}\,y - \tau_{xy}\,z)\,dS$; $M_y = -\int\int_\Omega \sigma_x\,z\,dS$; $M_z = \int\int_\Omega \sigma_x\,y\,dS$

Si además, se considera el plano XZ como plano de carga, las resultantes de las fuerzas estarán contenidas en este plano. Si se considera flexión pura y compuesta, los esfuerzos que intervienen son:

$\overrightarrow{R} = N\,\hat{i}$;$\overrightarrow{M} = M_y\,\hat{j}$; Ambos tipos de esfuerzo solo producen tensiones normales, $\sigma_x$.

Para poder determinar la distribución de tensiones a lo largo de la sección es necesario establecer hipótesis simplificativas relativas a la deformación de la pieza. Ya que para saber cómo resiste un elemento, necesitamos información sobre cómo se deforma ante diferentes tipos de solicitación. Puesto que nos encontramos en el rango de pequeñas deformaciones y movimientos, se considera la hipótesis de Navier que establece que *las caras permanecen planas tras la deformación.*

Si el elemento prismático está sometido a flexión pura, el momento es constante, de forma que la directriz de la viga se deformará conforme a un arco de circunferencia. Por tanto, las fibras longitudinales formarán arcos concéntricos con la directriz de la pieza. No aparecen distorsiones como se muestra en la Figura 2.3. Así, tanto las deformaciones angulares como las tensiones tangenciales son nulas.

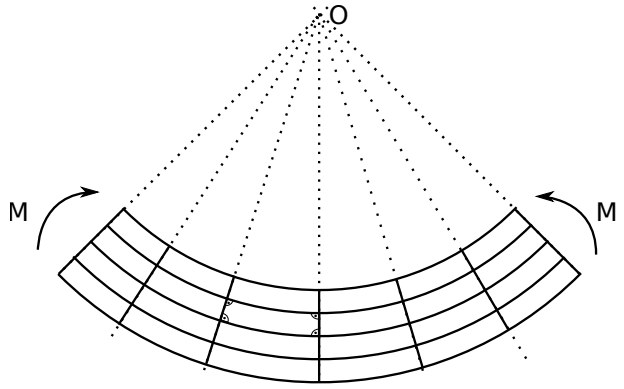



Figura 2.3: Deformación de un elemento prismático por flexión pura.

Para determinar la expresión que caracteriza la deformación por flexión pura consideramos una rebanada de longitud $dx$ de la pieza prismática como se muestra en la Figura 2.4.

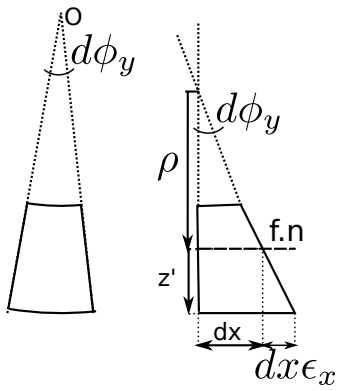



Figura 2.4: Deformación de la rebanada por flexión

Al tratarse de pequeñas deformaciones, el giro relativo entre dos caras de la rebanada ($d\phi_y$) es pequeño, consecuentemente se pueden asociar arcos con tangentes. Así, haciendo semejanza de triángulos obtenemos la relación cinemática que caracteriza a la deformación:

$\frac{\rho}{dx} = \frac{-z}{\epsilon_x\, dx}$; Por definición la curvatura ($\chi$) es la inversa del radio de curvatura: $\chi = \frac{1}{\rho}$.

$\epsilon_x = -\chi\, z$ (Hipótesis de Navier de deformación por flexión pura)

De la expresión se puede deducir que conforme nos alejamos de la posición de la fibra neutra las fibras experimentan deformaciones mayores y proporcionales a esta distancia.

Si la flexión es compuesta, la componente del axil no es nulo. Un axil produce deformaciones longitudinales uniformes a lo largo de la sección en el que está aplicado para así cumplir con la hipótesis de Navier. Por tanto, sumando las contribuciones del axil y del momento queda una deformada con el aspecto de la Figura 2.5.

Podemos concluir que tanto en flexión pura como compuesta la deformación se puede ver como un giro de la sección alrededor de un eje. Sobre este eje, llamado fibra neutra, la ley tanto de deformaciones como de tensiones se hace nula. De este modo, la deformación de la sección en flexión pura y compuesta queda determinada por la posición de la fibra neutra y la curvatura.

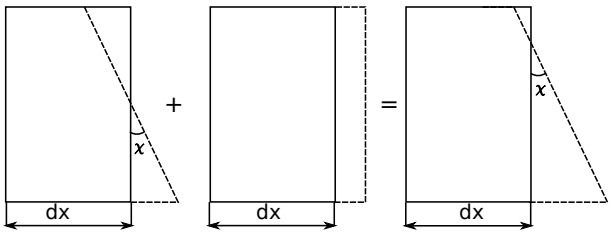



Figura 2.5: Hipótesis de Navier para flexión compuesta

Para poder completar las expresiones que relacionan las cantidades que determinan el comportamiento resistente de una sección, se necesita conocer la relación entre tensiones y deformaciones. Esta relación es característica del material, la cual se obtiene de ensayos mecánicos. Sea por ejemplo el ensayo de tracción en el que una probeta es sometida a esfuerzos de tracción y cuyos resultados se muestran en la Figura 2.6.

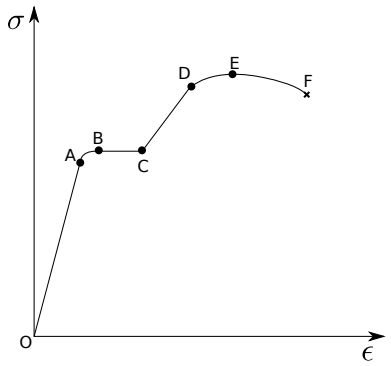



Figura 2.6: Curva tensión - deformación de un ensayo a tracción

Si se presta atención al diagrama, se pueden distinguir diferentes relaciones entre tensión y deformación: elástico lineal (tramo OA), elástico no lineal (tramo AB), fluencia (tramo BC), endurecimiento (CE), reblandecimiento (EF) y rotura (F). Donde un comportamiento elástico está caracterizado por presentar deformaciones reversibles y un comportamiento plástico por presentar irreversibilidades en las deformaciones. En consecuencia, si un elemento cargado solo presenta un comportamiento elástico, el estado de tensiones solo depende de ese instante y además, al ser descargado recupera su forma original. Por otro lado, un elemento que presenta un comportamiento plástico del material, al ser descargado parte de la deformación es reversible y otra parte no se recupera. Por este motivo al evaluar las tensiones de un elemento que presenta un comportamiento plástico se debe de contemplar la historia de las cargas.

Estas relaciones de tensión-deformación son las que nos interesan a la hora de modelar el comportamiento del material. Existen diferentes modelos ideales que simplifican los resultados reales de los ensayos. Sea por ejemplo en la Figura 2.7, donde se muestran 3 tipos de modelos de comportamiento del material [11]:

a) Perfectamente plástico con igual comportamiento a tracción y compresión: este modelo puede caracterizar materiales como el acero estructural, que presentan grandes deformaciones antes de la rotura y su capacidad resistente tanto a compresión como a tracción es similar.

b) Perfectamente plástico con diferente comportamiento a tracción y compresión: este modelo resulta interesante para modelar un material frágil como el hormigón, que presenta una capacidad resistente a compresión significativamente mayor que a tracción .

c) Material con endurecimiento: este modelo se utiliza para modelar el endurecimiento que presentan diferentes aceros aleados tras el escalón de fluencia, pudiendo cuantificar así la reserva de resistencia que ofrecen.

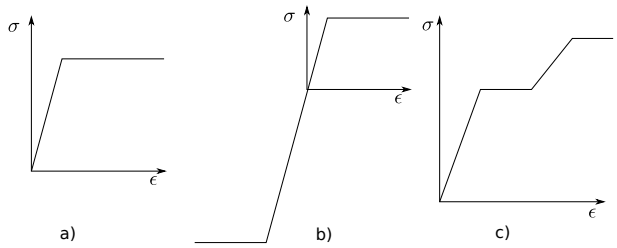



Figura 2.7: Diferentes modelos ideales de material

En resumen, la relación entre tensión, deformación y esfuerzos que caracteriza el comportamiento de una sección, queda determinada por las expresiones:

- Equivalencia estática entre tensiones y esfuerzos:
  $N = \int \int_\Omega \sigma(z)\, dS; \quad M_y = -\int \int_\Omega \sigma(z)\, z\, dS$
- Hipótesis de Navier para flexión pura y compuesta de deformaciones:

$\epsilon(z') = -\chi\, z'$

$z' = z - z_{fn}$

- Ley constitutiva del material:

$\sigma = \sigma(\epsilon)$

- Criterio de plastificación:

$\sigma_{ec} \leq \sigma \leq \sigma_{et}$

## 2.2. Plastificación de una sección sometida a flexión pura

En este apartado se verá cómo evoluciona una sección conforme aumenta el momento flector que la somete. Según el valor del momento flector, la sección puede trabajar en tres regímenes diferentes:

a) Régimen elástico, donde ninguna fibra ha alcanzado el valor del límite elástico de deformación y por tanto no presenta ninguna fibra plastificada. De manera que la relación entre tensión y deformación será proporcional en toda la sección.

b) Régimen elastoplástico, donde la relación entre tensión y deformación será proporcional solo en aquellas zonas donde no se aprecia plastificación. Otra característica que presenta este régimen es que las fibras que han plastificado no ofrecen mayor resistencia ante incrementos de carga, lo cual implica una redistribución de tensiones a lo largo de la sección.

c) Régimen plástico, es una situación ideal ya que se refiere al estado límite de la sección cuando todas su fibras han plastificado. De modo que presenta una curvatura infinita, lo que se traduce a deformaciones infinitas.

Conforme aumenta el momento flector, aumentan las deformaciones según la ley de Navier. Debido al comportamiento característico de cada uno de los regímenes, nos puede interesar conocer cómo se reparte las tensiones según aumentan los esfuerzos en la sección, para lo cual se requiere calcular los parámetros que caracterizan a la distribución de tensiones. Estos parámetros son la curvatura($\chi$) y la posición de la fibra neutra ($z_{fn}$).

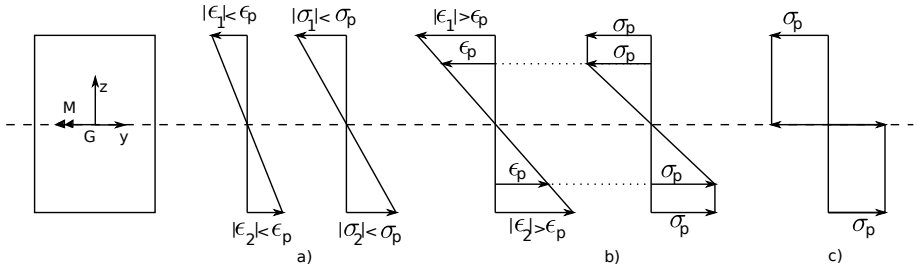



Figura 2.8: Plastificación de una sección sometida a flexión pura

- **Régimen elástico**

Como en este régimen el comportamiento del material es elástico y lineal, se pueden utilizar las expresiones:

$\sigma(z) = \frac{N}{A} - \frac{M_y}{I_y}\, z; \quad \epsilon(z) = \epsilon_g - \chi\, z; \quad \sigma = E\, \epsilon$

Al ser la flexión pura, N=0, la posición de la fibra neutra coincide con la fibra baricéntrica. De manera que las expresiones de la posición de la fibra neutra y curvatura quedan:

$z_{fn} = z_G = 0; \quad \chi = \frac{M_y}{E\, I_y}$

- **Régimen elastoplástico**

En el régimen elastoplástico, la relación entre la tensión y deformación deja de ser lineal. Por tanto, no se puede calcular los valores de posición de fibra neutra y curvatura mediante una expresión analítica. Estos valores tienen que ser tal que la distribución de tensiones sea estáticamente equivalente a la solicitación y ésta

ha de ser coherente con la deformación y la ley constitutiva del material. Lo que lleva a resolver un sistema de ecuaciones no lineales.

Solo en el caso de que la sección sea bisimétrica y con material perfectamente plástico con igual comportamiento a tracción y compresión y con una solicitación de flexión pura; debido a que la ley constitutiva es antisimétrica y la sección es simétrica, para que se pueda cumplir la condición de axil nulo, la fibra neutra tiene que coincidir con la baricéntrica.

En este régimen, la curvatura se calcula mediante la expresión: $\chi = -\frac{\epsilon_e}{z'_e}$. Donde $\epsilon_e$ es la deformación unitaria longitudinal en el límite elástico, y $z'_e$ la posición de la primera fibra plastificada en referencia a la posición de la fibra neutra.

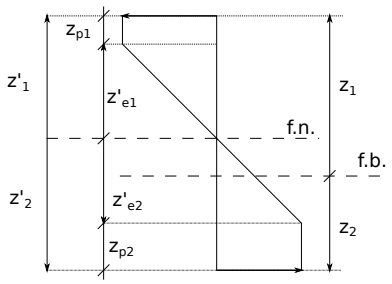



Figura 2.9: Diagrama de tensión normal en régimen elastoplástico

- **Régimen plástico**

  La distribución de tensiones cuando la sección está agotada por flexión pura es de la forma, Figura 2.11b:

  $$\sigma(z) = \begin{cases} \sigma_{p2} & z_2 \leq z \leq z_{fnp} \\ \sigma_{p1} & z_{fnp} \leq z \leq z1 \end{cases}$$

  se puede ver que la relación entre tensiones y solicitación queda:

  $N_p = \int \int_{\Omega} \sigma(z')\, dy'\, dz' = \sigma_{p1} \int \int_{\Omega_{p1}} dy'\, dz' + \sigma_{p2} \int \int_{\Omega_{p2}} dy'\, dz' = \sigma_{p1}\, A(\Omega_{p1}) + \sigma_{p2}\, A(\Omega_{p2});$

  Haciendo momentos respecto la posición de la fibra neutra:

  $M_p = N\, z'_G - \int \int_{\Omega} \sigma(z')\, z'\, dy'\, dz' = N\, z'_G - \sigma_{p1} \int \int_{\Omega_{p1}} z'\, dy'\, dz' - \sigma_{p2} \int \int_{\Omega_{p2}} z'\, dy'\, dz' = -\sigma_{p1}\, W(\Omega_{p1}) - \sigma_{p2} W(\sigma_{p2}) + N\, z'_G$

  Siendo $\Omega_{p1}$ y $\Omega_{p2}$, la porción de la sección que cae por encima y por debajo de la posición de la fibra neutra ($z_{fnp}$), respectivamente.

  Como en el régimen elastoplástico, para el caso en el que la sección sea bisimétrica con material perfectamente plástico e igual comportamiento a tracción y compresión, la posición de la fibra neutra debe de coincidir con la baricéntrica para que se cumpla la condición de axil nulo.

## 2.3. Caracterización de una sección sometida a flexión

Se puede ver del apartado anterior que la relación entre la flexión y la deformación tiene un carácter asintótico y esta relación es característica de la sección, depende del material que la compone y su geometría. Esta relación se puede visualizar por medio del diagrama Momento - Curvatura de la sección, Figura 2.10. Donde se puede apreciar el cambio de comportamiento que presenta la sección cuando empieza a presentar un comportamiento plástico del material; la curvatura deja de ser proporcional al momento y éste crece de forma asintótica conforme aumenta la deformación (curvatura). Por tanto, los valores característicos que se deducen del diagrama son: el momento elástico,

el momento plástico y la relación entre estos dos valores es el factor de forma. Este valor es característico de la sección y nos indica el margen que presenta ésta cuando empieza a presentar plastificación hasta el agotamiento. Aunque el momento plástico es un valor ideal, establece un valor límite de esfuerzo que puede ser soportado por la sección.

La combinación de estos valores límites en el caso de que la sección esté sometida a flexión compuesta se puede ver en un diagrama interacción Axil - Momento, Figura 2.11a. A su vez este diagrama encierra todas las combinaciones posibles axil - momento que puede soportar la sección. Para poder establecer el régimen para una combinación concreta, se pueden utilizar las expresiones del apartado anterior, con la diferencia de que en flexión compuesta el axil no es nulo.

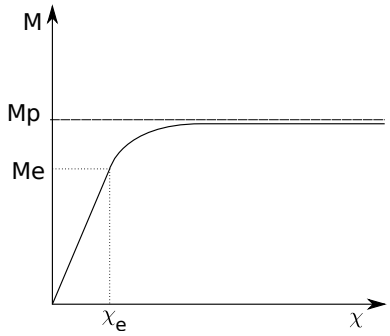
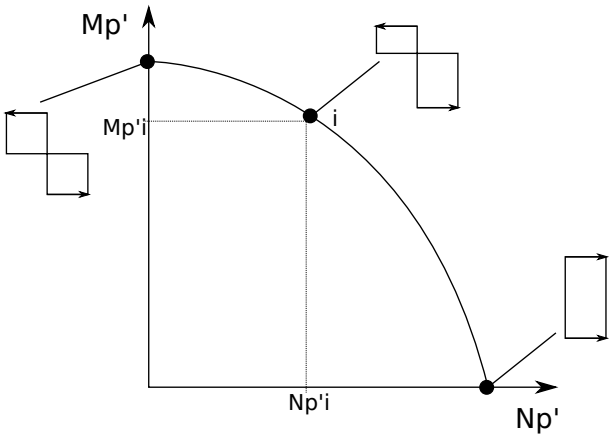
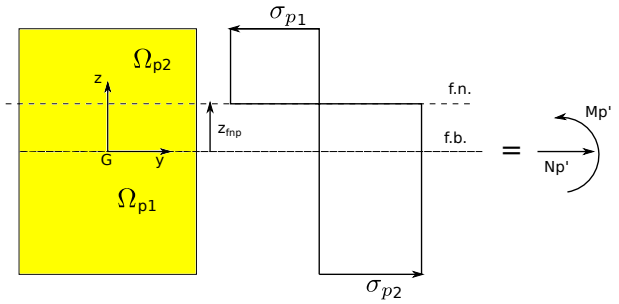



Figura 2.10: Diagrama Momento - Curvatura




(a) Diagrama de interacción Axil - Momento de una sección




(b) Diagrama de tensión de una sección agotada por flexión compuesta

Figura 2.11

# Capítulo 3

# Descripción del programa

## 3.1. Definición del problema

El programa ha de resolver las siguientes tareas:

1. A partir de la geometría y modelo de material de la sección, calcular el diagrama Momento - Curvatura.
2. A partir del carácter del axil (compresión/tracción), la geometría y modelo de material de la sección, calcular el diagrama de interacción Axil - Momento de la sección.
3. A partir de la solicitación, geometría y modelo de material de la sección, calcular la distribución de tensiones y deformaciones.

La ejecución del programa se llevará a cabo por medio de las siguientes fases:

1. Preproceso, donde se incluyen las tareas: escoger problema, definir geometría (a partir de la forma y dimensiones de la sección), definir el material (a partir del tipo de modelo del material y sus valores característicos) y definir la solicitación (si fuera necesario).
2. Análisis: el programa resuelve el programa en función de los parámetros determinados durante el preproceso.
3. Post-Proceso, donde se incluyen las tareas: visualización del modelo de entrada y presentación de los resultados de forma gráfica y numérica.

Por tanto, se requiere implementar funciones que se ocupen de todas las fases necesarias para la resolución de cada uno de los problemas. Por lo que el programa ha ser capaz de: recoger los datos introducidos por el usuario, generar el modelo, resolver el problema y presentar los resultados.

## 3.2. Diseño del programa

### 3.2.1. Estructura del programa

Se trata por separado las funciones de recogida y muestra de los resultados con las propias para la resolución de los problemas. Por tanto, el programa consta de dos módulos principales independientes, ver Figura 3.1:

*presentation/* : módulo principal que recoge las funciones propias de las fases de preproceso y postproceso.

*logic/*: módulo principal que recoge las funciones propias de la fase de resolución del problema.

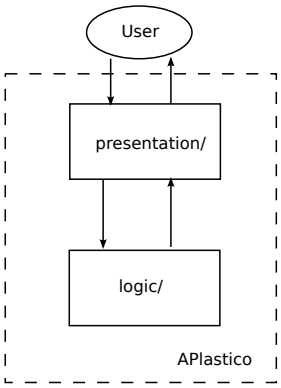



Figura 3.1: Enfoque del programa

Para la introducción de los datos se han considerado tres modos: por consola, por archivo, que permite correr varios casos automáticamente, y por interfaz gráfica de usuario, en la que el usuario puede introducir los datos sin la incomodidad de introducirlos de forma secuencial como ocurre con el modo por consola. Los resultados se muestran tanto en forma numérica como gráfica.

El módulo de análisis (*logic*) se ocupa de todas las tareas relacionadas con la resolución del problema. Desde la generación del modelo *(generacion_Modelo*) como el acondicionamiento de los datos tanto a la entrada como a la salida de los cálculos (*acondicionamientoDatos*) y los propios para resolver cada uno de los problemas de flexión (*problFlexion*). La resolución de los diferentes problemas se basa en hacer cumplir las condiciones de equivalencia estática entre esfuerzos y tensiones, la hipótesis de Navier de deformación y la ecuación constitutiva del material. Es por ello que se han implementado los siguientes módulos adicionales:

Para calcular los esfuerzos a partir de la distribución de tensiones se requiere resolver integrales dobles. Como la distribución de tensiones tanto en región elástica como elastoplástica y plástica se puede descomponer en regiones donde presenta un mismo comportamiento respecto a la deformación, esto supone integrales que pueden ser resueltas por medio de las propiedades geométricas (área, momento estático,...) de dichas regiones[9].

Descomponer la sección en diferentes regiones es un problema geométrico. Por lo que se ha implementado un módulo para resolver problemas geométricos. Es por esto que se han implementado los módulos de *problGeom*, para la resolución de problemas geométricos, y *propSeccion* para el cálculo de las propiedades tanto geométricas como del material de la sección.

Para hacer cumplir las condiciones se requiere un esquema iterativo ya que tanto en régimen elastoplástico como plástico supone resolver ecuaciones no lineales. Para ello se han utilizado métodos numéricos, es por esto que se ha implementado un módulo de métodos numéricos *(metNum*).

La Figura 3.2 muestra la estructura del programa.

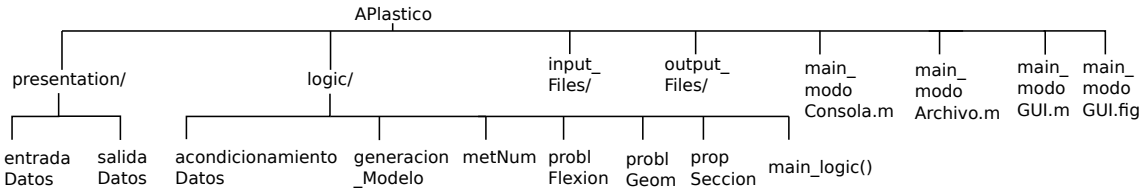



Figura 3.2: Estructura del programa

De donde podemos distinguir las diferentes carpetas y módulos siguientes:

*presentation/*: módulo principal que recoge las funciones propias de las fases de preproceso y postproceso.

*entradaDatos*: módulo que contiene las funciones para la recogida y almacenamiento de datos.

*salidaDatos*: módulo que contiene las funciones para la muestra de resultados.

*logic/*: módulo principal que recoge las funciones propias de la fase de resolución del problema.

*main_logic()*: función que resuelve cada uno de los problemas.

*acondicionamientoDatos*: módulo que contiene las funciones para preparar los datos a la entrada de la resolución del problema y para la presentación de los resultados.

*generacion_Modelo*: módulo que recoge las funciones para la generación del modelo.

*metNum*: módulo para la resolución de problemas de Métodos Numéricos (resolución de ecuaciones no lineales).

*problFlexion*: módulo que recoge las funciones para la resolución de problemas de flexión en régimen elasto-plástico.

*problGeom*: módulo que recoge las funciones que calculan los problemas geométricos.

*propSeccion*: módulo que recoge las funciones para la generación de las propiedades de la sección.

*input_Files/*: carpeta que contiene los archivos de entrada para la resolución del problema en modo archivo.

*output_Files/*: carpeta que contiene los archivos de salida en caso de elegir guardar los resultados.

*main_modoConsola.m*: script de arranque del programa en modo consola.

*main_modoArchivo.m*: script de arranque del programa en modo archivo.

*main_modoGUI.m*: script de arranque del programa en modo interfaz gráfica de usuario (GUI). *main_modoGUI.fig:* archivo imagen que contiene la interfaz del programa.

### 3.2.2. Diseño de los datos

En la Figura3.3, se puede ver el flujo de datos del programa de donde se distinguen los siguientes datos:

*tipoProblema:* tipo de problema (integer). <1: calcular diagrama Momento - Curvatura | 2: calcular diagrama interacción axil - momento | 3: calcular la distribución de la tensión a partir de la solicitación>

*datEntrada:* datos de entrada, incluye la información necesaria para resolver el problema escogido (estructura cell).

*datEntrada{1}*: *tipoSecc*, tipo de sección (string). <'R': sección rectangular | 'C': sección circular | 'I': sección doble T | 'Inb': sección doble T no bisimétrica | 'Rombo' | 'T': sección en T | 'H': sección en H | 'cajon': sección en cajón | 'Rh': secciń rectangular con hueco | 'Ch': sección circular con hueco | 'IPN' | 'IPE'>

*datEntrada{2}*: *dimSecc*, dimensiones de la sección (array). Valores depende del tipo de sección.

*datEntrada{3}*: *tipoMat*, tipo de modelo de material (string). <'G1': material perfectamente plástico con igual comportamiento a tracción y compresión | 'G2': material perfectamente plástico con diferente comportamiento a tracción y compresión | 'G3': material con endurecimiento e igual comportamiento a tracción y compresión >

*datEntrada{4}*: *valMat*, propiedades características del modelo del material (array). Valores dependen del tipo de modelo del material.

Para problema 2, cálculo del diagrama $N'_p - M'_p$ :

datEntrada{5}: *signoN*, indica si el axil del diagrama interacción axil - momento es de tracción o compresión (string). <'p': axil de tracción | 'n': axil de compresión>

Para el problema 3, cálculo de la distribución de tensiones y deformaciones dada una solicitación:

datEntrada{5}: $N$, valor del axil (double).

datEntrada{6}: $M$, valor del momento (double).

*valEntrada:* valores a la entrada de la resolución del problema (estructura cell). Son generados a partir de *datEntrada.*

valEntrada{1}: $P$, vértices del polígono que define a la sección (array).

valEntrada{2}: *ley_mat*, ley constitutiva del material (array).

*valSalida:* resultados, valores a la salida de la resolución del problema (estrucura cell).

Para el problema 1, cálculo del diagrama $M - \chi$ :

*valSalida{1}:* $M_e$, momento elástico (double)

*valSalida{2}:* $\chi_e$, curvatura correspondiente al momento elástico (double)

*valSalida{3}:* $M_p$, momento plástico (double)

*valSalida{4}:* $\lambda$, factor de forma de la sección (double)

*valSalida{5}: signo_zp*, carácter de la profundidad de plastificación, compresión/tracción (string). <'c': plastificación de compresión | 't': plastificación de tracción>

*valSalida{6}: condic*, desde qué extremo (superior/inferior) de la sección se mide la profundidad de plastificación de referencia (integer). <1: extremo superior | 2: extremo inferior>

*valSalida{7}*: *d_MC*, diagrama momeno - curvatura (array).

*d_MC(1)*: *zp*, profundidad de plastificación (double); *d_MC(2)*: $\chi_i$ (double), curvaturas; *d_MC(3)*: momento asociado a $\chi_i$, $M_i$ (double).

Para problema 2, cálculo del diagrama $N'_p - M'_p$ :

*valSalida{1}: d_int,* diagrama interacción axil - momento (array).

*d_int(1)*: $Np_i$, valores de axil de agotamiento (double); *d_int(2)*: $Mp_i$, valores de momento de agotamiento (double).

*valSalida{2}:* $N_p$, axil puro de agotamiento (double).

*valSalida{3}:* $M_p$, momento de agotamiento en flexión pura (double).

Para el problema 3, cálculo de la distribución de tensiones y deformaciones dada una solicitación:

*valSalida{1}:* $z'$, posiciones de los puntos característicos de la distribución de tensiones referidos a la posición de la fibra neutra (array).

*valSalida{2}:* $sigz'$, valores de tensión de la distribución de tensiones (array).

*valSalida{3}:* $epsz'$, valores de deformación de la distribución de deformaciones (array).

*valSalida{4}:* $\chi$, curvatura de la distribución de deformaciones

*valSalida{5}:* $Ap_1$, polígono del área plastificada superior (array).

*valSalida{6}:* $Ap_2$, polígono del área plastificada inferior (array).

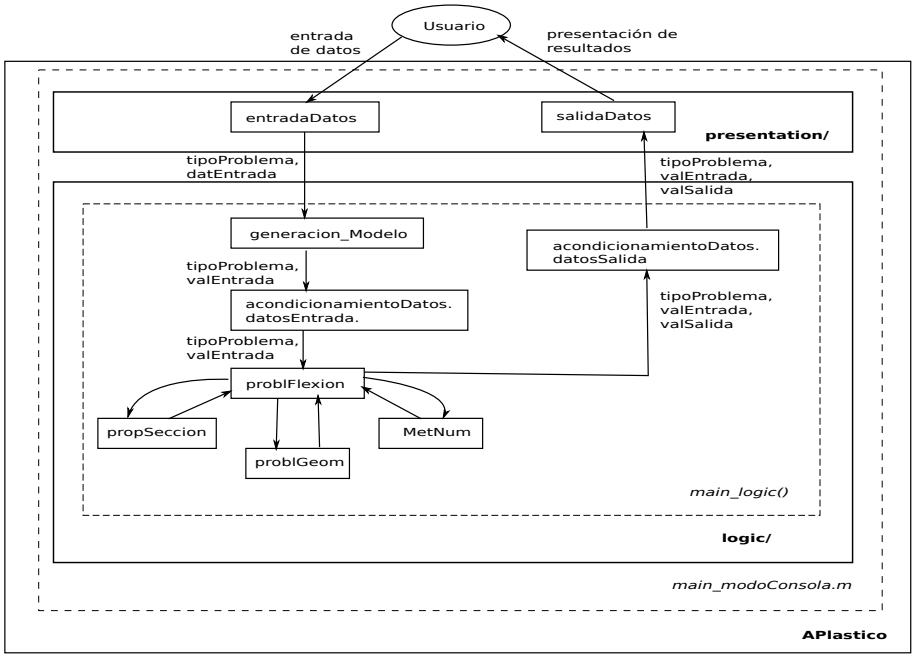



Figura 3.3: Diagrama de flujo del programa

# Capítulo 4

# Implementación

Este capítulo ofrece una visión de cómo han sido implementadas las diferentes funciones y módulos del programa. Para ver las funciones escritas en Matlab, ver el Apéndice B.

## 4.1. Scripts de arranque del programa

El programa se puede ejecutar de tres modos, según como quiera el usuario interaccionar con el programa.

Los tres modos siguen el mismo esquema, solo varía ligeramente de uno a otro debido a las diferencias a la hora de introducir los datos. El esquema que sigue cada uno de los modos es:

1. Entrada de datos
2. Resolución del problema
3. Salida de datos

### 4.1.1. *main_modoConsola.m*

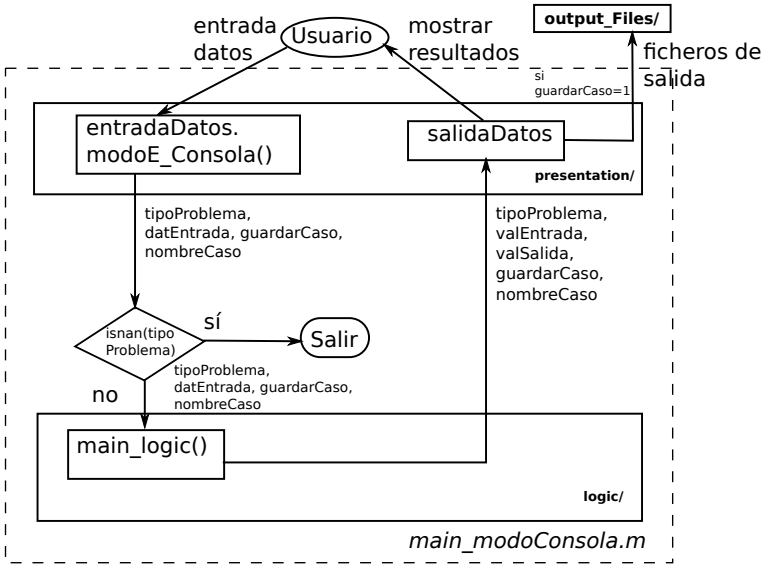



Figura 4.1: *main_modoConsola.m*

### 4.1.2. *main_modoArchivo.m*

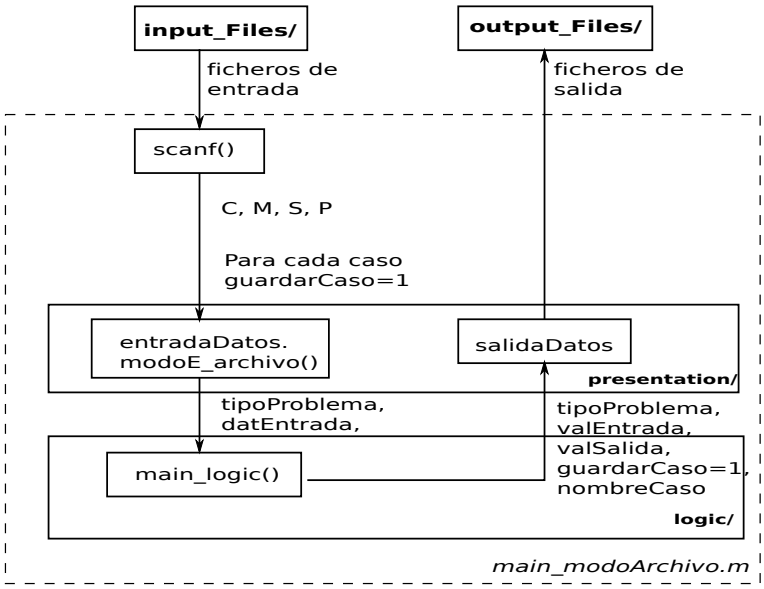



Figura 4.2: *main_modoArchivo.m*

### 4.1.3. *main_modoGUI.m*

Ver B.1.3.

## 4.2. Módulo de preproceso y post-proceso

*presentation/:* módulo que recoge las funciones relacionadas con la recogida de datos (preproceso) y la generación y muestra de resultados(postproceso). Por lo que está dividida en los submódulos: *entradaDatos* que corresponde a la fase de preproceso y *salidaDatos* que corresponde a la fase de postproceso.

### 4.2.1. Módulo de preproceso

El módulo *entradaDatos* se encarga de recoger los datos bien directamente del usuario por medio del modo consola o GUI, o bien desde los archivos que se encuentran en la carpeta *Input_Files/*. Estos datos son: el tipo de sección, dimensiones de la sección, tipo de modelo de material, valores del material, tipo de problema y valores propios del problema. Esta información es almacenada en la variable *datEntrada.*

Para la recogida de los datos por medio de consola está la función *entradaDatos.modoE_consola()* y para el modo por archivo, *entradaDatos.modoE_archivo().*

Para el modo entrada de datos por archivo se necesitan los archivos:

*input_casos.csv* de la que se genera la estructura (cell) C que recoge los casos que se quieren llevar a cabo.

*input_problemas.csv* de la que se genera la estructura (cell) P que recoge la definición de los problemas.

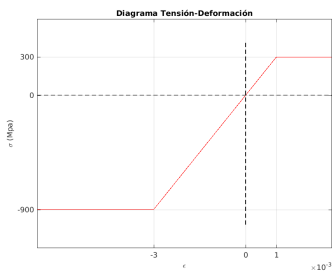
*input_materiales.csv* de la que se genera la estructura (cell) M que recoge la definición de los diferentes materiales.

*input_secciones.csv* de la que se genera la estructura (cell) S que recoge la definición de las diferentes secciones.

### 4.2.2. Módulo de post-proceso

El módulo *salidaDatos* se encarga de mostrar el modelo de entrada y los resultados del problema al usuario tanto en forma gráfica como numérica.

- Para la muestra del modelo de entrada de forma gráfica están las funciones: *mostrar_ley_material()* para visualizar la ley constitutiva del material y *mostrar_perfilConEjes()* para mostrar la geometría de la sección.




(a) Salida *mostrar_ley_material()* (b) Salida *mostrar_perfilConEjes()*

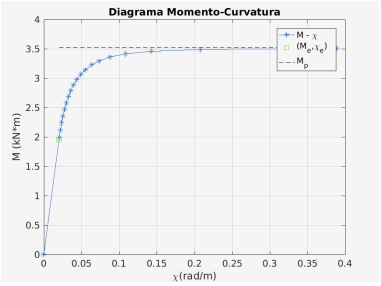
Figura 4.3: Mostrar modelo de forma gráfica.

- Para la muestra de los resultados de forma gráfica están las funciones según el tipo de problema:

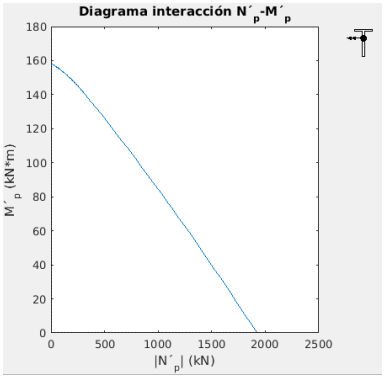
a) Para el problema 1, se muestra el diagrama momento - curvatura mediante *mostrar_diagrama_MC()*.

b) Para el problema 2, se muestra el diagrama de interacción axil - momento mediante la función *mostrar_diagr_interaccion()*.

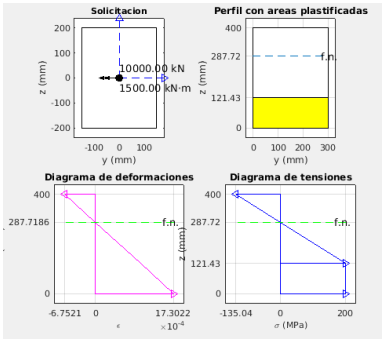
c) Para el problema 3, se muestra el perfil con las áreas plastificadas mediante *mostrar_areasPlastificadas()*, la distribución de tensiones por medio de *mostrar_ley_tensiones()* y la distribución de deformaciones mediante *mostrar_ley_deformaciones()*. Adicionalmente mediante la función *mostrar_solicitacion()* se muestra la solicitación que es estáticamente equivalente a la distribución de tensiones calculada.




(a) Mostrar diagrama $M - \chi$




(b) Mostrar diagrama $N'_p - M'_p$




(c) Muestra de resultados problema 3, solicitación $\sigma(z)$, $\epsilon(z)$ y áreas plastificadas

Figura 4.4: Mostrar resultados de forma gráfica

- Para la representación de los resultados numéricamente están las funciones según sea el tipo de problema:

a) Para el problema 1, *informe_problemaMC()* , donde se muestran los valores numéricos de las dimensiones, valores del material y una tabla con los valores numéricos del diagrama momento - curvatura.

b) Para el problema 2, *informe_diagrama_interacc()* , donde se muestran los valores numéricos de las dimensiones, valores del material y una tabla con los valores numéricos del diagrama interacción axil - momento.

c) Para el problema 3, *informe_leyTensiones()* , donde se muestran los valores numéricos de las dimensiones, valores del material y dos tablas: una con los puntos característicos de la distribución de tensiones y otra con los puntos característicos de la distribución de deformaciones.

Las unidades en la salida de los resultados son: tensiones (MPa), distancias (mm), curvaturas (rad/m).

## 4.3. Módulo de análisis

### 4.3.1. *main_logic.m*

La función *main_logic()* gestiona las tareas necesarias para la resolución de cada uno de los problemas, estas tareas son:

1. Generación del modelo:
   a) *generacion_Modelo.verticesPoly()* para la generación del modelo geométrico de la sección.
   b) *generacion_Modelo.comportMaterial()* para la generación del modelo del material del que está compuesta la sección.
2. Acondicionamiento de los datos a la entrada de la resolución del problema, *acondicionamientoDatos.datosEntrada*.
3. Resolución del problema. Depende del tipo de problema:
   a) Si se trata del problema 1, donde se caracteriza el comportamiento de la sección cuando está sometida por flexión pura: calcula el momento plástico ($M_p$) por medio de *problFlexion.N_Mp()*, el momento elástico ($M_e$) y su curvatura ($\chi_e$) mediante *problFlexion.N_Me()* y el diagrama momento - curvatura ($d\_MC$) mediante la función *problFlexion.N_diagramaMC()*. El factor de forma ($\lambda$) se obtiene dividiendo el momento plástico entre el momento elástico.
   b) Si se trata del problema 2, donde se busca calcular el diagrama interacción axil - momento de la sección ($d\_int$), este diagrama se calcula mediante la función *problFlexion.diagrama_interaccionNM()*.
   c) Si se trata del problema 3, donde se calcula el diagrama de tensiones ($\sigma(z)$) y deformaciones ($\epsilon(z)$) asociado a una solicitación definida por su axil y momento ($N$,$M$). Para ello se calculan los parámetros que determinan la distribución de deformaciones, posición de la fibra neutra respecto el sistema global de coordenadas ($z_{fn}$) y la curvatura ($\chi$), mediante la función *problFlexion.NAndM_zfnAndChi()*. Y con estos parámetros se calcula la distribución de tensiones y de deformación mediante la funciones:
      1) si $z_{fn} = \infty$, mediante la función *problFlexion.EpszAndSigz_AxilPuro*,
      2) si $z_{fn} \neq \infty$ mediante la función *problFlexion.zfnAndChi_EpszAndSigz()*.
4. Acondicionamiento de los datos a la salida de la resolución del problema, *acondicionamientoDatos.datosSalida*.

El modelo de entrada se guarda en la variable *valEntrada* y los resultados son almacenados en la variable *valSalida*.

### 4.3.2. Generación del modelo

A partir del tipo y dimensiones características de la sección, la función *generacion_Modelo.verticesPoly()* genera los vértices del polígono que define el contorno de la sección.

Figura 4.5: Secciones que genera el programa

El polígono está formado por una sucesión de vértices ordenados de manera que dos puntos consecutivos forman una arista. El polígono es almacenado en un array cuyo vector fila i son las coordenadas del punto $p_i$, $[y_i, z_i]$. Sea por ejemplo el polígono generado para una sección rectangular a partir de sus dimensiones: ancho (b) y canto(h):

$$P = \begin{bmatrix} 0 & 0 \\ 0 & b \\ b & h \\ 0 & h \end{bmatrix}$$

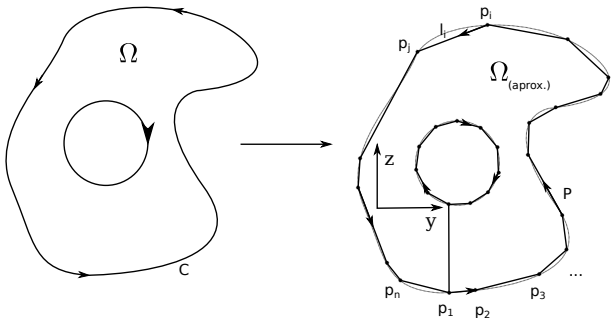



Figura 4.6: Polígono que representa una sección cualquiera

A partir del tipo y valores característicos del material, la función *generacion_Modelo.comportMaterial()* genera la ley constitutiva del material, $\sigma = \sigma(\epsilon)$. Ésta es almacenada en un array cuyo vector fila i es $[\epsilon_i, \sigma_i]$. Siendo por ejemplo para el material perfectamente plástico con igual comportamiento a tracción y compresión:

$$ley\_mat = \begin{bmatrix} -100 & -\sigma_e \\ -\epsilon_e & -\sigma_e \\ 0 & 0 \\ \epsilon_e & \sigma_e \\ 100 & \sigma_e \end{bmatrix}$$

Donde el valor 100 es un valor suficientemente grande de deformación unitaria para ser considerado infinito.

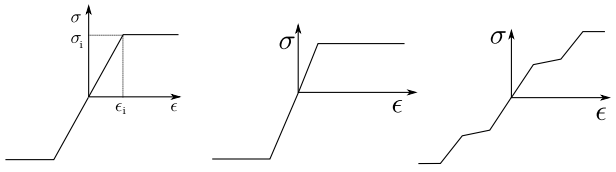



Figura 4.7: Modelos de material que genera el programa

### 4.3.3. Acondicionamiento de los datos

Las unidades empleadas en el programa vienen recogidas en las siguientes tablas:

| Magnitud | Unidad |
|---|---|
| Distancia | $mm$ |
| Fuerza | $kN$ |
| Momento | $kN \cdot m$ |
| Tensión | $MPa$ |
| Curvatura | $rad/m$ |

(a) Unidades en la Entrada/-Salida de resultados

| Magnitud | Unidad |
|---|---|
| Distancia | $m$ |
| Fuerza | $kN$ |
| Momento | $kN \cdot m$ |
| Tensión | $kPa$ |
| Curvatura | $rad/m$ |

(b) Unidades en los cálculos

Las funciones *acondicionamientoDatos.datosEntrada() y acondicionamientoDatos.datosSalida()* se encargan de asegurar los cambios de unidades para la entrada de datos en los cálculos y la muestra de los resultados, respectivamente.

El sistema de referencia que el programa espera, para la resolución del problema, es el sistema global ($OYZ$), donde $y$ y $z$ son ejes principales de inercia de la sección y su origen se encuentra en el centro de gravedad de ésta. Puesto que para definir los puntos del polinomio puede no ser práctico definirlos en referencia a su centro de gravedad, la función *acondicionamientoDatos.datosEntrada()* refiere los puntos a su centro de gravedad.

### 4.3.4. Problemas de flexión

*problFlexion*: módulo que resuelve los problemas relacionados con el comportamiento de secciones sometidas a flexión pura y compuesta con pequeñas deformaciones. Para lo cual interesará conocer la relación entre tensiones, deformaciones y esfuerzos de la sección.

La sección queda definida por su geometría y el material que la constituye. Por tanto, todos los cálculos de este módulo requieren conocer la ley constitutiva del material (*ley_mat*) y el polígono que define el contorno de la sección (P).

$$ley\_mat = \begin{bmatrix} \epsilon_1 & \sigma_1 \\ \epsilon_2 & \sigma_2 \\ ... & ... \\ \epsilon_n & \sigma_n \end{bmatrix}; \; P = \begin{bmatrix} y_1 & z_1 \\ y_2 & z_2 \\ ... & ... \\ y_n & z_n \end{bmatrix}.$$

De modo que la ley constitutiva del material queda de la forma:

$\sigma = \sigma_{i-1} + \frac{\sigma_i - \sigma_{i-1}}{\epsilon_i - \epsilon_{i-1}} (\epsilon - \epsilon_{i-1})$ si $\epsilon_{i-1} < \epsilon \leq \epsilon_i$

$i = 2, ..., n$; n: número de puntos característicos de la ley constitutiva.

En el caso de la Figura 4.8, n=5.

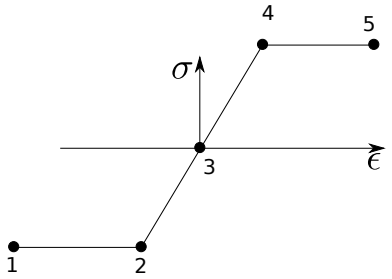



Figura 4.8: Ley constitutiva tramos lineales

#### Planteamiento del problema

Se asume:

- Sección forma parte de un elemento prismático.

- Tipos de solicitación de la sección: flexión pura y compuesta.

- Plano de carga: plano XZ.

- Ejes z e y son ejes principales de inercia de la sección.

- Hipótesis de pequeños movimientos y deformaciones.

- El comportamiento elástico del material se considera lineal. El comportamiento del material en su etapa final hasta la rotura se considera perfectamente plástico.

Relación entre tensiones, deformaciones y esfuerzos:

- Equivalencia estática entre tensiones y esfuerzos:

$N = \int \int_\Omega \sigma(z)\, dS; \quad M_y = -\int \int_\Omega \sigma(z)\, z\, dS$

- Hipótesis de Navier para flexión pura y compuesta de deformaciones:

$\epsilon(z') = -\chi\, z'$

$z' = z - z_{fn}$

- Ley constitutiva del material:

$\sigma = \sigma(\epsilon)$

**Convenio de signos**

Se considera:

- Momento positivo aquel que produce compresiones en el extremo superior y tracciones en el extremo inferior de la sección.

- Axil positivo aquel que produce tracciones.

- Deformación unitaria longitudinal positiva si produce elongaciones.

- Tensión normal positiva, si la tensión es de tracción.

**Sistemas de referencia y posiciones relevantes**

En los cálculos se consideran dos sistemas de referencia, $OYZ$ y $O'Y'Z'$.

Siendo:

$OYZ$ : sistema de ejes global. Tiene su origen en el centro de gravedad y los ejes $y$,$z$ son ejes principales de inercia de la sección. El eje y coincide con la fibra baricéntrica (f.b.).

$O'Y'Z'$ : sistema de ejes local. El eje $y$ es paralelo a $y$'. El eje $y$' coincide con el eje neutro (f.n.). Está relacionado con el sistema global de la forma:

$$\begin{cases} y' = y \\ z' = z - z_{fn} \end{cases}$$

**Posiciones relevantes que se consideran en el sistema de referencia global ($OYZ$):**

$y = 0;\ z = 0$ : posición del centro de gravedad de la sección.

$z = z_{f.n.}$ : posición de la fibra neutra.

$z_1$ : posición del extremo superior de la sección.

$z_2$ : posición del extremo inferior de la sección.

**Posiciones relevantes que se consideran en el sistema de referencia local ($O'Y'Z'$):**

$z' = 0$, posición de la fibra neutra.

$z'_{e1}$ : posición de la primera fibra plastificada de la sección que cae por encima de la fibra neutra.

$z'_{e2}$ : posición de la primera fibra plastificada de la sección que cae por debajo de la fibra neutra.

$z'_1$ : posición del extremo superior de la sección.

$z'_2$ : posición del extremo inferior de la sección.

$z = z'_G$ : posición de la fibra baricéntrica.

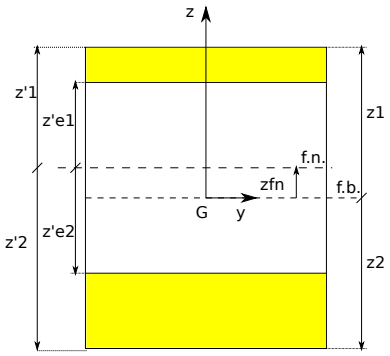



Figura 4.9: Posiciones relevantes en los sistemas $OYZ$ y $O'Y'Z'$

En el caso de un material con endurecimiento, pueden existir posiciones intermedias a considerar entre $[z'_{e1},\ z'_1]$ y $[z'_2,\ z'_{e2}]$.

#### 4.3.4.1. Cálculo de tensiones y deformaciones

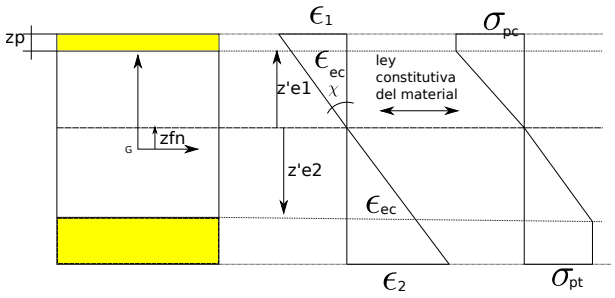



Figura 4.10: Distribución de tensiones y deformaciones en régimen elastoplástico

Un diagrama de deformaciones puede quedar determinado por los parámetros de posición de fibra neutra ($z_{fn}$) y curvatura ($\chi$). También se podría determinar conociendo la profundidad de plastificación $\boldsymbol{z_p}$ y la posición de la fibra neutra ($z_{fn}$), ya que la relación de la profundidad de plastificación($\boldsymbol{zp}$) y curvatura es:

$\chi = \frac{-\epsilon_e}{z'_e}$

donde,

Si la profundidad de plastificación está medida desde el extremo superior de la sección $\rightarrow z'_e = z'_{e1} = z'_1 - zp$

Si la profundidad de plastificación está medida desde el extremo inferior de la sección $\rightarrow z'_e = z'_{e2} = z'_2 + zp$

Además,

Si la zona plastificada considerada con $\boldsymbol{zp}$ es de compresión $\rightarrow \epsilon_e = \epsilon_{ec}$.

Si la zona plastificada considerada con $\boldsymbol{zp}$ es de tracción $\rightarrow \epsilon_e = \epsilon_{et}$.

De esto último, podemos concluir que para especificar una profundidad de plastificación ($\boldsymbol{zp}$) necesitamos una distancia (zp), especificar desde que extremo de la sección se mide la distancia (superior/inferior) y cual es el carácter de la zona plastificada (compresión/tracción), ver Figura 4.10.

Las tensiones se calculan por medio de la deformación($\epsilon$) y la ley constitutiva del material (*ley_mat*) mediante la función *sigma()*, que sigue el siguiente procedimiento:

i. Obtener los valores $\epsilon_{i-1}$, $\sigma_{i-1}$ y $\epsilon_i$ (ver Figura 4.11) a partir de la ley constitutiva, de manera que $\epsilon_{i-1} < \epsilon \leq \epsilon_i$, i=2,...,n; n: número total de puntos que definen la ley constitutiva.

ii. Calcular la tensión por medio de la expresión:

$\sigma = \sigma_{i-1} + \frac{\sigma_i - \sigma_{i-1}}{\epsilon_i - \epsilon_{i-1}} (\epsilon - \epsilon_{i-1})$

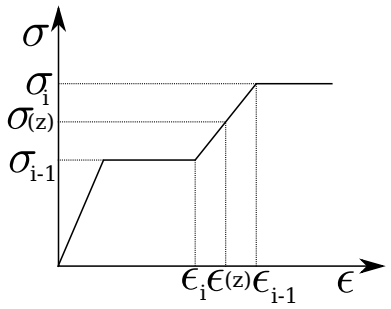



Figura 4.11: Cálculo de la tensión de una fibra

- Calcular la distribución de tensiones y deformaciones a partir de los parámetros $z_{fn}$ y $\chi$:

  - Si la solicitación es flexión ($\chi \neq 0$) , la distribución de deformaciones y tensiones se calcula por medio de la función *zfnAndChi_EpszAndSigz()*:

    Procedimiento:

    i.a Si $\chi = \infty$, la sección se encuentra en régimen plástico$\rightarrow z' = [z'_2, 0, z'_1]$; $\epsilon(z') = [\infty, \infty, \infty]$;$\sigma(z') = [\sigma_{p1}, 0, \sigma_{p2}]$.

    i.b Si $\chi \neq \infty$, la sección se encuentra en régimen elástico o elastoplástico:

    ii.b Se calcula el vector $z'$ que contiene las posiciones características del diagrama de tensiones, siendo $z'_i = \frac{-\epsilon_i}{\chi}$. $\epsilon_i$: deformaciones de los puntos característicos de la ley constitutiva del material. Se descartan las posiciones que no cumplan: $z'_2 \leq z'_i \leq z'_1$;

    iii.b Se calcula la distribución de deformaciones, $\epsilon(z') = -\chi\, z'$.

    iv.b Se calcula la distribución de tensiones ($\sigma(z')$) a partir de la distribución de deformaciones y la ley constitutiva del material por medio de la función *sigma()*.

  - Si la solicitación es axil puro ($\chi = 0$), la tensión normal y la deformación son uniformes a lo largo de la sección.

    i. Se calcula el vector $z'$. $z' = [z'_2, z'_1]$.

    ii. Se calcula la tensión uniforme, $\sigma = N/A(\Omega)$

    iii. Se calcula la deformación longitudinal unitaria a lo largo de la sección, $\epsilon = \sigma/E$.

    iv. Las leyes de tensiones y deformaciones quedan: $\sigma(z') = [\sigma, \sigma]$ y $\epsilon(z') = [\epsilon, \epsilon]$.

Debido a la relación entre tensión, deformación y esfuerzos se pueden implementar esquemas iterativos con los que a partir de cierta información se pueden determinar los parámetros $(z_{fn}, \chi)$ que definen una distribución de tensiones. Estos esquemas iterativos hacen cumplir las condiciones y se resuelven por medio de métodos de resolución de ecuaciones no lineales. Para ello se requiere una función con una variable independiente, que representa la condición que tenemos interés que se cumpla y dos valores de la variable independiente que o bien sea un intervalo que encierra al valor que hace cumplir a la condición o bien son valores próximos a dicho valor. Más una tolerancia, que es el grado de precisión con que la raíz aproximada cumple la condición. El procedimiento que siguen estos esquemas iterativos es:

1. Definir una condición $f(x) = 0$.

2. Especificar dos valores del dominio de x (a,b) que o bien estén próximos a la raíz de $f(x)$ o la acote.

3. Buscar un intervalo de x que contenga a la raíz a partir de los valores (a,b) mediante la función *metNum.approot()*.

4. Si se ha conseguido un intervalo que encierra a la raíz, se obtiene ésta por medio del método de Brent con la función *metNum.zero()*.

5. Si no se ha conseguido un intervalo, se obtiene la raíz por medio del método de la secante mediante *metNum.secant()*.

Los esquemas iterativos que se emplean en el programa son los que siguen:

- a) Calcular el valor de la posición de la fibra neutra a partir de los valores de curvatura ($\chi$) y axil ($N^*$), *chiAndN_zfn():*

  Para cierto valor de curvatura, solo existe un valor de $z_{fn}$ que hace cumplir la condición del axil.

  Si $z_{fne} = z_{fnp} \rightarrow z_{fn} = z_{fne}$

  Si $z_{fne} \neq z_{fnp} \rightarrow z_{fn}$ se calcula por medio del esquema iterativo:

  Condición: $f(z_{fn}) = N^* - N(z_{fn})|_\chi$

  Valores a,b: $\begin{cases} si\, z_{fne} > z_{fnp} & a = z_{fne};\, b = z_2 \\ si\, z_{fne} < z_{fnp} & a = z_{fne};\, b = z_1 \end{cases}$

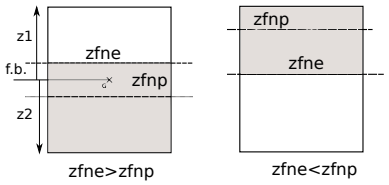



Figura 4.12: Búsqueda de $z_{fn}$ con *chiAndN_zfn()*

tolerancia: $10^{-6}$

Donde: $N(z_{fn})|_\chi$ esta implementado por medio de la función *zfnAndChi_N()*

$z_{fne}$ y $z_{fnp}$ se calculan por medio de *N_Me()* y *N_Mp()*, respectivamente

- b) Calcular el valor de la posición de la fibra neutra a partir de la profundidad de plastificación (***zp***) y el valor del axil ($N^*$) ,*zpAndN_zfn():*

  Para cierta profundidad de plastificación existe un valor de $z_{fn}$ que hace cumplir la condición del axil.

  Si $z_{fne} = z_{fnp} \rightarrow z_{fn} = z_{fne}$

  Si $z_{fne} \neq z_{fnp} \rightarrow z_{fn}$ se calcula por medio del esquema iterativo:

  Condición: $f(z_{fn}) = N^* - N(z_{fz})|_{\boldsymbol{zp}}$

  Valores a,b: $\begin{cases} si\, zp = z'_1 - z'_{e1} & a = z_2 + zp;\, b = max(z_{fne}, z_{fnp}) \\ si\, zp = z'_{e2} - z'_2 & a = z_1 - zp;\, b = min(z_{fne}, z_{fnp}) \end{cases}$

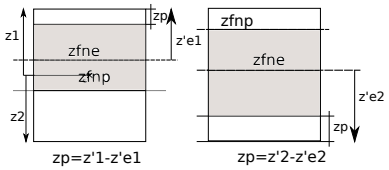



Figura 4.13: Búsqueda de $z_{fn}$ con *zpAndN_zfn()*

tolerancia: $10^{-6}$

Donde: $N(z_{fz})|_{\boldsymbol{zp}}$ esta implementado por medio de la función *zpAndzfn_N()*.

$z_{fne}$ y $z_{fnp}$ se calculan por medio de *N_Me()* y *N_Mp()*, respectivamente.

*zpAndN_zfn()* puede presentar limitaciones para material con endurecimiento con axil elevado en relación al momento. Debido a que una misma profundidad de plastificación puede estar asociado a más de una distribución de tensiones como se muestra en la Figura 4.14 (donde una misma profundidad medida desde el extremo superior de una sección, le corresponden diferentes valores de $z_{fn}$).

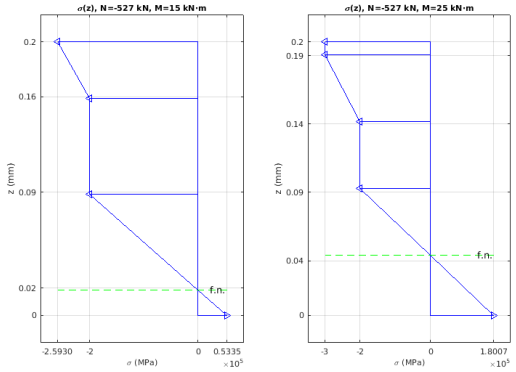



Figura 4.14: Limitación *zpAndN_zfn()*

- c) Calcular la posición de la fibra neutra y curvatura a partir de los valores del axil y momento (*NAndM_zfnAndChi()*)

Dados unos valores de axil ($N^*$) y momento ($M^*$), les corresponden una combinación de valores de $z_{fn}$ y $\chi$.

i. Garantizar que la solicitación $(N^*, M^*)$ cae dentro de la superficie de interacción, para ello:

Si $N < 0 \rightarrow N_p = \sigma_{pc}\, A(\Omega)$; si $N > 0 \rightarrow N_p = \sigma_{pt}\, A(\Omega)$;

Se debe cumplir $|N^*| < |N_p|$. Si $|N^*| > |N_p|$, el punto cae fuera del diagrama de interacción Axil - Momento y no tiene sentido continuar.

Se calcula $M_e|_{N^*}$ y $M_p|_{N^*}$ mediante *N_Me()* y *N_Mp()*, respectivamente.

Si $|M^*| > |M_p|_{N^*}|$ →la solicitación $(N^*, M^*)$ cae fuera del diagrama de interacción, por lo que es una solicitación inadmisible para la sección y no tiene sentido continuar.

ii. Determinar el régimen de trabajo de la solicitación (ver Figura 4.15a) y calcular los parámetros $z_{fn}$ y $\chi$:

Como $M(\chi)|_{N^*}$ es asintótico, para evitar que en la búsqueda de los valores de $z_{fn}$ y $\chi$ (en régimen elásto-plástico) tienda a infinito, se considera un momento máximo y curvatura máxima $(M_{max}|_{N^*}, \chi_{max}|_{N^*})$ que están asociadas a una profundidad de plastificación igual a 0,95 *zpmax*, estos valores se calculan mediante la función *zpAndN_momento()*.

*zpmax* se ha calculado previamente con *N_Mp()*.

ii.a Si $|M^*| \leq |M_e|_{N^*}| \rightarrow$régimen elástico. Los valores de $z_{fn}$ y $\chi$ se calcula por medio de las expresiones:

$\chi = \frac{M^*}{E\, I_y(\Omega)}$

$\sigma_1 = \sigma(z_1) = \frac{N^*}{A(\Omega)} - \frac{M^*}{I_y(\Omega)}\, z_1$; $\sigma_2 = \sigma(z_2) = \frac{N^*}{A(\Omega)} - \frac{M^*}{I_y(\Omega)}\, z_2$;

$z_{fn} = \frac{z_1\, \sigma_2 - z_2\, \sigma_1}{\sigma_2 - \sigma_1}$

ii.b Si $|M_e|_{N^*}| < |M^*| \leq |M_{max}|_{N^*}| \rightarrow$régimen elastoplástico. Los valores de $z_{fn}$ y $\chi$ se calculan por medio del siguiente esquema iterativo:

Condición: $f(\chi) = M^* - M(\chi)|_{N^*}$

Valores a y b: $a = \chi_e|_{N^*}$, $b = \chi_{max}|_{N^*}$

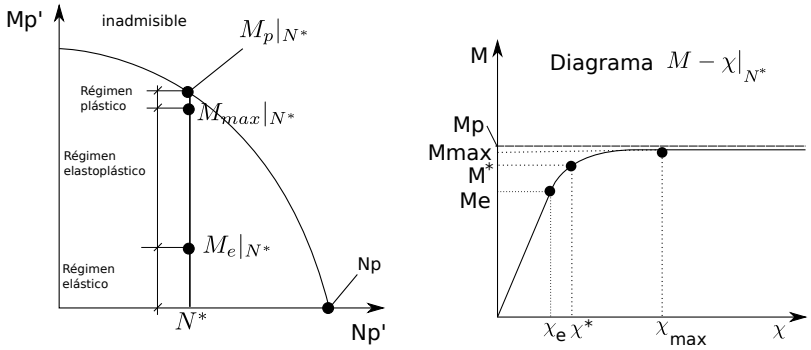
tolerancia: $10^{-6}$

Donde, $M(\chi)|_{N^*}$ está implementada por medio de la función *chiAndN_momento()*, ver Figura 4.15b. $\chi_e|_{N^*}$ ha sido previamente calculada por medio de la función *N_Me()*.

ii.c Si $|M_{max}|_{N^*}| < |M^*| \leq |M_p|_{N^*}|$ →régimen plástico. Por tanto, los valores $z_{fn}$ y $\chi$ son:

$z_{fn} = zfnp$, $\chi = \infty$

donde zfnp ha sido previamente calculada por medio de *N_Mp()*.




(a) Determinar régimen de la solicitación $(N^*, M^*)$

(b) $M(\chi)|_{N^*}$

Figura 4.15

#### 4.3.4.2. Cálculo de esfuerzos

A partir de la distribución de tensión se calcula la solicitación de la sección mediante las relaciones (estando los vértices de la sección referidos al sistema local $O'Y'Z'$):

$N = \int\int_\Omega \sigma(z')\, dy'\, dy'\, dz'$

$M_y = N\, z'_G - \int\int_\Omega \sigma(z')\, z'\, dy'\, dz'$

Para ello se requiere información que determine la distribución de tensiones normales, para así obtener los esfuerzos estáticamente equivalentes. Esta información depende del régimen de trabajo de la sección.

**Régimen elástico y elastoplástico**

Sabemos que en régimen elástico y elastoplástico la distribución de tensiones queda determinado por la fibra neutra y la curvatura. En el caso de régimen elastoplástico también puede quedar determinado por la profundidad de plastificación (***zp***) y la posición de fibra neutra ($z_{fn}$). Sabiendo la relación entre deformaciones, tensiones y esfuerzos, se puede calcular el axil y momento a partir de diferente información como sigue:

- A partir de un valor de axil, calcular el máximo momento que actuando simultáneamente con el axil ($N^*$) la sección se encuentra en régimen elástico (*N_Me()*):

  Si sabemos que la curvatura de $M_e|_{N^*}$ es positiva → $M_e|_{N^*} > 0$.

  Al ser régimen elástico, podemos aplicar el principio de superposición, por tanto la ley de tensiones viene definida por la expresión de Navier: $\sigma(z) = \frac{N}{A} - \frac{M_y}{I_y}\, z$.

  i. Cálculo de $M_e|_{N^*}$:

  Suponemos que plastifica primero el extremo superior, esto significa, ver Figura 4.16: $\sigma_1 = \sigma_{ec}$, $\sigma_2 \leq \sigma_{et}$, entonces:

$M_e|_{N^*} = \frac{-\sigma_{ec}+N^*/A(\Omega)}{z_1}\, I_y(\Omega)$,

si $\sigma_2 > \sigma_{et}$→plastifica primero la cara inferior→ $M_e|_{N^*} = \frac{-\sigma_{et}+N^*/A(\Omega)}{z_2}\, I_y(\Omega)$

ii. Cálculo de las tensiones en los extremos de la sección:

$\sigma_1 = \sigma(z_1) = \frac{N^*}{A(\Omega)} - \frac{M_e|_{N^*}}{I_y(\Omega)}\, z_1$; $\sigma_2 = \sigma(z_2) = \frac{N^*}{A(\Omega)} - \frac{M_e|_{N^*}}{I_y(\Omega)}\, z_2$

iii. Calcular la posición de la fibra neutra ($z_{fne}$):

$z_{fne} = \frac{\sigma_2\, z_1 - \sigma_1\, z_2}{\sigma_2 - \sigma_1}$

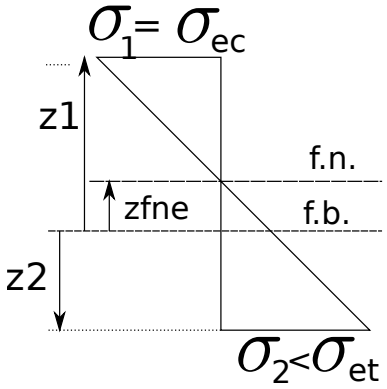



Figura 4.16: $\sigma(z')$ cuando plastifica primero el extremo superior

- El cálculo del axil y momento a partir del diagrama de tensiones, se obtiene mediante las funciones *zfnAndChi_N()* y *zfnAndChi_M(),* respectivamente. Se calculan por medio del siguiente procedimiento:

  Consideramos de forma general que la distribución de tensiones queda dividida en n tramos. El tramo k, tiene una relación de tensión - deformación de la forma:

  $\sigma_k(z') = \sigma_{i-1} + \frac{\sigma_i - \sigma_{i-1}}{\epsilon_i - \epsilon_{i-1}}\, (\epsilon(z') - \epsilon_{i-1})$, siendo la deformación de las fibras dentro de la región $\Omega_k$, $\epsilon_{i-1} \leq \epsilon(z') \leq \epsilon_i$, $(y', z') \in \Omega_k$.

  La distribución de la tensión en la porción $\Omega_k$ queda:

  $\sigma_k(z') = \alpha_k + \beta_k\, z'$

  Siendo, $\alpha_k$ y $\beta_k$ constantes propias de la región $\Omega_k$:

  $\alpha_k = \sigma_{i-1} - \frac{\sigma_i - \sigma_{i-1}}{\epsilon_i - \epsilon_{i-1}}\, \epsilon_{i-1}$; $\beta_k = -\frac{\sigma_i - \sigma_{i-1}}{\epsilon_i - \epsilon_{i-1}}\, \chi$

  Por tanto el cálculo del axil y momento quedan:

  $N = \sum_{k=1}^{n} \int\int_{\Omega_k} \sigma_k(z') dy'\, dz' = \sum_{k=1}^{n} \int\int_{\Omega_k} (\alpha_k + \beta_k\, z')\, dy'\, dz' = \sum_{k=1}^{n} [\alpha_k\, A(\Omega_k) + \beta_k\, W_{y'}(\Omega_k)]$

  $M = N\, z'_G - \sum_{k=1}^{n} \int\int_{\Omega_k} \sigma_k(z')\, z'\, dy'\, dz' = \sum_{k=1}^{n} \int\int_{\Omega_k} (\alpha_k\, z' + \beta_k\, z'^2)\, dy'\, dz' = \sum_{k=1}^{n} -[\alpha_k\, W_{y'}(\Omega_k) + \beta_k\, I_{y'}(\Omega_k)]$

  Las regiones $\Omega_k$ se calculan mediante la función *problGeom.clipArea().*

- Calcular el axil a partir de la profundidad de plastificación (***zp***) y el valor de la posición de la fibra neutra ($z_{fn}$), *(zpAndzfn_N())*:

  i. Se calcula la curvatura ($\chi$) a partir de la profundidad de plastificación (***zp***), la posición de la fibra neutra ($z_{fn}$) y ley constitutiva, mediante la expresión: $\chi = -\frac{\epsilon_e}{z'_e}$.

  ii. Se calcula el valor del axil mediante la función *zfnAndChi_N().*

- Calcular el momento a partir de la profundidad de plastificación (***zp***) y el valor del axil ($N^*$) que actúa en la sección (*zpAndN_momento()*)

  i. Se calcula la posición de la fibra neutra ($z_{fn}$) y curvatura ($\chi$) mediante la función *zpAndN_zfn()*.

  ii. Con $z_{fn}$, $\chi$ y $N^*$, se calcula el momento por medio de *zfnAndChi_M()*.

- Calcular el momento a partir de la curvatura y el valor del axil (*chiAndN_momento()*)

  i. Se calcula la posición de la fibra neutra mediante la función *chiAndN_zfn()*.

  ii. Con $z_{fn}$, $\chi$ y $N^*$, se calcula el momento por medio de la función *zfnAndChi_M()*.

**Régimen plástico**

Sabemos que en régimen plástico los valores de curvatura y posición de fibra neutra cumplen: $\chi = \infty$, $z_2 \leq z_{fn} \leq z_1$. Por lo que solo se requiere la posición de la fibra neutra ($z_{fnp}$) para determinar una distribución de tensiones cuando la sección tiene todas sus fibras plastificadas por flexión pura o compuesta:

$$\sigma(z) = \begin{cases} \sigma_{p2} & z_2 \leq z \leq z_{fnp} \\ \sigma_{p1} & z_{fnp} < z \leq z_1 \end{cases}$$

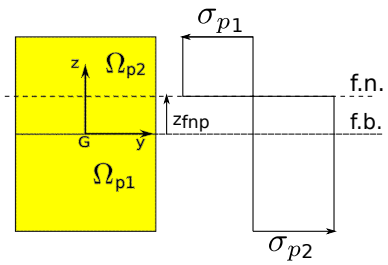



Figura 4.17: $\sigma(z)$ sección con todas las fibras plastificadas.

- Calcular la solicitación de agotamiento a flexión compuesta de una sección a partir del valor de la posición de la fibra neutra (*zfnp_NpMp()*)

  Teniendo el valor de la posición de la fibra neutra ($z_{fnp}$), se puede calcular el axil y momento que son estáticamente equivalentes a la distribución de tensiones, mediante:

  $N_p = \sigma_{p1}\, A(\Omega_{p1}) + \sigma_{p2}\, A(\Omega_{p2})$

  $M_p = N\, z'_G - \sigma_{p1}\, W_{y'}(\Omega_{p1}) - \sigma_{p2}\, W_{y'}(\Omega_{p2})$

  Donde, $\Omega_{p1}$ y $\Omega_{p2}$ son la región de la sección que queda por encima y por debajo de la fibra neutra ($z_{fnp}$), respectivamente. Estas regiones se calculan por medio de *problGeom.clipArea()*.

- A partir de un valor de axil, calcular el momento que actuando simultáneamente con el axil agota la sección (*N_Mp()*)

  i. Calcular la posición de fibra neutra ($z_{fnp}$). Este valor se calcula por medio de un esquema iterativo que hace cumplir la condición del axil:

  Condición: $f(z_{fnp}) = N^* - N(z_{fnp})$

  Valores a, b: $a = z_2$ , $b = z_1$

  Tolerancia: $10^{-6}$

  ii. Se calcula $M_p|_{N^*}$ por medio de la función *zfnp_NpMp()*

iii. Se calcula la profundidad máxima de plasticación (zpmax), para el valor de axil $N^*$. Si la sección plastifica primero el extremo superior$\rightarrow zpmax = z_1 - z_{fnp}$. Si la sección plastifica primero el extremo inferior $\rightarrow zpmax = zfnp - z_2$. Conocer qué sección plastifica primero se determina previamente mediante la función *N_Me()*.

#### 4.3.4.3. Cálculo del factor de forma

El factor de forma ($\lambda$) es la relación entre el momento plástico y momento elástico de una sección sometida a flexión pura (axil es nulo). Este valor se calcula por medio de la función *calcular_factorForma()* que sigue el procedimiento:

i. N=0.

ii. Se calcula el momento plástico ($M_p$) por medio de *N_Mp()*.

iii. Se calcula el momento elástico ($M_e$) por medio de *N_Me()*.

iv. Se calcula el factor de forma, $\lambda = \frac{M_p}{M_e}$

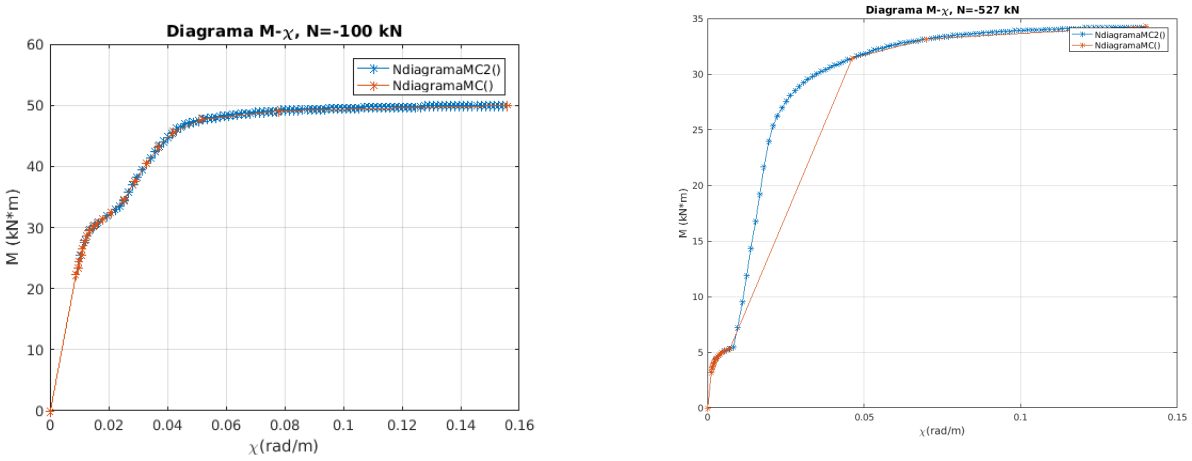
#### 4.3.4.4. Cálculo del diagrama Momento - Curvatura

El programa por defecto calcula el diagrama momento curvatura correspondiente al axil nulo, pero se puede calcular el diagrama $M - \chi$ de la sección para un valor de axil concreto, diagrama $M - \chi|_N$.

El diagrama Momento - Curvatura (d_MC) se puede calcular mediante dos funciones: *N_diagramaMC()* o *N_diagramaMC2()*.

*N_diagramaMC* calcula los valores de curvatura y momento a partir de la profundidad de plastificación medida desde el extremo que plastifica primero. *N_diagramaMC2()* calcula los valores de momento directamente a partir de los valores de curvatura.

*N_diagramaMC()* presenta la ventaja de que con un menor número de puntos (20 puntos) el diagrama queda bien definido. Por medio de *N_diagramaMC2()*, se requieren 100 puntos ($\chi_i, M_i$) para que el diagrama sea representativo. Sin embargo, la desventaja de *N_diagramaMC()* es que para el caso de material con endurecimiento con valores grandes de axil, no ofrece buenas soluciones debido a la limitación de la función *zpAndN_zfn()*.




(a) Axil no elevado

(b) Axil elevado

Figura 4.18: Limitación de la función *N_diagramaMC()* con secciones con material con endurecimiento.

El procedimiento que sigue la función *N_diagramaMC()* es:

i. Se determina el extremo (superior/inferior) de la sección que plastifica primero por medio de *N_Me().*

ii. Se calcula la profundidad de plastificación máxima (zpmax) y su carácter (compresión/tracción) a partir de la función *N_Mp().*

iii. Se crea el vector que contienen todas las profundidades de plastificación ($zp_i$) a las que se va a calcular su momento ($M_i$) y curvatura ($\chi_i$) correspondientes. Siendo $zp_i \in [0, 0'95\, zpmax]$. i=1,...,20.

iv. Para cada $zp_i$ se calcula $M_i$ y $\chi_i$ mediante la función *zpAndN_momento().*

El procedimiento que sigue la función *N_diagramaMC2()* es:

i. Se calcula la profundidad de plastificación máxima (zpmax) mediante *N_Mp().*

ii. Se calcula la curvatura máxima que se considera ($\chi_{max}$). Esta curvatura corresponde al la profundidad de plastificación 0,95 *zpmax*. $\chi_{max}$ se calcula por medio de la función *zpAndN_momento().*

iii. Se calcula el vector que contiene todas las curvaturas, $\chi_i$. $\chi_i \in [0, \chi_{max}]$, i=1,...,100.

iv. Para cada $\chi_i$ se calcula el momento correspondiente $M_i$,por medio de la función *chiAndN_momento().*

#### 4.3.4.5. Cálculo del diagrama interacción $N'_p - M'_p$

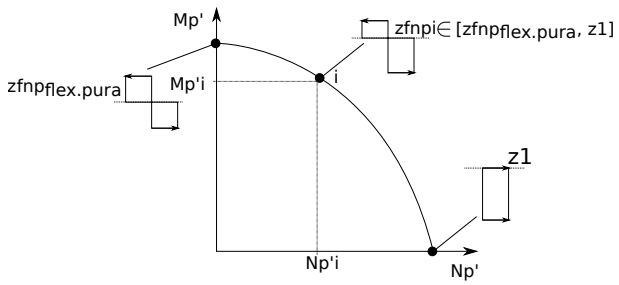



Figura 4.19: Diagrama interacción axil - momento ($N > 0$, $M > 0$)

El diagrama de interacción Axil - Momento se calcula por medio de la función *diagrama_interaccionNM()* y su procedimiento es:

i. Se calcula la posición de la fibra neutra asociada al diagrama de tensión normal de agotamiento cuando la sección está solicitada por flexión pura ($zfnp_{flex.pura}$).

ii. Se calcula el vector que contiene todas las posiciones de fibra neutra ($zfnp_i$)

ii.a Si el axil es de tracción y el momento comprime el extremo superior de la sección, $zfnp_i \in [zfnp_{flex.pura}, z1]$, ver Figura 4.19.

ii.b Si el axil es de compresión y el momento comprime el extremo superior de la sección,$zfnp_i \in [z_2, zfnp_{flex.pura}]$.

iii. Se calculan los puntos del diagrama ($N'p_i, M'p_i$). Para cada posición de fibra neutra, $zfnp_i$, se calcula $N'p_{pi}$ y $M'p_i$ mediante la función zfnp_NpMp().

### 4.3.5. Propiedades de las secciones

Las secciones se caracterizan por su geometría y por el material del que están compuestas, de esto se encarga el módulo *propSeccion.* Este módulo está compuesto, por tanto, de dos funciones principales:

*propSeccion.propGeom():* función que calcula las propiedades geométricas de las secciones o porciones de éstas.

*propSeccion.propMaterial():* calcula las propiedades del material del que están compuestas las secciones.

#### 4.3.5.1. Propiedades geométricas

Por medio del Teorema de Green se puede relacionar la integral de superficie con la integral de línea. [8]

**Teorema 1** ***(Teorema de Green)*** *Sea C una curva continua a trozos y Ω una región que consiste en C y su interior. Si P(y,z) y Q(y,z) son funciones continuas de Clase 1 a lo largo de la región Ω, entonces:*

$\int\int_\Omega (\frac{\partial P}{\partial y} - \frac{\partial Q}{\partial z})\, dy\, dz = \oint_C (P\, dy + Q\, dz)$

El signo de la integral de línea de la derecha está asociado a la orientación de la curva, de manera que si se camina sobre ella con la orientación de la misma, si la región Ω queda a la izquierda, entonces la integral es positiva.

Un polígono es una curva cerrada compuesta por varios segmentos $C_i$. Por lo que el Teorema de Green queda:

$\oint (P(y,z)\, dy + Q(y,z)\, dz) = \sum_i \int_{C_i} (P(y,z)\, dy + Q(y,z)\, dz)$

Cada uno de estos segmentos está definido por dos vértices consecutivos con coordenadas $(y_i, z_i)$ y $(y_j, z_j)$, que forman una arista del polígono y tiene como ecuación:

$z = z_i + \lambda\,(y - y_i)$

o bien, $y = y_i + \frac{1}{\lambda}\,(z - z_i)$

Siendo:

$\lambda = \frac{z_j - z_i}{y_j - y_i}; \quad i = 1, 2, ..., n;\ j = \begin{cases} i+1 & i < n \\ 1 & i = n \end{cases}$

Por tanto, aplicando el Teorema de Green a las expresiones del área y momentos de área, se pueden obtener expresiones con las que se pueden calcular dichas propiedades geométricas a partir de las coordenadas de los vértices de los polígonos. Estas expresiones son:

$A = \int\int_\Omega dy\, dz = -\oint_C z\, dy = -\sum_{i=1}^{n} \int_{y_i}^{y_j} (z_i + \lambda\,(y - y_i))\, dy = -\frac{1}{2} \sum_{i=1}^{n} (z_i + z_j)\,(y_j - y_i);\ (P = -z,\ Q = 0)$

$W_y = \int\int_\Omega z\, dy\, dz = -\frac{1}{2} \oint_C z^2\, dy = -\frac{1}{2} \sum_{i=1}^{n} \int_{y_i}^{y_j} [z + \lambda\,(y - y_i)]^2\, dy = -\frac{1}{6} \sum_{i=1}^{n} (y_j - y_i)\,(z_i^2 + z_i\, z_j + z_j^2);$
$(P = -z^2/2,\ Q = 0)$

$W_z = \int\int_\Omega y\, dy\, dz = \frac{1}{2} \oint_C y^2\, dz = \frac{1}{2} \sum_{i=1}^{n} \int_{z_i}^{z_j} [y_i + \frac{1}{\lambda}\,(z - z_i)]^2\, dz = \frac{1}{6} \sum_{i=1}^{n} (z_j - z_i)\,(y_i^2 + y_i\, y_j + y_j^2);\ (P = 0,\ Q = y^2/2)$

$I_y = \int\int_\Omega z^2\, dy\, dz = -\frac{1}{3} \oint_C z^3\, dy = -\frac{1}{3} \sum_{i=1}^{n} \int_{y_i}^{y_j} [z_i + \lambda\,(y - y_i)]^3\, dy = -\frac{1}{12} \sum_{i=1}^{n} (y_j - y_i)\,(z_i^3 + z_i^2\, z_j + z_i\, z_j^2 + z_j^3);$
$(P = -y^3/3,\ Q = 0)$

$I_z = \int\int_\Omega y^2\, dy\, dz = \frac{1}{3} \oint_C y^3\, dz = \frac{1}{3} \sum_{i=1}^{n} \int_{z_i}^{z_j} [y_i + \frac{1}{\lambda}\,(z - z_i)]^3\, dz = \frac{1}{12} \sum_{i=1}^{n} (z_j - z_i)\,(y_i^3 + y_i^2\, y_j + y_i\, y_j^2 + y_j^3);$
$(P = 0,\ Q = z^3/3)$

A partir de los valores de área y momentos estáticos se obtienen los valores de las coordenadas del centro de gravedad:

$y_G = \frac{W_z}{A};\ z_G = \frac{W_y}{A}.$

**Cálculo de propiedades geométricas de secciones compuestas**

Si una sección está compuesta por varias regiones que no se superponen entre ellas, $\Omega = \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_n$, sus propiedades geométricas se calculan de la siguiente manera:

$I_y(\Omega) = \sum_{i=1}^{n} I_y(\Omega_i);\ I_z(\Omega) = \sum_{i=1}^{n} I_z(\Omega_i)$

$W_y(\Omega) = \sum_{i=1}^{n} W_y(\Omega_i);\ W_z(\Omega) = \sum_{i=1}^{n} W_z(\Omega_i)$

$y_G(\Omega) = \frac{\sum_{i=1}^{n} y_G(\Omega_i)\, A(\Omega_i)}{\sum_{i=1}^{n} A(\Omega_i)};\ z_G(\Omega) = \frac{\sum_{i=1}^{n} z_G(\Omega_i)\, A(\Omega_i)}{\sum_{i=1}^{n} A(\Omega_i)}$

#### 4.3.5.2. Propiedades del material

Para los cálculos puede ser necesario determinar las siguientes propiedades del material a partir de su ley constitutiva ($\sigma = \sigma(\epsilon)$).

$\epsilon_{ec}$, $\epsilon_{et}$ :deformación unitaria en el límite elástico a compresión y tracción, respectivamente.

$\sigma_{ec}$, $\sigma_{et}$ : tensión en el límite elástico a compresión y tracción, respectivamente.

$E$ :módulo de Young, que es igual a $E = \frac{\sigma_{et}}{\epsilon_{et}}$.

$\sigma_{pc}$, $\sigma_{pt}$ :tensión última de fluencia (antes de romper) a compresión y tracción.

Estas propiedades se obtienen, para cualquier tipo de modelo de material, sabiendo que $\sigma_{pt}$ es el valor máximo de tensión y $\sigma_{pc}$ es el valor mínimo de tensión. Puesto que los puntos característicos de la ley constitutiva están ordenados de menor a mayor, $(\epsilon_{ec}, \sigma_{ec})$ es el punto anterior a (0,0) y $(\epsilon_{et}, \sigma_{et})$ es punto posterior a (0,0).

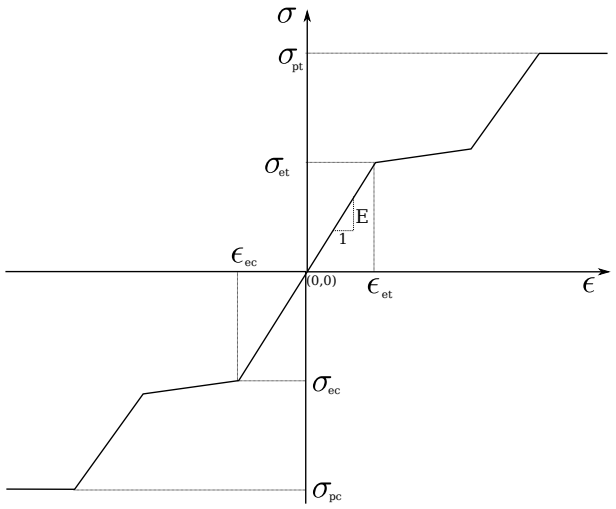



Figura 4.20: Cálculo de propiedades del material

### 4.3.6. Problemas geométricos

Los problemas geométricos que requiere el programa son: calcular los puntos de corte entre un polígono y una recta, determinar la porción de un polígono que queda a un lado de una recta y cambiar el sistema de referencia de un conjunto de puntos. De estos problemas se ocupa el módulo *problGeom*.

Como nos podemos encontrar con la situación que muestra la Figura 4.21, los polígonos se pueden ver de dos formas:

polígono simple: formado por un sólo polígono (array), o

polígono compuesto: formado por un conjunto de polígonos. Estos se definen por medio de una estructura cell cuyos componentes son polígonos simples (array).

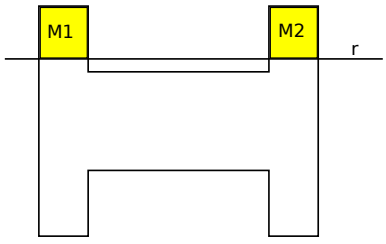



Figura 4.21: Ejemplo de un polígono compuesto

Por tanto las variables que maneja este módulo son:

P: en el caso de ser un polígono simple: polígono formado por una sucesión de puntos ordenados de manera que dos puntos consecutivos forman una arista y de forma que la región que encierra siempre queda a la izquierda de la arista. Si se trata de un polígono compuesto, $P = \{P_1, P_2, ..., P_n\}$, donde $P_i$ es un polígono simple.

r: recta definida por las constantes a,b y c. Siendo así la ecuación de la recta: $c\,y = a\,x + b$. $r = [a, b, c]$.

#### 4.3.6.1. Cambio de sistema de referencias

Se busca cambiar el sistema de referencia de un conjunto de puntos, $P$, referidos al sistema de referencia $OXY$ a otro $O'X'Y'$. Este cambio de sistema de referencia puede suponer una rotación y una traslación de ejes.

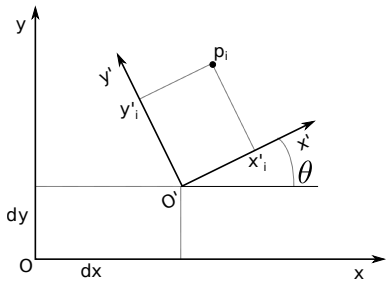



Figura 4.22: Cambio de sistema de referencia (traslación y rotación)

Por tanto, para todo $p_i \in P$

Siendo $p_i$ un punto del conjunto $P$ con coordenadas $(x_i, y_i)$:

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} -dx \\ -dy \end{bmatrix}$$

*cambioSistCoord()* es la función que cambia el sistema de coordenadas de un polígono simple.

*calculo_cambioSistCoord()* es la función que cambia el sistema de coordenadas de un conjunto de polígonos compuestos.

#### 4.3.6.2. Calcular los puntos de corte entre un polígono y una recta

Sea una región $\Omega \subset \mathbb{R}^2$, con un contorno P compuesta por un conjunto de segmentos rectos $l_i$.

Sea una recta r que se desarrolla en el espacio $\mathbb{R}^2$ definida por la ecuación:

$r : c\,y = a\,x + b$

Siendo a,b,c: constantes.

Los puntos de corte se obtienen de la intersección de P y r, y cumplen:

$\boldsymbol{pc} = P \cap r = \{(x, y) \in P : c\,y - a\,x = b\}$

Para calcular los puntos de corte ***pc*** hay que evaluar la intersección de la recta r con cada uno de los segmentos rectos $l_i$.

$l_i$ está delimitado por dos puntos con coordenadas $(x_i, y_i)$ y $(x_j, y_j)$.

Siendo,

$$i = 1, ..., nvert; \qquad j = \begin{cases} i+1 & i < nvert \\ 1 & i = nvert \end{cases}$$

$nvert$ : número de vértices del polígono $P$.

Existe un punto de intersección entre $l_i$ y r si:

[I] $c\,y_i - a\,x_i > b$ y $c\,y_j - a\,x_j < b$ o

[II] $c\,y_i - a\,x_i < b$ y $c\,y_j - a\,x_j > b$ o

[III] $c\,y_i - a\,x_i = b$ o

[IV] $c\,y_j - a\,x_j = b$

Para evitar calcular dos veces el mismo punto de corte, evaluaremos para cada $l_i$ las condiciones [I], [II] y [III]. De manera que la condición [IV] se evaluaría con el siguiente segmento $l_{i+1}$.

- Si existe un punto de corte, se calcula de manera que:
  - Si se cumple la condición [III] $\rightarrow pc_k = (x_i, y_i)$
  - Si se cumple la condición [I] o [II]$\rightarrow$ se calcula por el método de Cramer las coordenadas del punto de corte entre $l_i$ y r:

    $pc_k = l_i \cap r = \{(x, y) \in l_i : c\,y - a\,x = b\}$

    Sea la ecuación de la recta de $l_i : c_i\,y - a_i\,x = b_i$

    Por tanto el punto de corte se calcula resolviendo:

    $$\begin{bmatrix} c_i & -a_i \\ c & -a \end{bmatrix} \begin{bmatrix} yc_k \\ xc_k \end{bmatrix} = \begin{bmatrix} b_i \\ b \end{bmatrix}$$

    $xc_k = \frac{c_i\,b - c\,b_i}{-c_i\,a + c\,a_i}$; $yc_k = \frac{-b_i\,a + b\,a_i}{-c_i\,a + c\,a_i}$

    Siendo, $k = 1, ..., n_{pc}$

    $n_{pc}$ :número de puntos de corte entre P y r.

*pc_poly_r()*: es la función que calcula los puntos de corte entre una recta y un polígono simple.

*calculo_pc_poly_r()*: es la función que calcula los puntos de corte entre una recta y un polígono compuesto.

#### 4.3.6.3. Calcular la porción de una sección que cae a un lado de una recta

Sea una región $\Omega \subset \mathbb{R}^2$, con un contorno P compuesta por un conjunto de segmentos rectos $l_i$.

Sea una recta r que se desarrolla en el espacio $\mathbb{R}^2$ definida por la ecuación:

$r : c\,y = a\,x + b$

Siendo a,b,c: constantes.

Se busca determinar la porción o porciones de P que quedan a un lado de r. Esto es un problema conocido de Geometría Computacional. La función *clipArea()* sigue la estrategia de solución de Shutherland - Hodgman.

Este problema se resuelve por medio de los siguientes pasos:

1. Determinar qué vértices del polígono cumplen la condición.
2. Comprobar si se dan situaciones especiales.
3. Calcular todos los vértices que forman parte del contorno de la porción del polígono que queda a un lado de la recta.
4. Clasificar las aristas y generar el polígono o polígonos resultantes.

**Paso 1: Determinar qué vértices cumplen la condición**

En esta fase se crea la matriz booleana B, cuyos componentes son 1 si el vértice de P cumple la condición y 0 en caso contrario.

Los puntos $p_i \in P$ que cumplen la condición son los que cumplen alguna de las siguientes condiciones:

[I] $c\,y_i - a\,x_i > b\ y\ c\,y_j - a\,x_j < b$ o

[II] $c\,y_i - a\,x_i < b\ y\ c\,y_j - a\,x_j > b$ o

[III] $c\,y_i - a\,x_i = b$ o

[IV] $c\,y_j - a\,x_j = b$

Si algún punto cumple la condición [I] o [II] se añade un 1, en caso contrario se añade un 0 a la matriz B.

Si el vértice $p_i$ cumple la condición [III], se incrementa una unidad a la variable vi_In_r, que cuenta el número de vértices del polígono P que caen sobre la recta r.

**Paso 2: Comprobar situaciones especiales**

En esta fase se contemplan las situaciones:

a) El polígono P y r no cortan, este caso se da bien porque todos los elementos de B son nulos o bien porque siendo todos los elementos de B la unidad, ningún punto del polígono coincide con r (vi_In_r=0). Por tanto, la salida de la función en este caso sería un conjunto vacío.

b) La recta r es tangente al polígono P. En este caso todos los elementos de B son la unidad y algún vértice de P coincide con r. Así, la salida de la función en este caso sería el polígono P.

De darse alguna de estas dos situaciones la función no continuaría con los siguientes pasos.

**Paso 3: Calcular todos los vértices**

Este paso consiste en añadir todos los vértices que forman el contorno de la región buscada y los cuales son recogidos en la matriz R. Este paso se basa en el algoritmo de Shutherland - Hodgman que distingue cuatro situaciones al evaluar dos vértices de P consecutivos (pi y pj), las podemos ver con la Figura 4.23:

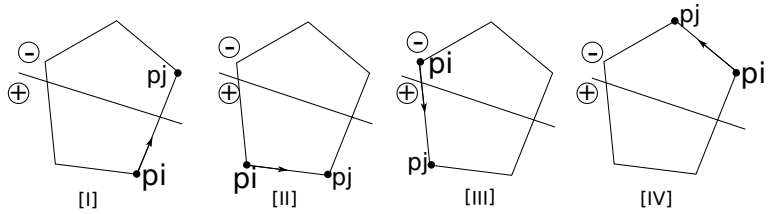



Figura 4.23: Evaluación de los vértices según Sutherland-Hodgman

Caso [I]: pi cumple la condición y pj no cumple condición, se añade el vértice pi y se añade el punto de intersección entre $l_i$ y r.

Caso [II]: pi y pj cumplen la condición, ambos vértices se añaden.

Caso [III]: pi no cumple condición y pj sí cumple condición, en este caso se añade el punto de intersección entre $l_i$ y r, y también el vértice pj. En este caso además añadimos el índice del punto nuevo generado en este paso y lo almacenamos en la variable outInArray, que son los puntos candidatos a formar el final de un polígono y el comienzo del siguiente, evaluación que se considera en el siguiente Paso 4.

Caso [IV]: tanto pi como pj no cumplen condición, en este caso no se añade ningún punto a la matriz R.

**Paso 4: Clasificar aristas y generar polígonos**

Del paso anterior pueden resultar que una región del polígono P, que cumple la condición, esté dividida en varias regiones como se muestra en la Figura 4.24.

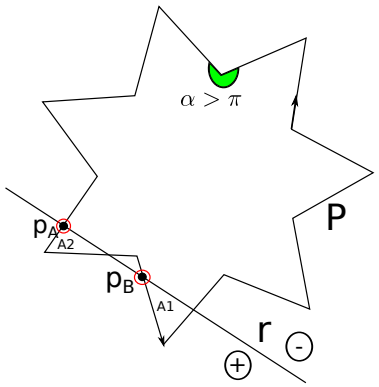



Figura 4.24: Polígono cóncavo con más de una porción resultante ($A_1$ y $A_2$)

Esto sucede con los polígonos cóncavos (polígonos que presentan al menos un ángulo interno entre dos aristas mayor que $\pi$ radianes). Por tanto se requiere este cuarto paso, en el que se clasifican las aristas. Esto es, determinar si pertenecen o no al nuevo contorno. Si no pertenece, significa que es el inicio de un nuevo polígono.

Las aristas que se consideran son las formadas por los puntos generados en el paso anterior cuando se evaluaba el caso [IV] y el vértice anterior de $R$, estos son, los puntos candidatos a ser el inicio de una nueva porción y además, se encuentran sobre la recta r. Los índices de estos puntos vienen recogidos en la variable *index_ouInArray*. Se puede ver en la Figura 4.24 los puntos marcados $p_A$ y $p_B$ son candidatos a iniciar una nueva porción.

Una arista forma parte del contorno si se cumplen las condiciones: un punto intermedio de la arista se encuentra en el interior del polígono P, no existe ningún punto de corte en el interior de la arista. Estas condiciones se evalúan de la siguiente forma:

Un punto candidato a ser el inicio de una nueva porción ($R_n$), tiene asociados dos puntos: el punto anterior que cumple la condición ($R_{n-1}$) y el punto previo dentro de P, que no cumple la condición ($p_{out}(R_n)$). Para que $R_n$ sea continuación de $R_{n-1}$, esto significa que $R_n$ no es el comienzo de una nueva porción, se debe cumplir:

i. Área del triángulo formado por los puntos $R_{n-1}$, $p_{out}(R_n)$ y $R_n$ ha de ser positivo.

ii. No debe haber ningún punto de corte entre $R_{n-1}$ y $R_n$.

Sea por ejemplo el caso de la Figura 4.25, donde $R_i$ y $R_j$ son candidatos a ser puntos que inician una nueva porción.

Evaluando el punto $R_i$: i. Área[triángulo($R_{i-1}$, $p_{out}(R_i)$, $R_i$)]<0→$R_i$ no es continuación de $R_{i-1}$. Por tanto, comienza una nueva porción con $R_i$.

En el caso de $R_j$: i. Área[triángulo($R_{j-1}$, $p_{out}(R_j)$, $R_j$)]>0 . ii. No existe ningún punto de corte entre $R_j$ y $R_{j-1}$. Como se cumplen las condiciones i y ii, $R_j$ es continuación de $R_{j-1}$.

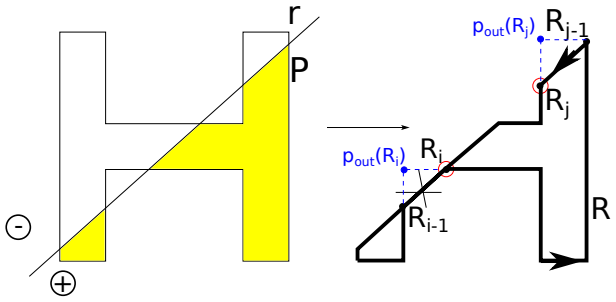



Figura 4.25: Evaluación de las aristas

### 4.3.7. Resolución de ecuaciones no lineales

Este problema consiste en hallar el valor de x que hace cumplir la condición $f(x) = 0$.

Los métodos de resolución de ecuaciones no lineales de una variable se dividen en dos grupos principales: aquellos que parten de un intervalo que contiene a la raíz y los que no parten de dicho intervalo.

Siempre se recomienda que el método iterativo no salga de un intervalo, ya que podría suponer la no convergencia de la solución[16]. Es por esto que siempre que se obtenga un intervalo que contenga a la raíz por medio de la función *approot()* se optará por el método de Brent (*zero()*). En caso de no haya sido posible encontrar un intervalo, se recurrirá al método de la secante (*secant()*).

#### 4.3.7.1. Cálculo del intervalo que contiene una raíz

Se busca un intervalo $(a, b)$ que contenga una raíz de $f(x)$. Dicho intervalo deberá cumplir la condición $f(a)\, f(b) < 0$. Si la función es continua por el Teorema del Valor Medio se sabe que existe un valor de x entre a y b que cumple $f(x) = 0$.

Para obtener dicho intervalo $(a, b)$, se evalúa un tramo $(A, B)$ del dominio de la función. Este tramo se subdivide en n subtramos. Se evalúa cada subtramo $(x_i, x_{i+1})$, si $f(x_i)\, f(x_{i+1}) < 0 \rightarrow a = x_i$, $b = x_{i+1}$.

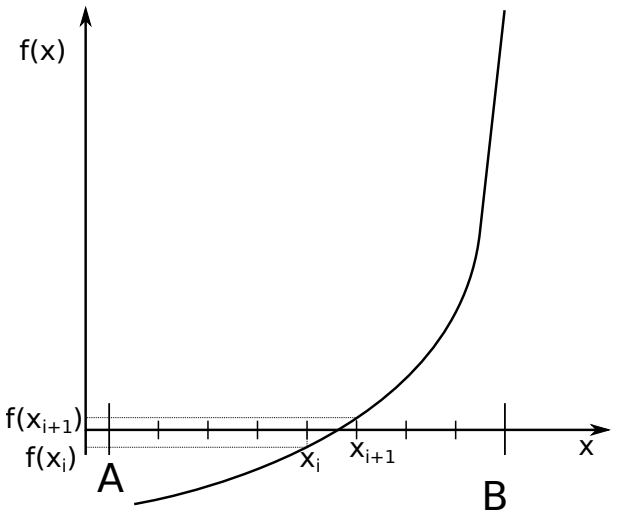



Figura 4.26: Búsqueda de un intervalo mediante *approot()*

#### 4.3.7.2. Método de la secante

Se busca el valor $x_r$ que haga $f(x_r) = 0$.

El método de la secante es un método iterativo en el que cada iteración considera la función $f(x)$ localmente lineal en el intervalo entre dos puntos $x_1$ y $x_2$, de manera que considera: $f(x) = A\, x + B$, siendo $A = \frac{f(x_2)-f(x_1)}{x_2-x_1}$ y $B = f(x_1) + \frac{f(x_2)-f(x_1)}{x2-x1}\, x_1$, por lo que haciendo $0 = a\, x + b$ y despejando x obtendríamos una estimación de la raíz de $f(x)$ a partir de los valores $x_1$ y $x_2$.

Por tanto, el esquema iterativo queda:

$x_0 = a$

$x_1 = b$

$x_i = x_{i-1} - f(x_{i-1})\, \frac{x_{i-1}-x_{i-2}}{f(x_{i-1})-f(x_{i-2})}$

si $|f(x_i)| < tolerancia \rightarrow x_r = x_i$

sino, $i = i + 1$

Siendo,

$x_r$ :raíz de $f(x)$, que cumple $|f(x_r)| < tolerancia$

$x_{i-1}, x_{i-2}$ :las dos estimaciones de la raíz

$a, b$ : dos valores del dominio de $f(x)$. A ser posible, próximos al valor de la raíz, con los que se pone en marcha el método

*tolerancia* :valor que consideramos

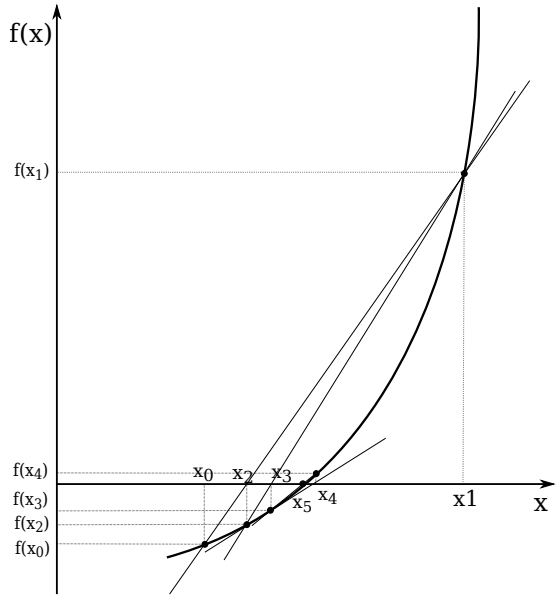



Figura 4.27: Ejemplo de uso del método de la secante

#### 4.3.7.3. Método de Brent

Se busca el valor $x$ que haga $f(x) = 0$. Se parte de un intervalo $(a, b)$ que contiene la raíz.

El método de la bisección es confiable, pero puede suponer gran número de iteraciones para alcanzar una raíz con una tolerancia prefijada. Por otro lado, el método de la secante es rápido pero no se garantiza que la búsqueda de la raíz esté dentro de un intervalo, lo que puede llevar a que la solución no converja.

El método de Brent combina la fiabilidad del método de la bisección con la rapidez de un método como el de la secante, es por ello la razón de su elección siempre que dispongamos de un intervalo que acote a la raíz.

En cada iteración, la estimación se puede llevar a cabo por medio de una interpolación inversa cuadrática, el método de la secante (interpolación lineal) o bien por el método de la bisección.

El método de Brent se basa en hacer una estimación en cada paso por medio de una interpolación inversa cuadrática. De esta forma se hace la interpolación de $f(x)$ a partir de tres puntos a, b y c, por tanto su expresión es:

$x = \frac{(y-f(a))\,(y-f(b))\,c}{(f(c)-f(a))\,(f(c)-f(b))} + \frac{(y-f(b))\,(y-f(c))\,a}{(f(a)-f(b))\,(f(a)-f(c))} + \frac{(y-f(c))\,(y-f(a))\,b}{(f(b)-f(c))\,(f(b)-f(a))}$

Haciendo y=0 y despejando x, obtenemos la expresión de la estimación por medio de una interpolación inversa cuadrática[16]:

$x = b + P/Q$, siendo:

$P = S\,[T\,(R-T)\,(c-b) - (1-R)\,(b-a)]$

$Q = (T-1)\,(R-1)\,(S-1)$

$R = f(b)/f(c)$; $S = f(b)/f(a)$; $T = f(a)/f(c)$.

$b$ :es la mejor estimación, por tanto $|f(b)| < |f(a)|, |f(b)|$

$P/Q$ :es una pequeña corrección.

$c$ : es el contrapunto, valor que cumple $f(c)\,f(b) < 0$

Puesto que esta estimación ofrece malos resultados cuando $Q \simeq 0$, el método de Brent se anticipa a este problema. Así, si $P/Q$ no cae dentro de cierto rango, el método de Brent fuerza que la estimación se haga por medio de una bisección. De esta forma se garantiza que la búsqueda de la raíz se encuentre dentro de un intervalo acotado.

Si de los puntos a,b y c hay dos que coinciden, entonces la estimación se haría haciendo una interpolación lineal de la función (método de la secante).

En consecuencia, la estimación en un paso (j), se lleva a cabo en función de una interpolación (lineal o cuadrática) si se cumplen las condiciones siguientes[14], en caso contrario la estimación del paso (j) se llevaría por medio del método de la bisección ($x = \frac{b+c}{2}$):

i. $|\frac{P}{Q}| \geqslant \frac{3}{4}\,|c-b|$

(ver Figura 4.28)

ii.a Si en el paso previo (j-1) se llevó a cabo por medio de una bisección, permite la interpolación si se cumple:

ii.a.1 $|\frac{P}{Q}|_j < \frac{1}{2}\,|\frac{P}{Q}|_{j-1}$

ii.a.2 $|\frac{P}{Q}|_{j-1} > \frac{\delta}{2}$

ii.b Si en el paso previo (j-1) se llevó a cabo por medio de una interpolación, permite la interpolación si se cumple:

ii.b.1 $|\frac{P}{Q}|_j < \frac{1}{2}\,|\frac{P}{Q}|_{j-2}$

ii.b.2 $|\frac{P}{Q}|_{j-2} > \frac{\delta}{2}$

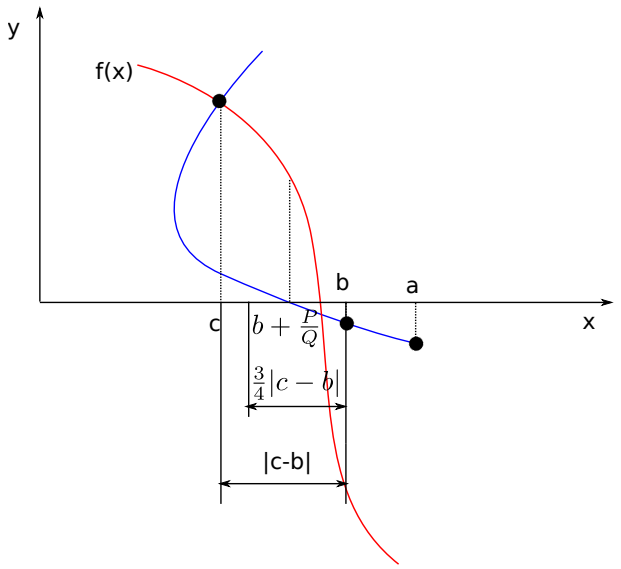



Figura 4.28: Condición i. del método de Brent

# Capítulo 5

# Ejemplos de aplicación

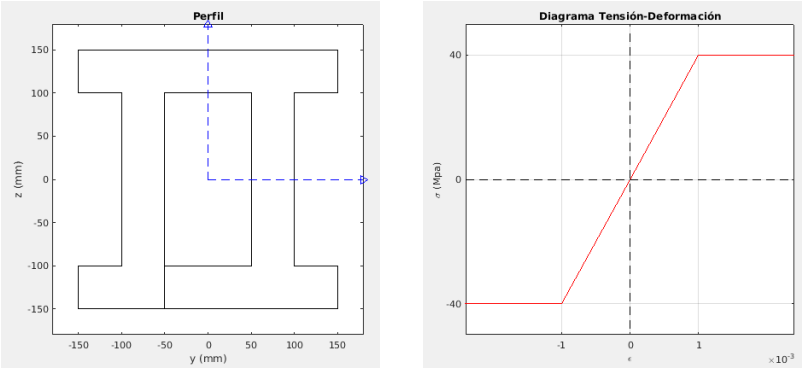
## 5.1. Caracterización del comportamiento de una sección

Para caracterizar el comportamiento de una sección sometida a flexión pura y compuesta, tras definir su geometría y material, mostraremos el diagrama Momento - Curvatura con el que podemos visualizar la capacidad de rotación de la sección ante un momento flector y posteriormente su diagrama de interacción Axil - Momento, que muestra todas las combinaciones de esfuerzos de axil y momento que se pueden dar simultáneamente en la sección. Finalmente, a modo de ejemplo, se tomará una de estas combinaciones para ver cual sería su distribución de tensiones ante la solicitación seleccionada.

### 5.1.1. Definición de la sección

```
Introducir caracteres entre comillas simple
Tipo de sección: 'cajon'
cajon: sección en cajón
distancia entre caras extremas de las almas, b (mm): 200
saliente alas, b1 (mm): 50
espesor ala superior, tf1 (mm): 50
altura almas, hw (mm): 200
espesor almas, tw (mm): 50
espesor ala inferior, tf2 (mm): 50

Tipo de material:'G1'
Tensión en el límite elástico (MPa): 40
Módulo de Young (MPa): 40e3
```




(a) (b)

Figura 5.1: Geometría y material de la sección

### 5.1.2. Diagrama Momento - Curvatura

```
Introducir número asociado al problema: 1
 *** RESULTADOS ***
Momento elástico: 144.4444 kN*m
Curvatura(M=Me): 0.006667 rad/m
Momento plástico: 190.0000 kN*m
Factor de forma: 1.315385
```

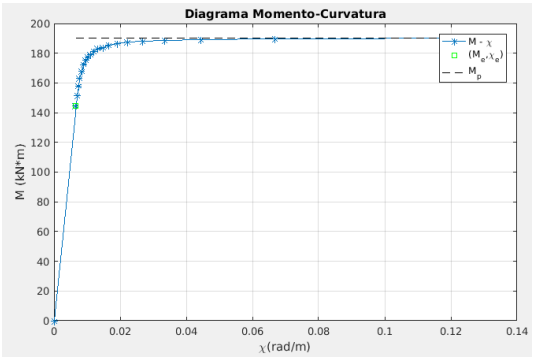



Figura 5.2: Diagrama Momento - Curvaura

### 5.1.3. Diagrama interacción Axil - Momento

```
Introducir número asociado al problema: 2
2: Cálculo del diagrama de interacción de la sección
N‘p - M‘p
Introduzca
n: axil de compresión
p: axil de tracción
SignoN: ’n’
*** RESULTADOS ***
Axil plástico (Mp=0): -2000.0000 kN
Momento plástico (Np=0): 190.0000 kN*m
```

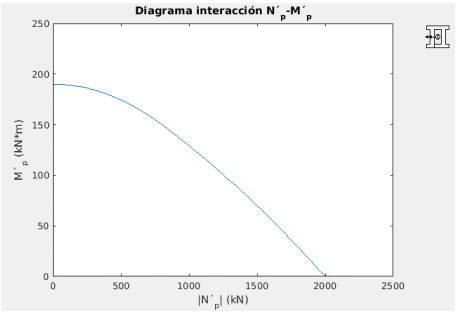



Figura 5.3: Diagrama interacción Axil - Momento ($M > 0$, $N < 0$)

### 5.1.4. Plastificación de una sección

El diagrama de tensiones y deformaciones para una solicitación de: N=-1000 kN; M= 120 $kN\,m$

```
Introducir número asociado al problema: 3
  3: Cálculo del diagrama de tensiones dada la
     solicitación de la sección (N,M)
 Introducir valores de la solicitación:
 Valor del axil, N (kN): -1000
 Valor del momento, M (kN·m): 120
*** RESULTADOS ***
Ley de Tensiones
z(mm) sigma(MPa)
0.0000 32.2790
53.0527 0.0000
118.7954 -40.0000
300.0000 -40.0000
Curvatura= 0.015211 rad/m
```

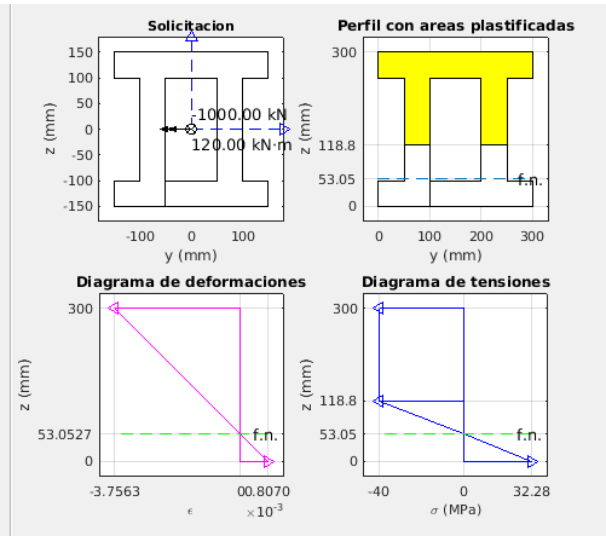



Figura 5.4: Distribución de tensiones y deformaciones

## 5.2. Casos de aplicación

En este apartado se utiliza el programa como una caja de herramientas (toolbox). De manera que se hará uso de diferentes funciones para resolver problemas específicos. Estos problemas están enfocados en el cálculo de la distribución de tensiones residuales (tras la descarga), cálculo de movimientos y cálculo de cargas críticas en estructuras isostáticas.

En el apéndice B, se pueden ver las descripciones de las funciones que permiten el uso de éstas. Para poder utilizar estas funciones se tienen que añadir los directorios que contienen a las funciones, mediante *addDirectorios*.

### Caso 1 Comparar diagramas $M - \chi$ de diferentes secciones

Puede ser interesante comparar los diagramas de Momento - Curvatura normalizados, para comparar el comportamiento plástico de diferentes secciones como muestra la Figura 5.5. Para ello utilizamos para cada sección la función *problFlexion.N_diagramaMC()* y dividimos los momentos por el momento elástico ($M_e$) y las curvaturas por la curvatura asociada al momento elástico ($\chi_e$).

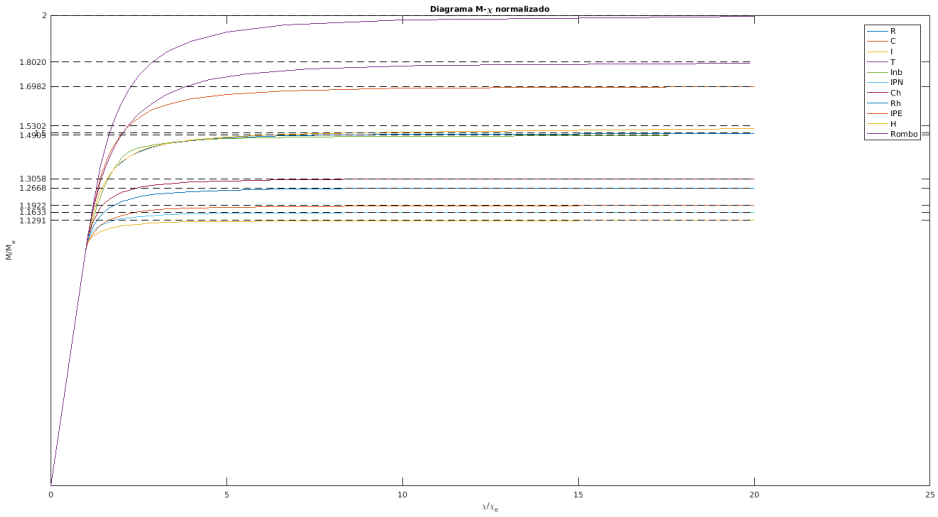



Figura 5.5: Comparación de diagramas $M/M_e - \chi/\chi_e$

### Caso 2 Calcular distribución de tensiones y curvatura residuales

*(Problema FlexPura 13 [13])*

*Sea la sección que se muestra en la figura, propiedades del material:* $\sigma_p = 20\,MPa$, $E = 20\,10^3 MPa$. *Se busca: a) Calcular el valor del momento flector Ma que plastifica la cabeza superior. b) Determinar la curvatura producida por el momento Ma. c) Representar el diagrama de tensiones residuales de la sección después de descargar el momento Ma. c) Determinar la curvatura residual.*

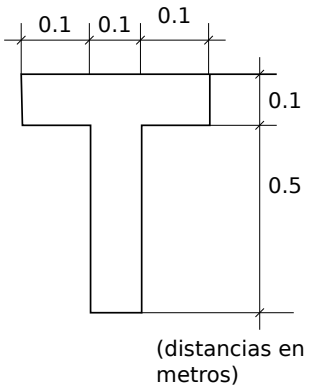



Figura 5.6: Sección problema FlexPura 13 [13]

```
addDirectorios% Añadir directorios
%Definición de la geometría
p=generacion_Modelo.verticesPoly('T',[600,300,100,100]);
zG=propSeccion.propGeom(p,'zG'); yG=propSeccion.propGeom(p,'yG');
p=problGeom.cambioSistCoord(p,-yG,-zG,0);
p=p*1e-3;
% Definición del material
ley_mat=generacion_Modelo.comportMaterial('G1',[20,20e3]);
ley_mat(:,2)=ley_mat(:,2)*1e3;
```

a) Calcular Ma:

```
% Valor del momento que plastifica cabeza (Ma)
Ma=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,0.1,'c',1,0)
```

$M_a = 253{,}3333\, kN \cdot m$

b) Curvatura que produce Ma ($\chi_a$):

```
% Curvatura que produce Ma
[zfn,chi]=problFlexion.NAndM_zfnAndChi(p,ley_mat,0,Ma);
```

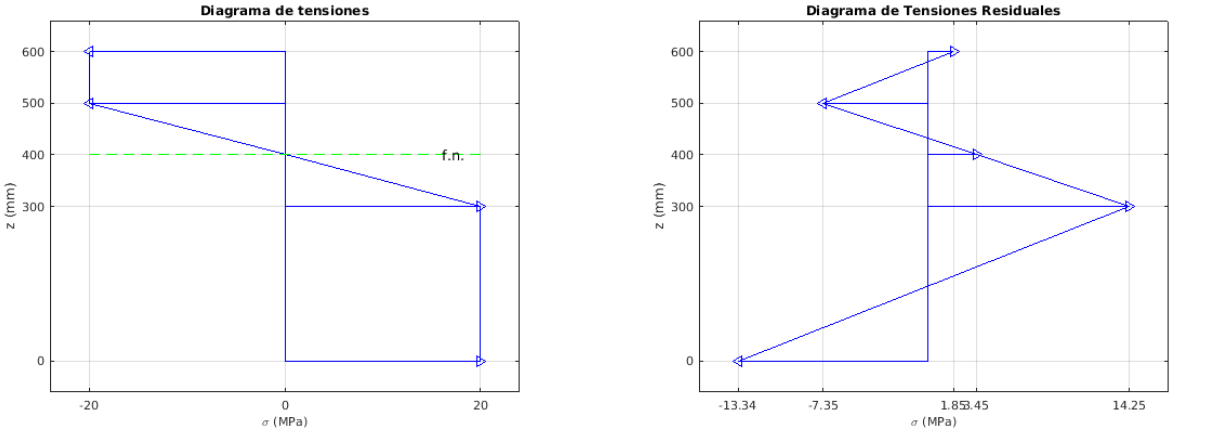
$\chi_a = 0{,}01\, rad/m$

c) Distribución de tensiones residuales (ver Figura 5.7b), tras la descarga de Ma y d) curvatura residual $\chi_{res}$:

```
[z_,epsz_,sigz_]=problFlexion.zfnAndChi_EpszAndSigz(p,ley_mat,zfn,chi);
% Calcular propiedades de la sección
Atot=propSeccion.propGeom(p,'area');
Iy=propSeccion.propGeom(p,'IGy');
zG=propSeccion.propGeom(p,'zG');
E=propSeccion.propMaterial(ley_mat,'E');
chiD=(-Ma)/(E*Iy);% el signo negativo, ya que aplicado de forma opuesta
% al de carga
zD_=z_+zfn;% referir los puntos a la posición cdg
% para calcular tensiones descarga
sigDescarga=-N/Atot*ones(size(z_))-chiD*zD_*E;
epsDescarga=chiD*zD_;
sigR=sigz_+sigDescarga;
```

```
epsR=epsz_+epsDescarga;
chiR=chi+chiD;
figure(1)
salidaDatos.mostrar_ley_tensiones([sigz_*1e-3,z_*1e3])
figure(2)
salidaDatos.mostrar_ley_tensiones([sigR*1e-3,z_*1e3])
title('Diagrama de Tensiones Residuales')
```

$\chi_{res} = 0{,}0054\, rad/m$




(a) Tras la carga

(b) Tras la descarga (distribución de tensiones residual)

Figura 5.7: Distribución de tensiones

## Caso 3 Calcular movimientos en estructuras isostáticas

Para estructuras isostáticas podemos calcular las leyes de esfuerzos de forma directa para los diferentes regímenes, ya que este cálculo no depende del material. Por tanto, a partir de la ley de momentos se puede obtener la ley de curvaturas de la estructura y con ello usando las fórmulas de Bresse se pueden calcular los movimientos [3]. Dado que el programa solo abarca los casos de flexión pura y compuesta, sólo sería válido para los casos en los que se pudieran despreciar los efectos del cortante en la deformación y plastificación de la sección.

*(Problema 3.13, Plasticidad Abreviada -Rafael Fernández Díaz-Munío [3])*

*Dada la estructura con una sección rectangular como se muestra en la Figura 5.8 y propiedades del material:* $\sigma_p = 200\, kp/cm^2$, $E = 2\,10^5 kp/cm^2$ *Se pide: a) Calcular el máximo valor de carga P para que ninguna sección plastifique más de la mitad. b) Determinar el giro y flecha producidos por dicha carga en el extremo. c) Determinar el giro y la flecha remanentes tras la descarga.*

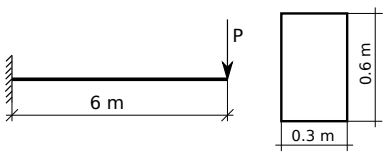



Figura 5.8: Estructura problema Caso 3 (Problema 3.13 [3])

```
%Definición de la geometría
p=generacion_Modelo.verticesPoly('R',[300,600]);
zG=propSeccion.propGeom(p,'zG'); yG=propSeccion.propGeom(p,'yG');
```

```
p=problGeom.cambioSistCoord(p,-yG,-zG,0);
p=p*1e-3;

% Definición del material
ley_mat=generacion_Modelo.comportMaterial('G1',[20,20e3]);
ley_mat(:,2)=ley_mat(:,2)*1e3;
```

a) El valor de carga P:

Como la ley de momentos es:

$M(x) = -P \cdot (6 - x)$

El mayor momento será en la sección de empotramiento. El momento asociado a una plastificación de la sección de la mitad es una profundidad de plastificación de 0.15 m:

```
[M,chi,zfn]=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,0.15,'c',2,0)
```

$M = -495\,kN \cdot m$

Por tanto, $P = \frac{M}{L} = -82{,}5\,kN$

b) Cálculo del giro y la flecha en el extremo A, se calcula mediante:

```
L=6;% longitud barra (m)
x=linspace(0,L,100);
% Ley de momentos
P=82.5;%kN
ley_M=@(x) -P*(L-x);
Marray=ley_M(x);
N=0;
% Ley de curvaturas
for i=1:length(Marray)  [zfn,chi(i)]=problFlexion.NAndM_zfnAndChi(p,ley_mat,N,Marray(i));
end
p_chi=[[0,0];[6,0];[x(end:-1:1)',chi(end:-1:1)']];
p_chi=p_chi(end:-1:1,:);% si chi(L)<0
areaA=p_chi;
titaA=propSeccion.propGeom(areaA,'area')%giro en A (rad)
areaA=problGeom.cambioSistCoord(areaA,-L,0,0);
vA=-propSeccion.propGeom(areaA,'Wz')%flecha en A (m)
```

De donde se obtiene:

$\theta_A = 0{,}014547\,rad$

$v_A = 0{,}059513\,m$

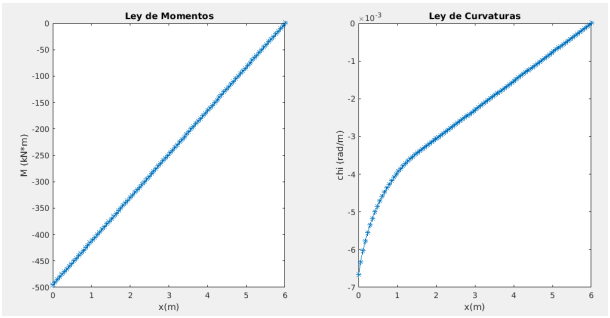



Figura 5.9: Caso 3 - Ley de momentos y de curvaturas

c) Giro y flecha tras la descarga del extremo A

```
% Calcular propiedades de la sección
Atot=propSeccion.propGeom(p,'area');
Iy=propSeccion.propGeom(p,'IGy');
zG=propSeccion.propGeom(p,'zG');
E=propSeccion.propMaterial(ley_mat,'E');
%%%% Ley de curvaturas Residuales
N=0;
for i=1:length(Marray)
  chiD=(-Marray(i))/(E*Iy);% el signo negativo, ya que aplicado de forma opuesta al de carga
  chiR(i)=chi(i)+chiD;
end
chi=chiR;
%%%%
p_chi=[[0,0];[6,0];[x(end:-1:1)',chi(end:-1:1)']];
p_chi=p_chi(end:-1:1,:);% si chi(L)<0
areaA=p_chi;
titaA=propSeccion.propGeom(areaA,'area')%giro en A (rad)
areaA=problGeom.cambioSistCoord(areaA,-L,0,0);
vA=-propSeccion.propGeom(areaA,'Wz')%flecha en A (m)
```

De donde se obtiene:

$\theta_A = 7{,}971132\,10^{-4}\,rad$

$v_A = 0{,}004513\,m$

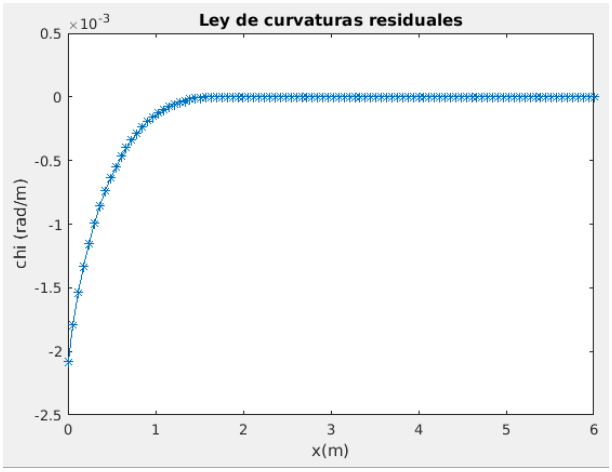



Figura 5.10: Caso 3 - Ley de Curvaturas Residuales

## Caso 4 Calcular cargas de colapso

*(Problema 4.5, Plasticidad Abreviada -Rafael Fernández Díaz-Munío [3])*

*Sea la estructura de la figura compuesta por la la columna AB de sección doble T (ver Figura 5.11) y por el pescante BC, que se considera infinitamente rígido. Se pide:*

*a) Calcular el valor de la carga P que produce el colapso de la estructura. b) Representar el diagrama de tensiones de la sección A cuando la carga P alcanza el 97 % del valor teórico de colapso del apartado anterior.*

*Propiedades del material:* $\sigma_p = 200\,kg/cm^2$, $E = 200\,10^3 kg/cm^2$

Primero se define la sección del problema:

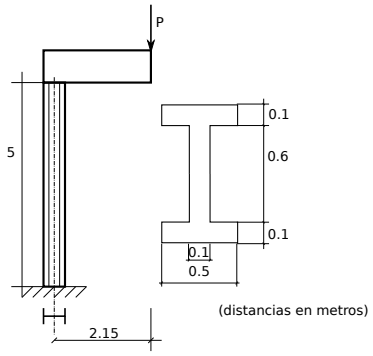



Figura 5.11: Estructura del Caso 4 (Problema 4.5 [3])

```
%Definición de la geometría
p=generacion_Modelo.verticesPoly('I',[800,500,100,100]);
zG=propSeccion.propGeom(p,'zG'); yG=propSeccion.propGeom(p,'yG');
p=problGeom.cambioSistCoord(p,-yG,-zG,0);
p=p*1e-3;

% Definición del material
ley_mat=generacion_Modelo.comportMaterial('G1',[20,20e3]);
ley_mat(:,2)=ley_mat(:,2)*1e3;
```

a) Valor de la carga (P) de colapso

Como la solicitación de la columna AB es:

N=P (de compresión) y $M = P \cdot 2{,}15$

La relación entre ambos esfuerzos es:

$M/N = 2{,}15$

Con ello determinamos el par de agotamiento asociado, sabiendo que

$N'_p < 0$ y $M'_p > 0$

Lo calculamos mediante:

```
% Para cierto valor de recta de carga, calcular el par de agotamiento
% donde m=M/N
[Np,Mp,zfnp,zpmax,signo_zpmax,condic]=problFlexion.NM_NpMp(p,ley_mat,-1,2.15)
```

De donde obtenemos:

$Np = -400{,}1198\,kN$

$Mp = 859{,}9880\,kN \cdot m$

Como P=N, $P = 400{,}1198\,kN$

b) El diagrama de tensiones de la sección A cuando $P = 0{,}97 \cdot P_{agotamiento}$

Lo calculamos, mediante:

```
% Definir solicitación
N=Np*0.97;%kN
M=Mp*0.97 ;% kN*m
[zfn,chi]=problFlexion.NAndM_zfnAndChi(p,ley_mat,N,M);
[z_,epsz_,sigz_]=problFlexion.zfnAndChi_EpszAndSigz(p,ley_mat,zfn,chi);
salidaDatos.mostrar_ley_tensiones([sigz_*1e-3,z_*1e3])
```

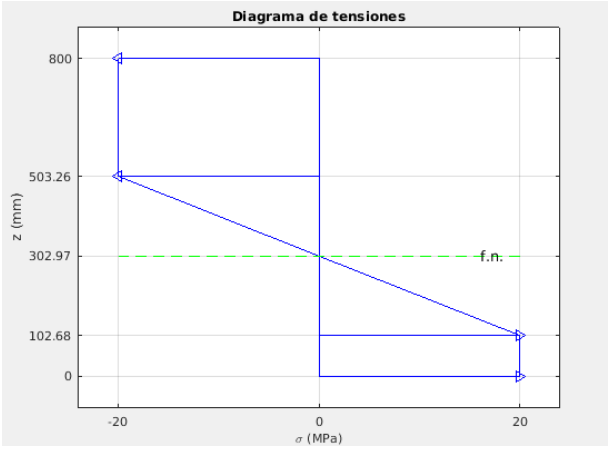



Figura 5.12: Caso 4 - Distribución de tensiones.

# Capítulo 6

# Conclusiones y trabajos futuros

## 6.1. Conclusiones

Aplicación resuelve problemas a nivel de sección, por tanto la implementación depende de la geometría, modelos de comportamiento del material de la sección y cómo está solicitada ésta. De manera que el dominio de la implementación es la que sigue:

- Se ha implementado para materiales perfectamente plástico y materiales que presentan endurecimiento tanto si tienen igual comportamiento a tracción y compresión como sino.
- Se ha implementado para secciones que pueden ser descritas por un polígono cóncavo, convexo o con hueco. De este último tipo se presenta la restricción de que los polígonos que forman un hueco han de ser convexos (debido a la función *clipArea*, con la que se resuelven las integrales dobles). No obstante, cubre la gran mayoría de las secciones que se consideran en elementos estructurales.
- Se ha implementado para secciones sometidas a flexión pura y compuesta.

El programa permite caracterizar el comportamiento de una sección sometida a flexión pura y compuesta mediante los diagramas Momento - Curvatura (con un axil constante), factor de forma y diagrama de interacción Axil - Momento. También permite representar a partir de la solicitación de la sección el reparto de tensiones y deformaciones.

Puesto que se puede representar el diagrama Momento - Curvatura asociado a un valor de axil, es posible estudiar la influencia del axil en la capacidad de rotación de la sección.

Haciendo uso de las funciones de este programa, se puede calcular: deformaciones y movimientos (tras la carga y la descarga) y cargas de colapso de una estructura isoestática donde sea válida la hipótesis en la que los efectos de los esfuerzos cortantes sean despreciables.

## 6.2. Trabajos futuros

Se propone como continuación de este trabajo, siguiendo la línea del libro de Plasticidad Abreviada de Rafael Fernández Díaz-Munío [3], implementar las funciones necesarias para estudiar el comportamiento de una sección sometida a flexión simple y el caso más general, en el que la sección esté sometida a una combinación de esfuerzos de axil, momento y cortante. Por último, se propone el estudio de secciones compuestas por más de un material.

# Apéndice A

# Guía de Usuario

## A.1. Descripción del programa

El programa *APlastico* resuelve tres tipos de problemas de flexión pura y compuesta. Estos problemas son:
1. Calcular el diagrama Momento - Curvatura de una sección.
2. Calcular el diagrama de interacción Axil - Momento de una sección.
3. Calcular la distribución de tensiones y deformaciones dada la solicitación de una sección.

Además, se pueden resolver de tres modos:
- Modo de entrada y salida de datos por Interfaz Gráfica de Usuario (GUI).
- Modo entrada y salida de datos por consola.
- Modo entrada y salida de datos por archivo.

### A.1.1. Desarrollo del programa

El desarrollo del programa se divide en las siguientes tareas:
1. Iniciar el programa
2. Generación del modelo:
   a) Especificar geometría
   b) Especificar material
3. Elegir tipo de problema y especificar los valores propios del problema
4. Generación de los resultados
5. Presentación de los resultados

### A.1.2. Datos de entrada

Con los datos de entrada generamos el modelo. Estos datos son:

1. Tipo de problema

   1: Cálculo del diagrama Momento - Curvatura y factor de forma.

   2: Cálculo del diagrama Interacción $N'_p - M'_p$.

   3: Cálculo de la ley de tensiones dada la solicitación de la sección.

2. Valores propios del problema

*a*) Problema 1: Diagrama Momento - Curvatura

No requiere introducir ningún valor adicional.

*b*) Problema 2: Diagrama Interacción $N'_p - M'_p$

Indicar si el axil es de tracción o compresión.

*c*) Problema 3: Ley de tensiones $\sigma(z)$

1) Valor del axil, N($kN$). Siendo N>0 tracciones.
2) Valor del momento, M($kN \cdot m$). Siendo M>0, momento que produce compresiones en la cara superior.

3. Tipo y dimensiones sección:

Se introducen los valores de las dimensiones en mm.

*a*) R: Sección rectangular

b: ancho; h: canto

*b*) C: Sección circular

r: radio

*c*) I: Sección doble T

h: altura total; bf: ancho ala; tf: espesor ala; tw: espesor alma

*d*) Inb: Sección doble T no bisimétrica

h: altura total; bf1: ancho ala superior; tf1: espesor ala superior; tw: espesor alma; bf2: ancho ala inferior; tf2: espesor ala inferior

*e*) IPN: Sección IPN

h: altura total perfil; b: ancho alas; r1: radio entre ala y alma; tf: espesor ala; r2: radio ala; d: altura alma

*f*) IPE: Sección IPE

h: altura total perfil; bf: ancho alas; tw: espesor alma; tf: espesor ala; r: radio ala; d: altura alma

*g*) H: Sección H

hf: ancho ala; b: altura total; tf: espesor ala; tw: espesor alma

*h*) T: Sección en T

h: altura total; bf: ancho ala; tf: espesor ala; tw: espesor alma

*i*) Rombo: Sección rombo

d1: diagonal horizontal; d2: diagonal vertical

*j*) Rh: Sección rectangular con hueco

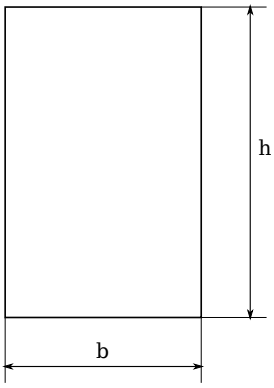
b: ancho; h: canto; e: espesor

*k*) Ch: Sección circular con hueco

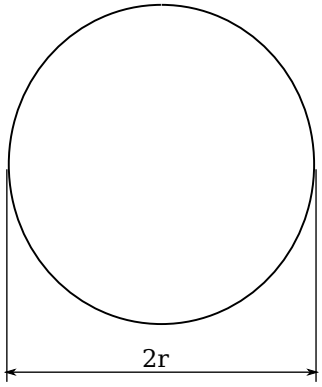
r: radio; e: espesor

*l*) cajon: Sección en cajón

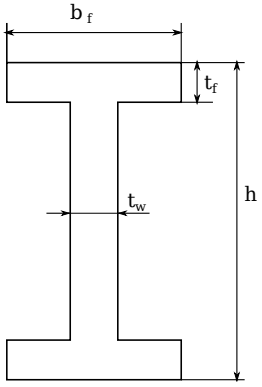
b: distancia entre caras extremas de las almas; b1: saliente alas; tf1: espesor ala superior; hw: altura almas; tw: espesor almas; tf2: espesor ala inferior



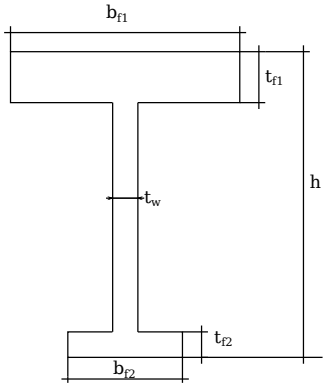

(a) Sección Rectangular



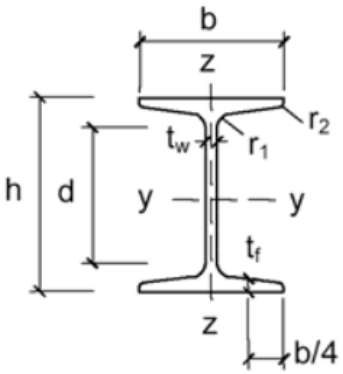

(b) Sección circular



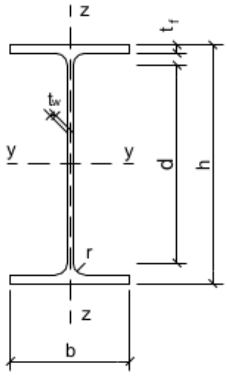

(c) Sección doble T



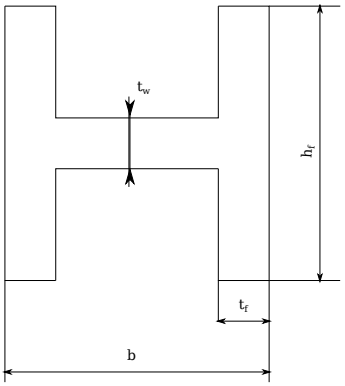

(d) Sección doble T no bisimétrica



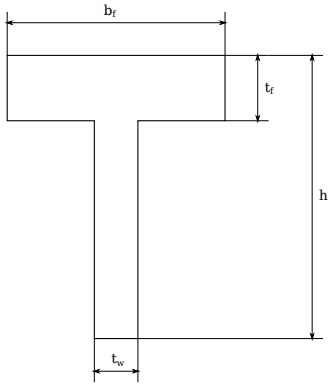

(e) Sección IPN



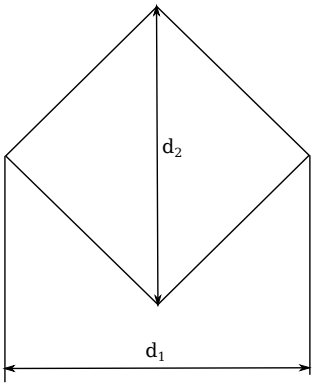

(f) Sección IPE



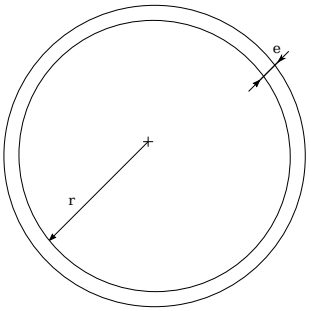

(g) Sección en H



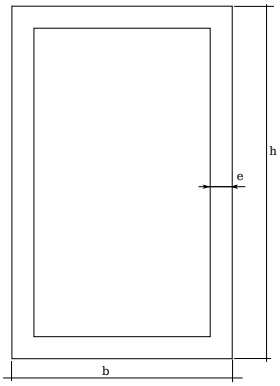

(h) Sección en T



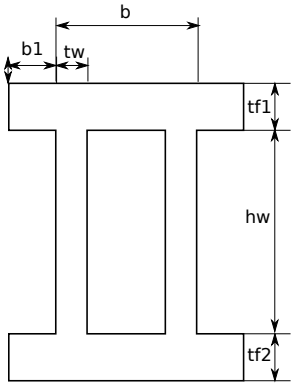

(i) Sección rombo




(j) Seccińcircular con hueco




(k) Sección rectangular con hueco




(l) Sección en cajón

Figura A.1: Secciones con sus dimensiones

4. Tipo y valores del material

   Los valores tanto de tensión como el módulo de Young se introducen en MPa.

   *a*) G1: Material perfectamente plástico con igual comportamiento a tracción que a compresión

   $\sigma_e$ : tensión en el límite elástico ($\sigma > 0$)

   $E$ : módulo de Young

   *b*) G2: Material perfectamente plástico con diferente comportamiento a tracción que compresión

   $E$ : módulo de Young

$\sigma'_p$ : tensión en el límite elástico a compresión ($\sigma < 0$)

$\sigma_p$ :tensión en el límite elástico a tracción ($\sigma > 0$)

*c*) G3: Material con endurecimiento

$\epsilon_1$ : deformación unitaria en el límite elástico ($\epsilon > 0$)

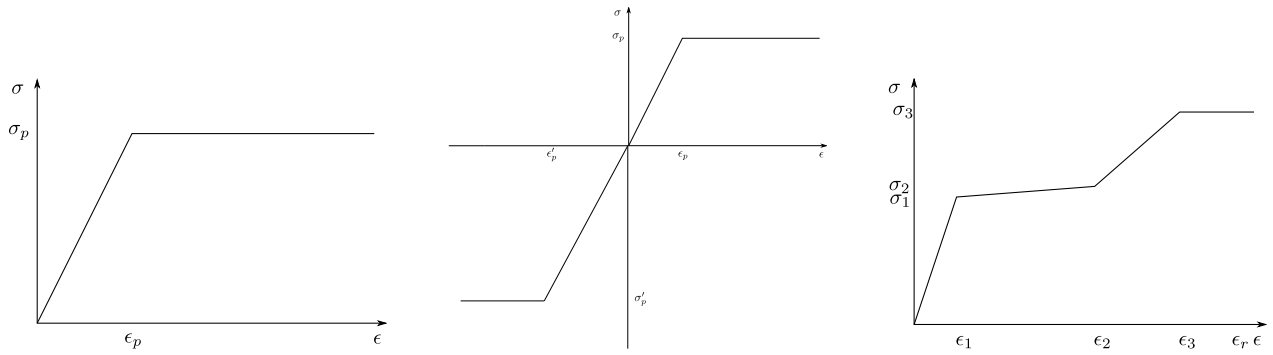
$\sigma_1$ : tensión en el límite elástico ($\sigma > 0$)

$\epsilon_2$ : deformación unitaria en el comienzo del endurecimiento ($\epsilon > 0$)

$\sigma_2$ : tensión en el comienzo del endurecimiento ($\sigma > 0$)

$\epsilon_3$ : deformación unitaria en el final del endurecimiento ($\epsilon > 0$)

$\sigma_3$ : tensión máxima ($\sigma > 0$)




(a) Material perfectamente plástico $|\sigma_{pc}| = \sigma_{pt}$

(b) Material perfectamente plástico $|\sigma_{pc}| \neq \sigma_{pt}$

(c) Material con endurecimiento

Figura A.2: Modelos de material con sus valores característicos

### A.1.3. Datos de salida

1. Problema 1: Cálculo del Diagrama Momento - Curvatura

   Diagrama Momento ($kN \cdot m$)-Curvatura ($rad/m$)

   Momento elástico ($M_e$) y la curvatura cuando el momento es elástico ($\chi_e$).

   Momento plástico ($M_p$).

2. Problema 2: Diagrama Interacción $N'_p - M'_p$ :

   Diagrama $N'_p - M'_p(kN - kN \cdot m)$

   $N_p$ cuando $M'_p = 0$

   $M_p$cuando $N'_p = 0$

3. Problema 3: Ley de tensiones dada una solicitación (N, M)

   Ley de tensiones, [$z$, $\sigma(z)$] (mm, MPa)

   Ley de deformaciones, [$z$, $\epsilon(z)$] (mm, )

### A.1.4. Visualización del modelo

Para garantizar que se está resolviendo el problema que buscamos, el programa muestra el perfil y material escogidos y la solicitación si se trata del problema de dada una solicitación, calcular la ley de tensiones (problema 3).

El perfil se muestra con un sistema de referencia cartesiano cuyo origen se sitúa en el centro de gravedad de la sección, como muestra la siguiente figura:

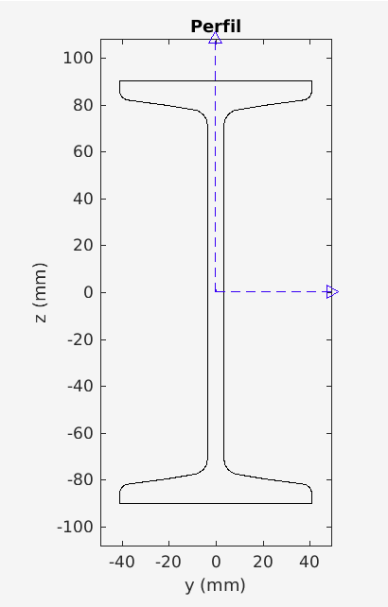



Figura A.3: Visualización del perfil

La visualización de la solicitación sigue la siguiente simbología:

● : Axil positivo (tracción)

⊗ : Axil negativo (compresión)

◄◄: Momento positivo (produce compresión en la cara superior)

►►: Momento negativo (produce tracción en la cara superior)

Por ejemplo para una sección rectangular con un axil de $-10^4 kN$ y un momento de $-1{,}5 \cdot 10^3\, kN \cdot m$

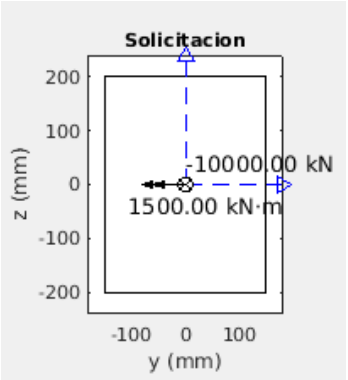



Figura A.4: Visualización de solicitaciones con $M < 0$ y $N < 0$

### A.1.5. Presentación de los resultados

Los resultados se muestran de forma gráfica y texto. De manera que para cada tipo de problema se muestra:

#### A.1.5.1. Gráficos

**Problema 1**

Diagrama Momento - Curvatura

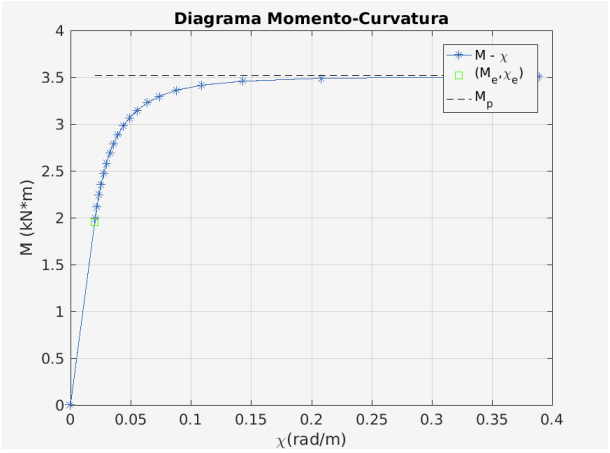
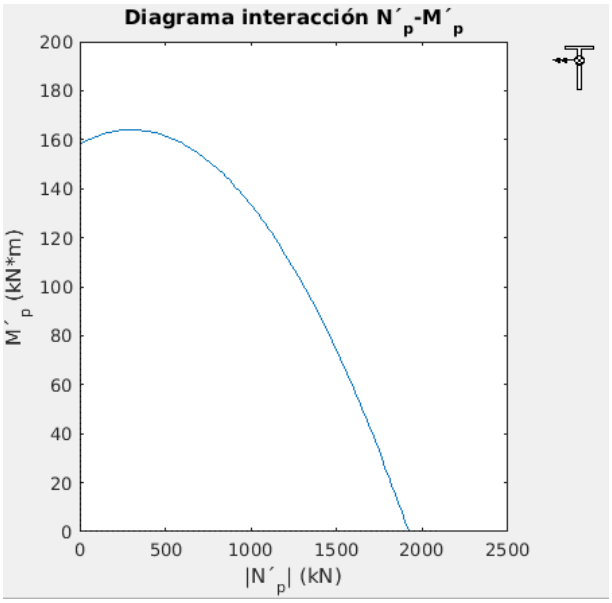


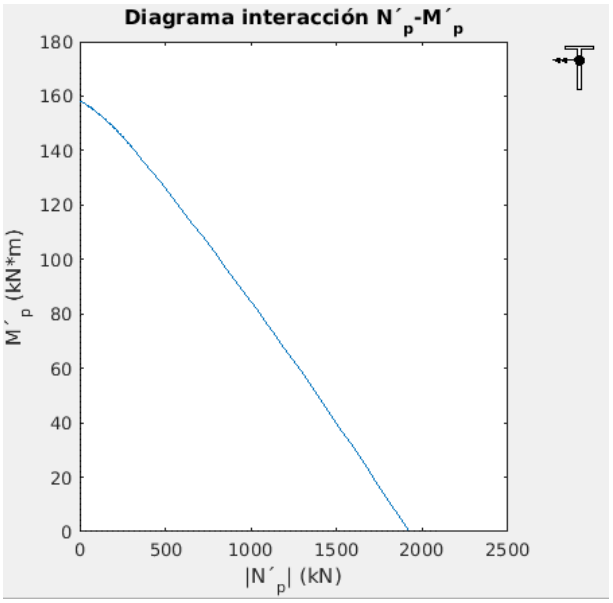

Figura A.5: Diagrama $M - \chi$. $(M > 0)$

**Problema 2**

Tipo de gráficos que obtenemos según los valores propios del problema escogido:




(a) Diagrama $N'_p - M'_p$ con axil de compresión




(b) Diagrama $N'_p - M'_p$ con axil de tracción

Figura A.6: Salida cálculo diagrama de interacción $N'_p - M'_p$

**Problema 3**

La salida de este problema muestra los gráficos

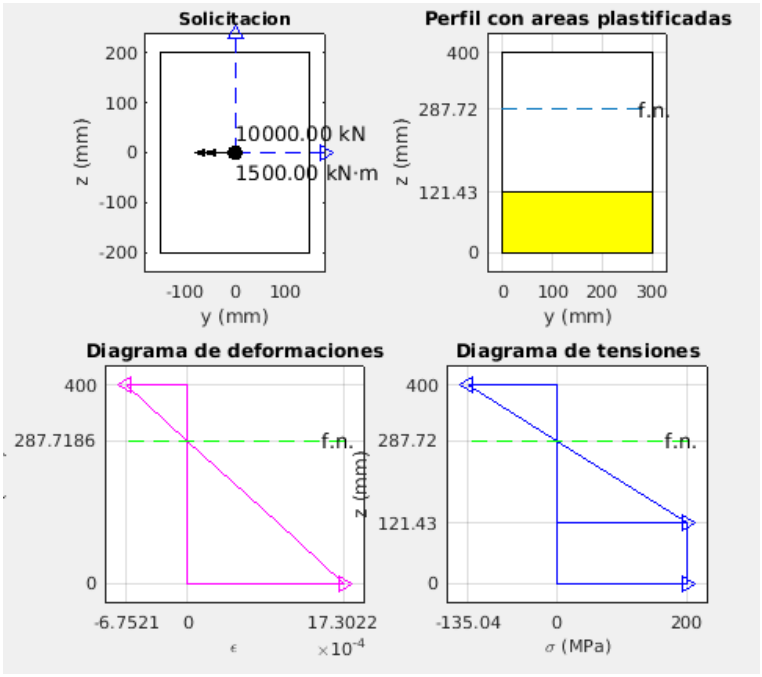



Figura A.7: Salida cálculo ley de tensiones

En el gráfico de perfil con áreas plastificadas muestra en amarillo las regiones de la sección que se encuentra en régimen elasto-plástico.

Se muestran estos gráficos con z=0 en el extremo inferior de la sección para facilitar su lectura.

### A.1.5.2. Generación de un informe

**Problema 1**

Ejemplo de salida del problema de cálculo de diagrama Momento - Curvatura:

```
+------ Cálculo Diagrama Momento - Curvatura ------+
 *** VALORES ENTRADA ***
 distancias(mm), tensiones(MPa), E(Mpa)
Tipo de seción: T
Dimensiones sección:
  h: 70.00 mm
  bf: 70.00 mm
  tf: 8.00 mm
  tw: 8.00 mm
Tipo de Material: G2
Valores del material:
Tensiones (MPa)
  E(MPa): 300000.000000
  sig_pc(Mpa): -900.000000
  sig_pt(MPa): 300.000000
 *** RESULTADOS ***
Momento elástico: 2.9303 kN*m
Curvatura(M=Me): 0.020177 rad/m
```

```
Momento plástico: 5.8778 kN*m
Factor de forma: 2.005902
 Diagrama Momento-Curvatura
      zp(mm)     chi(rad/m)        M(kN*m)
      0.0000         0.0202         2.9303
      3.3114         0.0216         3.1235
      ...
```

**Problema 2**

Ejemplo de salida del problema de cálculo de diagrama de interacción:

```
+------ Cálculo Diagrama Interacción N‘p-M‘p ------+
 *** VALORES ENTRADA ***
 distancias(mm), tensiones(MPa), E(Mpa)
Tipo de sección: T
Dimensiones sección:
  h: 70.00 mm
  bf: 70.00 mm
  tf: 8.00 mm
  tw: 8.00 mm
Tipo de Material: G2
Valores del material:
Tensiones (MPa)
  E(MPa): 300000.000000
  sig_pc(Mpa): -900.000000
  sig_pt(MPa): 300.000000
Signo del axil: p
Signo del momento: p
 *** RESULTADOS ***
Axil plástico (Mp=0): 316.8000 kN
Momento plástico (Np=0): 5.8778 kN*m
Diagrama Interacción N‘p-M´p
     zfnp(mm)       N´p(kN)     M‘p(kn*m)
     16.6680        0.0000        5.8778
     16.7061        3.2000        5.8244
     16.7442        6.4000        5.7709
     16.7823        9.6000        5.7172
     16.8203       12.8000        5.6635
     ...
```

**Problema 3**

Ejemplo de salida del problema de cálculo de diagrama de tensión y deformación dada la solicitación de una sección:

```
 +------ Cálculo Ley de Tensiones ------+
 *** VALORES ENTRADA ***
 distancias(mm), tensiones(MPa), E(Mpa)
```

```
Tipo de seción: T
Dimensiones sección:
  h: 70.00
  bf: 70.00
  tf: 8.00
  tw: 8.00
Tipo de Material: G2
Valores del material:
Tensiones (MPa)
  E(MPa): 300000.000000
  sig_pc(Mpa): -900.000000
  sig_pt(MPa): 300.000000
Solicitación de la sección
Axil: -847.64 kN·m
Momento: 4.40 kN·m
 *** RESULTADOS ***
Ley de Tensiones
       z(mm)     sigma(MPa)
      0.0000       300.0000
      1.2547       300.0000
      5.9794         0.0000
     20.1536      -900.0000
     70.0000      -900.0000
 Curvatura= 0.211652 rad/m
```

### A.1.6. Guardar resultados

En los modos de E/S de datos por consola y archivo existe la opción de guardar los resultados. Estos resultados se almacenan automáticamente en la carpeta *Output_Files/.*

Si por ejemplo se ha elegido resolver el problema 2, se generarán los siguientes archivos*:*

*<nombre_caso>_perfil.txt*, donde se muestra la geometría de la sección.

*<nombre_caso>_leyConstitutiva.txt*, donde se muestra la ley constitutiva del material.

*<nombre_caso>_dInteracc.txt*, donde se muestra el diagrama interacción.

*<nombre_caso>_dinteracc.txt*, donde se obtiene el informe en formato texto del problema.

## A.2. Modo E/S por medio de GUI

### A.2.1. Inicio

Para iniciar el programa en modo GUI, estando dentro de la carpeta APlastico, introducimos:

```
main_modoGUI
```

Una vez introducido el comando, aparecerá la siguiente interfaz en la que se pueden distinguir las siguientes partes:

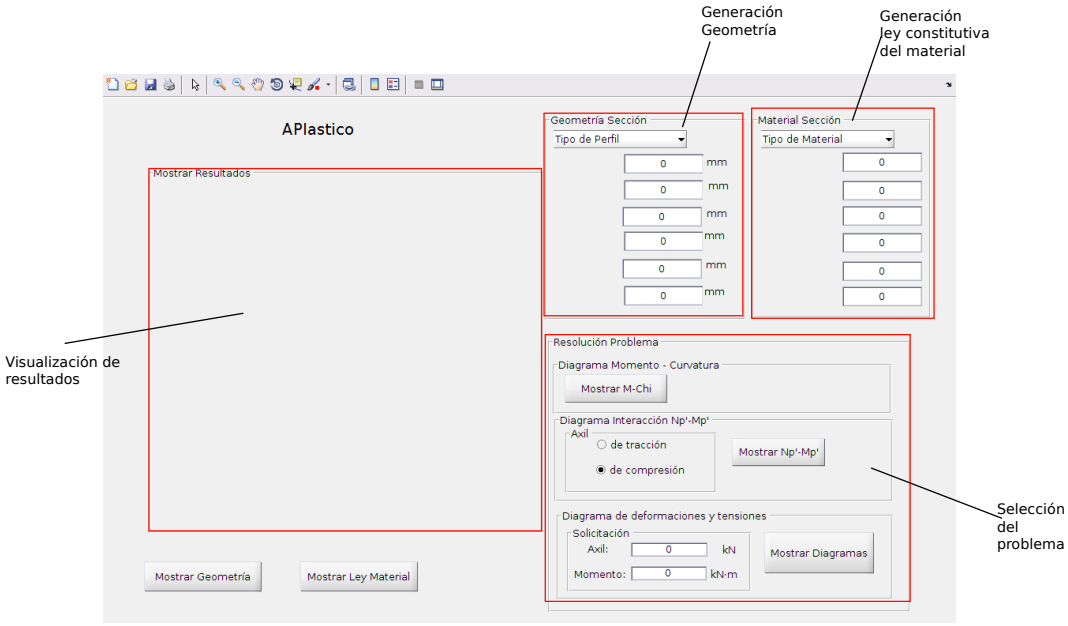



Figura A.8: Interfaz de usuario

## A.3. Modo E/S por consola

### A.3.1. Inicio

Para acceder al programa en modo consola, introducimos el comando:

```
>> programa_APlastico
```

### A.3.2. Generación del modelo

#### A.3.2.1. Generación de la geometría

Es importante considerar en esta parte que los valores con caracteres (a-z, A-Z) se introducen entre comillas simples y los valores numéricos sin comillas.

Ejemplo de entrada de datos para una sección rectangular:

```
Introducir caracteres entre comillas simple
Tipo de sección: 'R'
R: sección rectangular
ancho, b (mm): 400
canto, h (mm): 900
```

#### A.3.2.2. Generación del modelo del material

Es importante considerar en esta parte que los valores con caracteres (a-z, A-Z) se introducen entre comillas simples y los valores numéricos sin comillas.

Ejemplo de entrada de valores para la generación de un modelo de material perfectamente plástico con $|\sigma_{pc}| = \sigma_{pt}$ :

```
Introducir tipo de comportamiento del material del modelo
```

```
G1: Material perfectamente plástico con igual comportamiento a tracción y compresión
G2: Material perfectamente plástico con diferentecomportamiento a tracción y compresión
G3: Material con endurecimiento con igual comportamiento atracción y compresión
Tipo de material:'G1'
Tensión en el límite elástico (MPa): 300
Módulo de Young (MPa): 300E3
```

### A.3.3. Introducción de los valores propios del problema

Es importante considerar en esta parte que los valores con caracteres (a-z, A-Z) se introducen entre comillas simples y los valores numéricos sin comillas.

Ejemplo de entrada de datos para obtener un diagrama de interacción con axil de compresión:

```
Introducir número asociado al problema: 2
  2: Cálculo del diagrama de interacción de la sección
     N‘p - M‘p
 Introduzca
   n: axil de compresión
   p: axil de tracción
SignoN: 'n'
```

### A.3.4. Guardar resultados

Introducimos:

- 1: queremos guardar el caso.
- 0: no queremos guardar el caso (solo muestra resultados por consola).

En el caso de querer almacenar los resultados, asignamos un nombre al caso (entre comillas simples, sin extensión).

Ejemplo de entrada de datos:

```
¿Desea guardar el caso?
Pulse 1 si sí y 0 en caso negativo
Guardar resultados:1
Asignar un nombre al caso(entre comillas simples:)
Nombre:'caso_ejemplo'
```

### A.3.5. Presentación de resultados

Los resultados se muestran de forma gráfica y texto.

## A.4. Modo E/S por medio de archivo

En este modo, a diferencia de los anteriores, tenemos que primero definir la geometría, material y especificar los valores del problema antes de iniciar el programa. Para ello se especifican los valores en los archivos almacenados en la carpeta *Input_Files/:*

*input_casos.csv*, donde se especifican los casos que se quieren correr.

*input_materiales.csv*, donde se especifica el material.

*input_problemas.csv*, donde se especifican los tipos de problemas y sus valores correspondientes.

*input_secciones.csv*, donde se especifica los diferentes tipos de sección y sus dimensiones que se quieren considerar.

## A.4.1. Generación del modelo

### A.4.1.1. Generación de la geometría

Para generar el modelo geométrico de la sección hay que especificar el tipo y sus dimensiones características. Para ello en el archivo *input_secciones.csv* hay que rellenar los siguientes campos, separados por comas:

*ID_secc*: identificación de una sección.

*tipoSecc*: forma de la sección (rectangular, doble T,...)

*dim1,..,dim6*: dimensiones de la sección. Se recomienda asignar nan(not a number) en los campos vacíos.

Por ejemplo, a una sección T con dimensiones: h=70 mm, $b_f = 70\,mm$, $t_f = 8\,mm$ y $t_w = 8\,mm$,y que le asignamos la identificación SECC4, sería:

```
ID_secc,tipoSecc,dim1,dim2,dim3,dim4,dim5,dim6
SECC4,T,70,70,8,8,nan,nan
...
```

### A.4.1.2. Generación del modelo del material

Para generar el modelo material de la sección hay que especificar el tipo de material y sus valores característicos. Para ello en el archivo *input_materiales.csv* hay que rellenar los siguientes campos, separados por comas:

*ID_Mat*: identificación de una sección.

*tipoMat*: forma de la sección (rectangular, doble T,...)

*val1,..,val6*: dimensiones de la sección. Se recomienda asignar nan(not a number) en los campos vacíos.

Por ejemplo, para un material perfectamente plástico con diferente comportamiento a tracción y compresión (G2) con valores: $E = 200 \cdot 10^3 MPa$, $\sigma_{pc} = -400 MPa$ y $\sigma_{pt} = 200 MPa$, y al que le denominamos MAT2, sería:

```
ID_Mat,tipoMat,val1,val2,val3,val4,val5,val6  MAT2,G2,200E3,-400,200,nan,nan,nan
```

Se recomienda introducir *nan* a los campos vacíos.

## A.4.2. Introducción de los valores propios del problema

Para especificar un problema se ha de rellenar los siguientes campos en el archivo *input_problemas.csv:*

*ID_problema*: idenificación de un problema.

*tipo_problema*: <1 (calcular diagrama $M - \chi$) | 2 (calcular diagrama $N'_p - M'_p$) | 3 (calcular $\sigma(z)$ y $\epsilon(z)$)>

*signoN*: si el axil es de compresión o tracción, <n(compresión) | p(tracción)>

$N$: valor del axil.

$M$: valor del momento.

Sea P1, para calcular el diagrama Momento-Curvatura.

P2, para resolver el diagrama interacción con axil de compresión.

P4, para calcular la ley de tensiones y deformaciones de una sección con un axil de -847.64 kN y un momento de 4.4 $kN \cdot m$.

```
ID_problema,tipo_problema,signoN,N,M
P1,1, ,nan,nan
P2,2,n,nan,nan
P4,3, ,-847.64,4.4
```

Si el campo de signoN es vacío, se ha de introducir un espacio. Si los campos N, M son vacíos se ha de introducir *nan.*

### A.4.3. Definir un caso

Un caso está definido por una geometría, que ha sido especificada en *input_secciones.csv*, una ley constitutiva del material, especificada en *input_materiales.csv*, y el tipo de problema y sus valores, determinados en *input_problemas.csv*. Por lo que en el archivo *input_casos.csv* se pueden especificar cada uno de los casos rellenando los siguientes campos:

*ID_Caso*: identificador del caso.

*ID_Problema*: identificador del problema (determinado en *input_problemas.csv*).

*ID_Seccion*: identificador de la sección (determinado en *input_secciones.csv*)

*ID_Material*: identificador del material (determinado en *input_materiales.csv*)

Sea por ejemplo el caso con nombre SECC4_MAT2:

```
ID_Caso,ID_Problema,ID_Seccion,ID_Material
SECC4_MAT2,P1,SECC4,MAT2
```

### A.4.4. Inicio

Para ejecutar los casos previamente definidos en el archivo *input_casos.csv,* almacenado en la carpeta *Input_Files/*, se introduce el comando:

```
>> main_modoArchivo
```

### A.4.5. Presentación de resultados

Los resultados se almacenan de forma gráfica y texto en la carpeta *Output_Files/*.

# Apéndice B

# Código del programa

## B.1. Scripts de arranque del programa

### B.1.1. *main_modoConsola.m*

```
%%Programa Análisis Plástico en Flexión Pura y Compuesta
%%MODO 1: E/S de datos por consola
%

clear all
close all

%Añadir directorios
pathname1=fileparts('presentation/');
addpath(genpath(pathname1))

pathname2=fileparts('logic/');
addpath(genpath(pathname2))


guardarCaso=1;

%Entrada de datos
[tipoProblema,datEntrada,guardarCaso,nombreCaso]=entradaDatos.modoE_consola();
if isnan(tipoProblema), return, end %Se ha elegido salir del programa

%Resolución del problema
[valEntrada,valSalida]=main_logic(tipoProblema,datEntrada);


%Visualización del modelo
salidaDatos.visualizacion_modelo(tipoProblema,valEntrada,guardarCaso,nombreCaso)
    ;
```

```
28
29 % Presentación de resultados
30 salidaDatos.visualizar_resultados(tipoProblema,valEntrada,valSalida,guardarCaso,
       nombreCaso);
31 salidaDatos.generar_informeResultados(tipoProblema,datEntrada,valSalida,
       guardarCaso,nombreCaso);
```

### B.1.2. *main_modoArchivo.m*

```
1  %% Programa Análisis Plástico en Flexión Pura y Compuesta
2  %MODO 2: E/S de datos por archivo
3  %
4
5  clear all
6  close all
7
8  %Añadir directorios
9  pathname1=fileparts('presentation/');
10 addpath(genpath(pathname1))
11
12 pathname2=fileparts('logic/');
13 addpath(genpath(pathname2))
14
15
16 %Tomar datos de los archivos
17 fileC=fopen('Input_Files/input_casos.csv'); %archivo que recoge casos
18 fileP=fopen('Input_Files/input_problemas.csv'); %archivo que recoge
19                                     %la definición de los problemas
20 fileS=fopen('Input_Files/input_secciones.csv'); %archivo que recoge
21                                     %la definición de las secciones
22 fileM=fopen('Input_Files/input_materiales.csv');%archivo que recoge
23                                     %la definición de los materiales
24
25 %Almacenar los datos de los archivos
26 C=textscan(fileC,'%s %s %s %s','delimiter',',','headerlines',1);
27 P=textscan(fileP,'%s %f %s %f %f','delimiter',',','headerlines',1);
28 S=textscan(fileS,'%s %s %f %f %f %f %f %f', ...
29            'delimiter',',','headerlines',1);
30 M=textscan(fileM,'%s %s %f %f %f %f %f %f', ...
31            'delimiter',',','headerlines',1);
32
33
34 fclose(fileC);
35 fclose(fileP);
36 fclose(fileS);
```

```
37 fclose(fileM);
38
39 nCasos=size(C{1},1); %número de casos
40
41 %Ejecución de cada caso
42 for i=1:nCasos
43
44   guardarCaso=1;
45
46   %Entrada de datos
47   [tipoProblema,nombreCaso,datEntrada]=entradaDatos.modoE_archivo(i,C,P,S,M);
48
49
50   %Resolución del problema
51   [valEntrada,valSalida]=main_logic(tipoProblema,datEntrada);
52
53
54   %Visualización del modelo
55   salidaDatos.visualizacion_modelo(tipoProblema,valEntrada,guardarCaso,
         nombreCaso);
56
57   %Presentación de resultados
58   salidaDatos.visualizar_resultados(tipoProblema,valEntrada,valSalida,
         guardarCaso,nombreCaso);
59   salidaDatos.generar_informeResultados(tipoProblema,datEntrada,valSalida,
         guardarCaso,nombreCaso);
60
61   clear valEntrada; clear valSalida; clear guardarCaso; clear nombreCaso;
62   close all
63 end
```

### B.1.3. *main_modoGUI.fig*

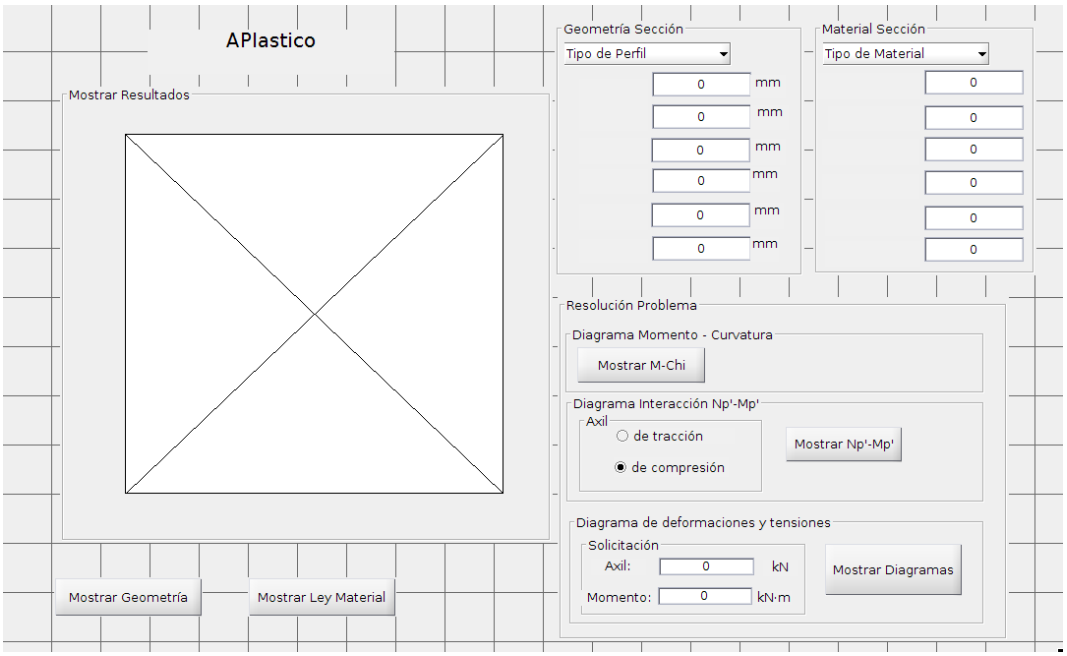



Figura B.1: *main_modoGUI.fig*

## Componentes de la Interfaz Gráfica de Usuario

- Panel (8)

  Mostrar Resultados (*show_uipanel*)

  Geometría Sección (*uipanel1*)

  Material Sección (*uipanel3*)

  Resolución Problema (*uipanel4*)

  Diagrama Momento - Curvatura (*uipanel5*)

  Diagrama interacción N'p-M'p (*uipanel6*)

  Diagrama de deformaciones y tensiones (*uipanel7*)

  Solicitación (*uipanel8*)

- Pop-up Menu (2)

  Tipo de Perfil (*tipoSecc_popupmenu*)

  Tipo de Material (*tipoMat_popupmenu*)

- Static Text (23)

  (6) *dim1_staticText*,..., *dim6_staticText* (indican los nombres de las dimensiones del tipo de sección seleccionado)

  6 static text que indican las unidades de las dimensiones de la sección (mm)

  (6) *valMat1*,...,*valMat6* (indican los nombres de las propiedades del tipo de modelo de material seleccionado)

(4) Axil, kN, Momento, kN*m (tipo y unidades de la solicitación)

(1) APlastico (nombre del programa)

- Edit Text (14)

  (6) *dim1_editText,..., dim6_editText* (campos en los que se introducen los valores numéricos de las dimensiones de la sección)

  (6) *valMat1_editText,..., valMat6_editText* (campos en los que se introducen los valores numéricos de las propiedades del material)

  (1) *valN_editText* (campo donde se introduce el valor del axil)

  (1) *valM_editText* (campo donde se introduce el valor del momento)

- Push-Button (5)

  *showGeo_pushbutton* (mostrar la geometría de la sección)

  *showMat_pushbutton* (mostrar el modelo del material)

  *showMC_pushbutton* (mostrar el diagrama Momento - Curvatura)

  *showNpMp_pushbutton* (mostrar el diagrama Interacción)

  *showSigEps_pushbutton* (mostrar el diagrama de deformaciones y tensiones)

- Button Group (1)

  *selectTipoAxil_buttongroup* (área donde se selecciona el caracter del axil del diagrama de interacción)

- Radio Button (2)

  *axilTracc_radiobutton* (seleccionar axil de tracción)

  *axilCompr_radiobutton* (seleccionar axil de compresión)

### B.1.4. *main_modoGUI.m*

```
1  function varargout = main_modoGUI(varargin)
2  %MAIN_MODOGUI MATLAB code for main_modoGUI.fig
3  %      MAIN_MODOGUI, by itself, creates a new MAIN_MODOGUI or raises the
       existing
4  %      singleton*.
5  %
6  %      H = MAIN_MODOGUI returns the handle to a new MAIN_MODOGUI or the handle
       to
7  %      the existing singleton*.
8  %
9  %      MAIN_MODOGUI('CALLBACK',hObject,eventData,handles,...) calls the local
10 %      function named CALLBACK in MAIN_MODOGUI.M with the given input arguments.
11 %
12 %      MAIN_MODOGUI('Property','Value',...) creates a new MAIN_MODOGUI or raises
       the
13 %      existing singleton*.  Starting from the left, property value pairs are
14 %      applied to the GUI before main_modoGUI_OpeningFcn gets called.  An
```

```
15 %      unrecognized property name or invalid value makes property application
16 %      stop.  All inputs are passed to main_modoGUI_OpeningFcn via varargin.
17 %
18 %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
19 %      instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help main_modoGUI
24
25 % Last Modified by GUIDE v2.5 08-Apr-2018 14:09:47
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                    'gui_Singleton',  gui_Singleton, ...
31                    'gui_OpeningFcn', @main_modoGUI_OpeningFcn, ...
32                    'gui_OutputFcn',  @main_modoGUI_OutputFcn, ...
33                    'gui_LayoutFcn',  [] , ...
34                    'gui_Callback',   []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46 %Añadir todos los directorios que requiere el programa
47 % Se asume que se parte de la carpeta donde se encuentra
48 % este script(APlastico)
49
50 pathname1=fileparts('presentation/');
51 addpath(genpath(pathname1))
52
53 pathname2=fileparts('logic/');
54 addpath(genpath(pathname2))
55
56
57 %--- Executes just before main_modoGUI is made visible.
58 function main_modoGUI_OpeningFcn(hObject, eventdata, handles, varargin)
59 % This function has no output args, see OutputFcn.
```

```
60 %hObject    handle to figure
61 %eventdata  reserved - to be defined in a future version of MATLAB
62 %handles    structure with handles and user data (see GUIDATA)
63 %varargin   command line arguments to main_modoGUI (see VARARGIN)
64
65 %Choose default command line output for main_modoGUI
66 handles.output = hObject;
67
68 %Update handles structure
69 guidata(hObject, handles);
70
71 %UIWAIT makes main_modoGUI wait for user response (see UIRESUME)
72 %uiwait(handles.figure1);
73
74 set(handles.selectTipoAxil_buttongroup,'SelectionChangeFcn', ...
75     @selectTipoAxil_buttongroup_SelectionChangeFcn);
76
77 set(hObject,'toolbar','figure');
78
79 %--- Outputs from this function are returned to the command line.
80 function varargout = main_modoGUI_OutputFcn(hObject, eventdata, handles)
81 %varargout  cell array for returning output args (see VARARGOUT);
82 %hObject    handle to figure
83 %eventdata  reserved - to be defined in a future version of MATLAB
84 %handles    structure with handles and user data (see GUIDATA)
85
86 %Get default command line output from handles structure
87 varargout{1} = handles.output;
88
89
90
91 function valMat6_editText_Callback(hObject, eventdata, handles)
92 %hObject    handle to valMat6_editText (see GCBO)
93 %eventdata  reserved - to be defined in a future version of MATLAB
94 %handles    structure with handles and user data (see GUIDATA)
95
96 %Hints: get(hObject,'String') returns contents of valMat6_editText as text
97 %        str2double(get(hObject,'String')) returns contents of valMat6_editText
    as a double
98
99 %Almacenar el contenido de valMat6_editText como un String. Si el String
100 %no es un número entonces input será empty
101 input=str2num(get(hObject,'String'));
102
103 %chaquear para ver si input es empty, establecer como valor de base, 0
```

```
104 if (isempty(input))
105   set(hObject,'String','0')
106 end
107
108
109 %--- Executes during object creation, after setting all properties.
110 function valMat6_editText_CreateFcn(hObject, eventdata, handles)
111 % hObject    handle to valMat6_editText (see GCBO)
112 % eventdata  reserved - to be defined in a future version of MATLAB
113 % handles    empty - handles not created until after all CreateFcns called
114
115 % Hint: edit controls usually have a white background on Windows.
116 %       See ISPC and COMPUTER.
117 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
118     set(hObject,'BackgroundColor','white');
119 end
120
121
122
123 function valMat5_editText_Callback(hObject, eventdata, handles)
124 % hObject    handle to valMat5_editText (see GCBO)
125 % eventdata  reserved - to be defined in a future version of MATLAB
126 % handles    structure with handles and user data (see GUIDATA)
127
128 % Hints: get(hObject,'String') returns contents of valMat5_editText as text
129 %        str2double(get(hObject,'String')) returns contents of valMat5_editText
    as a double
130
131 % Almacenar el contenido de valMat5_editText como un String. Si el String
132 % no es un número entonces input será empty
133 input=str2num(get(hObject,'String'));
134
135 % chaquear para ver si input es empty, establecer como valor de base, 0
136 if (isempty(input))
137   set(hObject,'String','0')
138 end
139
140
141 %--- Executes during object creation, after setting all properties.
142 function valMat5_editText_CreateFcn(hObject, eventdata, handles)
143 % hObject    handle to valMat5_editText (see GCBO)
144 % eventdata  reserved - to be defined in a future version of MATLAB
145 % handles    empty - handles not created until after all CreateFcns called
146
```

```
147 % Hint: edit controls usually have a white background on Windows.
148 %       See ISPC and COMPUTER.
149 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
150     set(hObject,'BackgroundColor','white');
151 end
152
153
154
155 function valMat4_editText_Callback(hObject, eventdata, handles)
156 % hObject    handle to valMat4_editText (see GCBO)
157 % eventdata  reserved - to be defined in a future version of MATLAB
158 % handles    structure with handles and user data (see GUIDATA)
159
160 % Hints: get(hObject,'String') returns contents of valMat4_editText as text
161 %        str2double(get(hObject,'String')) returns contents of valMat4_editText
    as a double
162
163 % Almacenar el contenido de valMat4_editText como un String. Si el String
164 % no es un número entonces input será empty
165 input=str2num(get(hObject,'String'));
166
167 % chaquear para ver si input es empty, establecer como valor de base, 0
168 if (isempty(input))
169   set(hObject,'String','0')
170 end
171
172
173 %--- Executes during object creation, after setting all properties.
174 function valMat4_editText_CreateFcn(hObject, eventdata, handles)
175 % hObject    handle to valMat4_editText (see GCBO)
176 % eventdata  reserved - to be defined in a future version of MATLAB
177 % handles    empty - handles not created until after all CreateFcns called
178
179 % Hint: edit controls usually have a white background on Windows.
180 %       See ISPC and COMPUTER.
181 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
182     set(hObject,'BackgroundColor','white');
183 end
184
185
186
187 function valMat3_editText_Callback(hObject, eventdata, handles)
188 % hObject    handle to valMat3_editText (see GCBO)
```

```
189 %eventdata  reserved - to be defined in a future version of MATLAB
190 %handles    structure with handles and user data (see GUIDATA)
191
192 %Hints: get(hObject,'String') returns contents of valMat3_editText as text
193 %        str2double(get(hObject,'String')) returns contents of valMat3_editText
        as a double
194
195 %Almacenar el contenido de valMat3_editText como un String. Si el String
196 %no es un número entonces input será empty
197 input=str2num(get(hObject,'String'));
198
199 %chaquear para ver si input es empty, establecer como valor de base, 0
200 if (isempty(input))
201   set(hObject,'String','0')
202 end
203
204
205
206 %--- Executes during object creation, after setting all properties.
207 function valMat3_editText_CreateFcn(hObject, eventdata, handles)
208 %hObject    handle to valMat3_editText (see GCBO)
209 %eventdata  reserved - to be defined in a future version of MATLAB
210 %handles    empty - handles not created until after all CreateFcns called
211
212 %Hint: edit controls usually have a white background on Windows.
213 %      See ISPC and COMPUTER.
214 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
215     set(hObject,'BackgroundColor','white');
216 end
217
218
219
220 function valMat2_editText_Callback(hObject, eventdata, handles)
221 %hObject    handle to valMat2_editText (see GCBO)
222 %eventdata  reserved - to be defined in a future version of MATLAB
223 %handles    structure with handles and user data (see GUIDATA)
224
225 %Hints: get(hObject,'String') returns contents of valMat2_editText as text
226 %        str2double(get(hObject,'String')) returns contents of valMat2_editText
        as a double
227
228 %Almacenar el contenido de valMat2_editText como un String. Si el String
229 %no es un número entonces input será empty
230 input=str2num(get(hObject,'String'));
```

```
231
232 %chaquear para ver si input es empty, establecer como valor de base, 0
233 if (isempty(input))
234   set(hObject,'String','0')
235 end
236
237
238 %--- Executes during object creation, after setting all properties.
239 function valMat2_editText_CreateFcn(hObject, eventdata, handles)
240 %hObject    handle to valMat2_editText (see GCBO)
241 %eventdata  reserved - to be defined in a future version of MATLAB
242 %handles    empty - handles not created until after all CreateFcns called
243
244 %Hint: edit controls usually have a white background on Windows.
245 %       See ISPC and COMPUTER.
246 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
247     set(hObject,'BackgroundColor','white');
248 end
249
250
251
252 function valMat1_editText_Callback(hObject, eventdata, handles)
253 %hObject    handle to valMat1_editText (see GCBO)
254 %eventdata  reserved - to be defined in a future version of MATLAB
255 %handles    structure with handles and user data (see GUIDATA)
256
257 %Hints: get(hObject,'String') returns contents of valMat1_editText as text
258 %       str2double(get(hObject,'String')) returns contents of valMat1_editText
    as a double
259
260 %Almacenar el contenido de valMat1_editText como un String. Si el String
261 %no es un número entonces input será empty
262 input=str2num(get(hObject,'String'));
263
264 %chaquear para ver si input es empty, establecer como valor de base, 0
265 if (isempty(input))
266   set(hObject,'String','0')
267 end
268
269
270 %--- Executes during object creation, after setting all properties.
271 function valMat1_editText_CreateFcn(hObject, eventdata, handles)
272 %hObject    handle to valMat1_editText (see GCBO)
273 %eventdata  reserved - to be defined in a future version of MATLAB
```

```
274 %handles      empty - handles not created until after all CreateFcns called
275
276 %Hint: edit controls usually have a white background on Windows.
277 %       See ISPC and COMPUTER.
278 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
279     set(hObject,'BackgroundColor','white');
280 end
281
282
283 %--- Executes on selection change in tipoMat_popupmenu.
284 function tipoMat_popupmenu_Callback(hObject, eventdata, handles)
285 %hObject    handle to tipoMat_popupmenu (see GCBO)
286 %eventdata  reserved - to be defined in a future version of MATLAB
287 %handles    structure with handles and user data (see GUIDATA)
288
289 %Hints: contents = cellstr(get(hObject,'String')) returns tipoMat_popupmenu
    contents as cell array
290 %        contents{get(hObject,'Value')} returns selected item from
    tipoMat_popupmenu
291
292 switch get(handles.tipoMat_popupmenu,'Value')
293   case 1
294     M.tipoMat=' ';
295   case 2
296     M.tipoMat='G1';
297   case 3
298     M.tipoMat='G2';
299   case 4
300     M.tipoMat='G3';
301   otherwise
302 end
303
304 set(handles.valMat1_editText,'String','0')
305 set(handles.valMat2_editText,'String','0')
306 set(handles.valMat3_editText,'String','0')
307 set(handles.valMat4_editText,'String','0')
308 set(handles.valMat5_editText,'String','0')
309 set(handles.valMat6_editText,'String','0')
310
311 set(handles.valN_editText,'String','0')
312 set(handles.valM_editText,'String','0')
313
314 panhandle=handles.show_uipanel;
315 paxGeo=subplot(1,1,1,'Parent',panhandle);
```

```
316 cla reset
317
318 if M.tipoMat==' '
319   valMatNames=[];
320 else
321   valMatNames=salidaDatos.valNamesMaterial(M.tipoMat);
322 end
323
324 if length(valMatNames)>=1
325   set(handles.valMat1_staticText,'String',[valMatNames{1},':'])
326 else
327   set(handles.valMat1_staticText,'String','')
328 end
329 if length(valMatNames)>=2
330   set(handles.valMat2_staticText,'String',[valMatNames{2},':'])
331 else
332   set(handles.valMat2_staticText,'String','')
333 end
334 if length(valMatNames)>=3
335   set(handles.valMat3_staticText,'String',[valMatNames{3},':'])
336 else
337   set(handles.valMat3_staticText,'String','')
338 end
339
340 if length(valMatNames)>=4
341   set(handles.valMat4_staticText,'String',[valMatNames{4},':'])
342 else
343   set(handles.valMat4_staticText,'String','')
344 end
345
346 if length(valMatNames)>=5
347   set(handles.valMat5_staticText,'String',[valMatNames{5},':'])
348 else
349   set(handles.valMat5_staticText,'String','')
350 end
351
352 if length(valMatNames)>=6
353   set(handles.valMat6_staticText,'String',[valMatNames{6},':'])
354 else
355   set(handles.valMat6_staticText,'String','')
356 end
357 set(handles.tipoMat_popupmenu,'UserData',M);
358
359
360 %--- Executes during object creation, after setting all properties.
```

```
361 function tipoMat_popupmenu_CreateFcn(hObject, eventdata, handles)
362 % hObject    handle to tipoMat_popupmenu (see GCBO)
363 % eventdata  reserved - to be defined in a future version of MATLAB
364 % handles    empty - handles not created until after all CreateFcns called
365
366 % Hint: popupmenu controls usually have a white background on Windows.
367 %       See ISPC and COMPUTER.
368 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
369     set(hObject,'BackgroundColor','white');
370 end
371
372
373 %--- Executes on selection change in tipoSecc_popupmenu.
374 function tipoSecc_popupmenu_Callback(hObject, eventdata, handles)
375 % hObject    handle to tipoSecc_popupmenu (see GCBO)
376 % eventdata  reserved - to be defined in a future version of MATLAB
377 % handles    structure with handles and user data (see GUIDATA)
378
379 % Hints: contents = cellstr(get(hObject,'String')) returns tipoSecc_popupmenu
    contents as cell array
380 %        contents{get(hObject,'Value')} returns selected item from
    tipoSecc_popupmenu
381
382 switch get(handles.tipoSecc_popupmenu,'Value')
383   case 1
384     S.tipoSecc=' ';
385   case 2
386     S.tipoSecc='R';
387   case 3
388     S.tipoSecc='C';
389   case 4
390     S.tipoSecc='I';
391   case 5
392     S.tipoSecc='H';
393   case 6
394     S.tipoSecc='T';
395   case 7
396     S.tipoSecc='Inb';
397   case 8
398     S.tipoSecc='Rh';
399   case 9
400     S.tipoSecc='Ch';
401   case 10
402     S.tipoSecc='IPN';
```

```
403     case 11
404       S.tipoSecc='IPE';
405     case 12
406       S.tipoSecc='Rombo';
407     case 13
408       S.tipoSecc='cajon';
409     otherwise
410   end
411
412   set(handles.dim1_editText,'String','0')
413   set(handles.dim2_editText,'String','0')
414   set(handles.dim3_editText,'String','0')
415   set(handles.dim4_editText,'String','0')
416   set(handles.dim5_editText,'String','0')
417   set(handles.dim6_editText,'String','0')
418
419   set(handles.valN_editText,'String','0')
420   set(handles.valM_editText,'String','0')
421
422   panhandle=handles.show_uipanel;
423   paxGeo=subplot(1,1,1,'Parent',panhandle);
424   cla reset
425
426   if S.tipoSecc==' '
427     dimNames=[];
428   else
429     dimNames=salidaDatos.dimSeccNames(S.tipoSecc);
430   end
431
432   if length(dimNames)>=1
433     set(handles.dim1_staticText,'String',[dimNames{1},':'])
434   else
435     set(handles.dim1_staticText,'String','')
436   end
437   if length(dimNames)>=2
438     set(handles.dim2_staticText,'String',[dimNames{2},':'])
439   else
440     set(handles.dim2_staticText,'String','')
441   end
442   if length(dimNames)>=3
443     set(handles.dim3_staticText,'String',[dimNames{3},':'])
444   else
445     set(handles.dim3_staticText,'String','')
446   end
447
```

```
448 if length(dimNames)>=4
449   set(handles.dim4_staticText,'String',[dimNames{4},':'])
450 else
451   set(handles.dim4_staticText,'String','')
452 end
453
454 if length(dimNames)>=5
455   set(handles.dim5_staticText,'String',[dimNames{5},':'])
456 else
457   set(handles.dim5_staticText,'String','')
458 end
459
460 if length(dimNames)>=6
461   set(handles.dim6_staticText,'String',[dimNames{6},':'])
462 else
463   set(handles.dim6_staticText,'String','')
464 end
465
466 set(handles.tipoSecc_popupmenu,'UserData',S);
467
468 %--- Executes during object creation, after setting all properties.
469 function tipoSecc_popupmenu_CreateFcn(hObject, eventdata, handles)
470 % hObject    handle to tipoSecc_popupmenu (see GCBO)
471 % eventdata  reserved - to be defined in a future version of MATLAB
472 % handles    empty - handles not created until after all CreateFcns called
473
474 % Hint: popupmenu controls usually have a white background on Windows.
475 %       See ISPC and COMPUTER.
476 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
477     set(hObject,'BackgroundColor','white');
478 end
479
480
481
482 function dim1_editText_Callback(hObject, eventdata, handles)
483 % hObject    handle to dim1_editText (see GCBO)
484 % eventdata  reserved - to be defined in a future version of MATLAB
485 % handles    structure with handles and user data (see GUIDATA)
486
487 % Hints: get(hObject,'String') returns contents of dim1_editText as text
488 %        str2double(get(hObject,'String')) returns contents of dim1_editText as
    a double
489
490 % Almacenar el contenido de dim1_editText como un String. Si el String
```

```
491 %no es un número entonces input será empty
492 input=str2num(get(hObject,'String'));
493
494 %chaquear para ver si input es empty, establecer como valor de base, 0
495 if (isempty(input))
496   set(hObject,'String','0')
497 end
498
499
500 %--- Executes during object creation, after setting all properties.
501 function dim1_editText_CreateFcn(hObject, eventdata, handles)
502 %hObject    handle to dim1_editText (see GCBO)
503 %eventdata  reserved - to be defined in a future version of MATLAB
504 %handles    empty - handles not created until after all CreateFcns called
505
506 %Hint: edit controls usually have a white background on Windows.
507 %       See ISPC and COMPUTER.
508 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
509     set(hObject,'BackgroundColor','white');
510 end
511
512
513
514 function dim2_editText_Callback(hObject, eventdata, handles)
515 %hObject    handle to dim2_editText (see GCBO)
516 %eventdata  reserved - to be defined in a future version of MATLAB
517 %handles    structure with handles and user data (see GUIDATA)
518
519 %Hints: get(hObject,'String') returns contents of dim2_editText as text
520 %       str2double(get(hObject,'String')) returns contents of dim2_editText as
    a double
521
522 %Almacenar el contenido de dim2_editText como un String. Si el String
523 %no es un número entonces input será empty
524 input=str2num(get(hObject,'String'));
525
526 %chaquear para ver si input es empty, establecer como valor de base, 0
527 if (isempty(input))
528   set(hObject,'String','0')
529 end
530
531
532 %--- Executes during object creation, after setting all properties.
533 function dim2_editText_CreateFcn(hObject, eventdata, handles)
```

```
534 %hObject    handle to dim2_editText (see GCBO)
535 %eventdata  reserved - to be defined in a future version of MATLAB
536 %handles    empty - handles not created until after all CreateFcns called
537
538 %Hint: edit controls usually have a white background on Windows.
539 %       See ISPC and COMPUTER.
540 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
541     set(hObject,'BackgroundColor','white');
542 end
543
544
545
546 function dim3_editText_Callback(hObject, eventdata, handles)
547 %hObject    handle to dim3_editText (see GCBO)
548 %eventdata  reserved - to be defined in a future version of MATLAB
549 %handles    structure with handles and user data (see GUIDATA)
550
551 %Hints: get(hObject,'String') returns contents of dim3_editText as text
552 %       str2double(get(hObject,'String')) returns contents of dim3_editText as
    a double
553
554 %Almacenar el contenido de dim3_editText como un String. Si el String
555 %no es un número entonces input será empty
556 input=str2num(get(hObject,'String'));
557
558 %chequear para ver si input es empty, establecer como valor de base, 0
559 if (isempty(input))
560   set(hObject,'String','0')
561 end
562
563
564
565 %--- Executes during object creation, after setting all properties.
566 function dim3_editText_CreateFcn(hObject, eventdata, handles)
567 %hObject    handle to dim3_editText (see GCBO)
568 %eventdata  reserved - to be defined in a future version of MATLAB
569 %handles    empty - handles not created until after all CreateFcns called
570
571 %Hint: edit controls usually have a white background on Windows.
572 %       See ISPC and COMPUTER.
573 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
574     set(hObject,'BackgroundColor','white');
575 end
```

```
576
577
578
579 function dim4_editText_Callback(hObject, eventdata, handles)
580 % hObject    handle to dim4_editText (see GCBO)
581 % eventdata  reserved - to be defined in a future version of MATLAB
582 % handles    structure with handles and user data (see GUIDATA)
583
584 % Hints: get(hObject,'String') returns contents of dim4_editText as text
585 %        str2double(get(hObject,'String')) returns contents of dim4_editText as
    a double
586
587 % Almacenar el contenido de dim4_editText como un String. Si el String
588 % no es un número entonces input será empty
589 input=str2num(get(hObject,'String'));
590
591 % chequear para ver si input es empty, establecer como valor de base, 0
592 if (isempty(input))
593   set(hObject,'String','0')
594 end
595
596
597
598 %--- Executes during object creation, after setting all properties.
599 function dim4_editText_CreateFcn(hObject, eventdata, handles)
600 % hObject    handle to dim4_editText (see GCBO)
601 % eventdata  reserved - to be defined in a future version of MATLAB
602 % handles    empty - handles not created until after all CreateFcns called
603
604 % Hint: edit controls usually have a white background on Windows.
605 %       See ISPC and COMPUTER.
606 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
607     set(hObject,'BackgroundColor','white');
608 end
609
610
611
612 function dim5_editText_Callback(hObject, eventdata, handles)
613 % hObject    handle to dim5_editText (see GCBO)
614 % eventdata  reserved - to be defined in a future version of MATLAB
615 % handles    structure with handles and user data (see GUIDATA)
616
617 % Hints: get(hObject,'String') returns contents of dim5_editText as text
618 %        str2double(get(hObject,'String')) returns contents of dim5_editText as
```

```
    a double
619
620 % Almacenar el contenido de dim5_editText como un String. Si el String
621 % no es un número entonces input será empty
622 input=str2num(get(hObject,'String'));
623
624 % chaquear para ver si input es empty, establecer como valor de base, 0
625 if (isempty(input))
626   set(hObject,'String','0')
627 end
628
629
630
631 %--- Executes during object creation, after setting all properties.
632 function dim5_editText_CreateFcn(hObject, eventdata, handles)
633 % hObject    handle to dim5_editText (see GCBO)
634 % eventdata  reserved - to be defined in a future version of MATLAB
635 % handles    empty - handles not created until after all CreateFcns called
636
637 % Hint: edit controls usually have a white background on Windows.
638 %       See ISPC and COMPUTER.
639 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
640     set(hObject,'BackgroundColor','white');
641 end
642
643
644
645 function dim6_editText_Callback(hObject, eventdata, handles)
646 % hObject    handle to dim6_editText (see GCBO)
647 % eventdata  reserved - to be defined in a future version of MATLAB
648 % handles    structure with handles and user data (see GUIDATA)
649
650 % Hints: get(hObject,'String') returns contents of dim6_editText as text
651 %        str2double(get(hObject,'String')) returns contents of dim6_editText as
    a double
652
653 % Almacenar el contenido de dim6_editText como un String. Si el String
654 % no es un número entonces input será empty
655 input=str2num(get(hObject,'String'));
656
657 % chaquear para ver si input es empty, establecer como valor de base, 0
658 if (isempty(input))
659   set(hObject,'String','0')
660 end
```

```
661
662
663
664 %--- Executes during object creation, after setting all properties.
665 function dim6_editText_CreateFcn(hObject, eventdata, handles)
666 % hObject    handle to dim6_editText (see GCBO)
667 % eventdata  reserved - to be defined in a future version of MATLAB
668 % handles    empty - handles not created until after all CreateFcns called
669
670 % Hint: edit controls usually have a white background on Windows.
671 %       See ISPC and COMPUTER.
672 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
673     set(hObject,'BackgroundColor','white');
674 end
675
676
677 %--- Executes on button press in showGeo_pushbutton.
678 function showGeo_pushbutton_Callback(hObject, eventdata, handles)
679 % hObject    handle to showGeo_pushbutton (see GCBO)
680 % eventdata  reserved - to be defined in a future version of MATLAB
681 % handles    structure with handles and user data (see GUIDATA)
682
683 S=get(handles.tipoSecc_popupmenu,'UserData');
684
685 dim(1)=str2num(get(handles.dim1_editText,'String'));
686 dim(2)=str2num(get(handles.dim2_editText,'String'));
687 dim(3)=str2num(get(handles.dim3_editText,'String'));
688 dim(4)=str2num(get(handles.dim4_editText,'String'));
689 dim(5)=str2num(get(handles.dim5_editText,'String'));
690 dim(6)=str2num(get(handles.dim6_editText,'String'));
691
692
693 if dim(1)~=0
694 p=generacion_Modelo.verticesPoly(S.tipoSecc,dim);
695 yG=propSeccion.propGeom(p,'yG');
696 zG=propSeccion.propGeom(p,'zG');
697 p=problGeom.cambioSistCoord(p,-yG,-zG,0);
698
699 panhandle=handles.show_uipanel;
700 paxGeo=subplot(1,1,1,'Parent',panhandle);
701 cla reset
702 salidaDatos.mostrar_perfilConEjes(p)
703
704 handles.panhandle=panhandle;
```

```
705 handles.paxGeo=paxGeo;
706 end
707
708 guidata(hObject,handles);
709
710
711 %--- Executes on button press in showMat_pushbutton.
712 function showMat_pushbutton_Callback(hObject, eventdata, handles)
713 % hObject    handle to showMat_pushbutton (see GCBO)
714 % eventdata  reserved - to be defined in a future version of MATLAB
715 % handles    structure with handles and user data (see GUIDATA)
716
717 M=get(handles.tipoMat_popupmenu,'UserData');
718 valores(1)=str2num(get(handles.valMat1_editText,'String'));
719 valores(2)=str2num(get(handles.valMat2_editText,'String'));
720 valores(3)=str2num(get(handles.valMat3_editText,'String'));
721 valores(4)=str2num(get(handles.valMat4_editText,'String'));
722 valores(5)=str2num(get(handles.valMat5_editText,'String'));
723 valores(6)=str2num(get(handles.valMat6_editText,'String'));
724
725 if valores(1)~=0
726 ley_mat=generacion_Modelo.comportMaterial(M.tipoMat,valores);
727
728 panhandle=handles.show_uipanel;
729 paxMat=subplot(1,1,1,'Parent',panhandle);
730 cla reset
731 salidaDatos.mostrar_ley_mat(ley_mat)
732
733 handles.panhandle=panhandle;
734 hadles.paxMat=paxMat;
735 end
736
737 guidata(hObject,handles);
738
739 %--- Executes on button press in showSigEps_pushbutton.
740 function showSigEps_pushbutton_Callback(hObject, eventdata, handles)
741 % hObject    handle to showSigEps_pushbutton (see GCBO)
742 % eventdata  reserved - to be defined in a future version of MATLAB
743 % handles    structure with handles and user data (see GUIDATA)
744
745 %tomar valores de la geometría
746 S=get(handles.tipoSecc_popupmenu,'UserData');
747
748 dim(1)=str2num(get(handles.dim1_editText,'String'));
749 dim(2)=str2num(get(handles.dim2_editText,'String'));
```

```
750 dim(3)=str2num(get(handles.dim3_editText,'String'));
751 dim(4)=str2num(get(handles.dim4_editText,'String'));
752 dim(5)=str2num(get(handles.dim5_editText,'String'));
753 dim(6)=str2num(get(handles.dim6_editText,'String'));
754
755 %tomar valores del material
756 M=get(handles.tipoMat_popupmenu,'UserData');
757
758 valores(1)=str2num(get(handles.valMat1_editText,'String'));
759 valores(2)=str2num(get(handles.valMat2_editText,'String'));
760 valores(3)=str2num(get(handles.valMat3_editText,'String'));
761 valores(4)=str2num(get(handles.valMat4_editText,'String'));
762 valores(5)=str2num(get(handles.valMat5_editText,'String'));
763 valores(6)=str2num(get(handles.valMat6_editText,'String'));
764
765 Axil=str2num(get(handles.valN_editText,'String'));
766 Momento=str2num(get(handles.valM_editText,'String'));
767
768 tipoProblema=3;
769
770 if valores(1)~=0 && dim(1)~=0
771 datEntrada={S.tipoSecc,dim,M.tipoMat,valores,Axil,Momento};
772 [valEntrada,valSalida]=main_logic(tipoProblema,datEntrada);
773
774 guardarCaso=0;
775 nombreCaso='sinNombre';
776
777 p=valEntrada{1};
778
779 z_=valSalida{1};
780 sigz_=valSalida{2};
781 epsz_=valSalida{3};
782 Ap1=valSalida{5};
783 Ap2=valSalida{6};
784
785 panhandle=handles.show_uipanel;
786
787 pax1=subplot(2,2,1,'Parent',panhandle);
788 cla reset
789 salidaDatos.mostrar_solicitacion(p,Axil,Momento)
790
791 pax2=subplot(2,2,2,'Parent',panhandle);
792 cla reset
793 salidaDatos.mostrar_areasPlastificadas(p,z_,Ap1,Ap2);
794
```

```
795 pax3=subplot(2,2,3,'Parent',panhandle);
796 cla reset
797 ancho=max(p(:,1))-min(p(:,1));
798 alto=max(p(:,2))-min(p(:,2));
799
800 salidaDatos.mostrar_ley_deformaciones([epsz_,z_])
801 if alto/ancho<=1.05, pbaspect([ancho, alto, 1]); end
802
803 pax4=subplot(2,2,4,'Parent',panhandle);
804 cla reset
805 salidaDatos.mostrar_ley_tensiones([sigz_,z_])
806 if alto/ancho<=1.05, pbaspect([ancho, alto, 1]); end
807
808
809 handles.panhandle=panhandle;
810 handles.pax1=pax1;
811 handles.pax2=pax2;
812 handles.pax3=pax3;
813 handles.pax4=pax4;
814 end
815
816 guidata(hObject,handles);
817
818
819
820 %--- Executes on button press in showNpMp_pushbutton.
821 function showNpMp_pushbutton_Callback(hObject, eventdata, handles)
822 % hObject    handle to showNpMp_pushbutton (see GCBO)
823 % eventdata  reserved - to be defined in a future version of MATLAB
824 % handles    structure with handles and user data (see GUIDATA)
825
826 %tomar valores de la geometría
827 S=get(handles.tipoSecc_popupmenu,'UserData');
828
829 dim(1)=str2num(get(handles.dim1_editText,'String'));
830 dim(2)=str2num(get(handles.dim2_editText,'String'));
831 dim(3)=str2num(get(handles.dim3_editText,'String'));
832 dim(4)=str2num(get(handles.dim4_editText,'String'));
833 dim(5)=str2num(get(handles.dim5_editText,'String'));
834 dim(6)=str2num(get(handles.dim6_editText,'String'));
835
836 %tomar valores del material
837 M=get(handles.tipoMat_popupmenu,'UserData');
838
839 valores(1)=str2num(get(handles.valMat1_editText,'String'));
```

```
840 valores(2)=str2num(get(handles.valMat2_editText,'String'));
841 valores(3)=str2num(get(handles.valMat3_editText,'String'));
842 valores(4)=str2num(get(handles.valMat4_editText,'String'));
843 valores(5)=str2num(get(handles.valMat5_editText,'String'));
844 valores(6)=str2num(get(handles.valMat6_editText,'String'));
845
846 tipoProblema=2;
847
848 probl2=get(handles.selectTipoAxil_buttongroup,'UserData');
849 signoN=probl2.signoAxil;
850
851 if valores(1)~=0 && dim(1)~=0
852 datEntrada={S.tipoSecc,dim,M.tipoMat,valores,signoN};
853 [valEntrada,valSalida]=main_logic(tipoProblema,datEntrada);
854 guardarCaso=0;
855 nombreCaso='sinNombre';
856
857 p=valEntrada{1};
858 d_int=valSalida{1};
859
860 panhandle=handles.show_uipanel;
861 pax_dInt=subplot(1,1,1,'Parent',panhandle);
862 cla reset
863 salidaDatos.mostrar_diagr_interacc(p,d_int)
864
865 handles.panhandle=panhandle;
866 handles.pax_dInt=pax_dInt;
867 end
868
869 guidata(hObject,handles);
870
871
872
873 %--- Executes on button press in showMC_pushbutton.
874 function showMC_pushbutton_Callback(hObject, eventdata, handles)
875 % hObject    handle to showMC_pushbutton (see GCBO)
876 % eventdata  reserved - to be defined in a future version of MATLAB
877 % handles    structure with handles and user data (see GUIDATA)
878
879 %tomar valores de la geometría
880 S=get(handles.tipoSecc_popupmenu,'UserData');
881
882 dim(1)=str2num(get(handles.dim1_editText,'String'));
883 dim(2)=str2num(get(handles.dim2_editText,'String'));
884 dim(3)=str2num(get(handles.dim3_editText,'String'));
```

```
885 dim(4)=str2num(get(handles.dim4_editText,'String'));
886 dim(5)=str2num(get(handles.dim5_editText,'String'));
887 dim(6)=str2num(get(handles.dim6_editText,'String'));
888
889 %tomar valores del material
890 M=get(handles.tipoMat_popupmenu,'UserData');
891
892 valores(1)=str2num(get(handles.valMat1_editText,'String'));
893 valores(2)=str2num(get(handles.valMat2_editText,'String'));
894 valores(3)=str2num(get(handles.valMat3_editText,'String'));
895 valores(4)=str2num(get(handles.valMat4_editText,'String'));
896 valores(5)=str2num(get(handles.valMat5_editText,'String'));
897 valores(6)=str2num(get(handles.valMat6_editText,'String'));
898
899 tipoProblema=1;
900
901 if valores(1)~=0 && dim(1)~=0
902
903 datEntrada={S.tipoSecc,dim,M.tipoMat,valores};
904 [valEntrada,valSalida]=main_logic(tipoProblema,datEntrada);
905 guardarCaso=0;
906 nombreCaso='sinNombre';
907
908 Me=valSalida{1};
909 chie=valSalida{2};
910 Mp=valSalida{3};
911 d_MC=valSalida{7};
912
913 panhandle=handles.show_uipanel;
914 paxMC=subplot(1,1,1,'Parent',panhandle);
915 cla reset
916 salidaDatos.mostrar_diagr_MC(d_MC)
917 hold on
918 plot(chie,Me,'sg')
919 plot([min(d_MC(:,1)),max(d_MC(:,1))],[Mp,Mp],'--k')
920 legend('M - \chi','(M_e,\chi_e)','M_p')
921
922 handles.panhandle=panhandle;
923 handlex.paxMC=paxMC;
924 end
925
926 guidata(hObject,handles);
927
928
929
```

```
930 function valN_editText_Callback(hObject, eventdata, handles)
931 % hObject    handle to valN_editText (see GCBO)
932 % eventdata  reserved - to be defined in a future version of MATLAB
933 % handles    structure with handles and user data (see GUIDATA)
934
935 % Hints: get(hObject,'String') returns contents of valN_editText as text
936 %        str2double(get(hObject,'String')) returns contents of valN_editText as
    a double
937
938 % Almacenar el contenido de valN_editText como un String. Si el String
939 % no es un número entonces input será empty
940 input=str2num(get(hObject,'String'));
941
942 % chaquear para ver si input es empty, establecer como valor de base, 0
943 if (isempty(input))
944   set(hObject,'String','0')
945 end
946
947
948
949 %--- Executes during object creation, after setting all properties.
950 function valN_editText_CreateFcn(hObject, eventdata, handles)
951 % hObject    handle to valN_editText (see GCBO)
952 % eventdata  reserved - to be defined in a future version of MATLAB
953 % handles    empty - handles not created until after all CreateFcns called
954
955 % Hint: edit controls usually have a white background on Windows.
956 %       See ISPC and COMPUTER.
957 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
958     set(hObject,'BackgroundColor','white');
959 end
960
961
962
963 function valM_editText_Callback(hObject, eventdata, handles)
964 % hObject    handle to valM_editText (see GCBO)
965 % eventdata  reserved - to be defined in a future version of MATLAB
966 % handles    structure with handles and user data (see GUIDATA)
967
968 % Hints: get(hObject,'String') returns contents of valM_editText as text
969 %        str2double(get(hObject,'String')) returns contents of valM_editText as
    a double
970
971 % Almacenar el contenido de valM_editText como un String. Si el String
```

```
% no es un número entonces input será empty
input=str2num(get(hObject,'String'));

% chaquear para ver si input es empty, establecer como valor de base, 0
if (isempty(input))
  set(hObject,'String','0')
end



%--- Executes during object creation, after setting all properties.
function valM_editText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valM_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function selectTipoAxil_buttongroup_SelectionChangeFcn(hObject,eventdata,handles
    )
handles=guidata(hObject);


switch (get(eventdata.NewValue,'Tag'))
    case 'axilTracc_radiobutton'
      probl2.signoAxil='p';
    case 'axilCompr_radiobutton'
      probl2.signoAxil='n';
    otherwise
end

set(handles.selectTipoAxil_buttongroup,'UserData',probl2);
```

## B.2. *presentation/*

Módulo de preproceso y post-proceso.

### B.2.1. *entradaDatos*

```
classdef entradaDatos
methods(Static)

%*********************************************************************%
%                                                                     %
% entradaDatos: módulo que contiene las funciones para la recogida    %
%               y almacenamiento de datos                             %
%                                                                     %
% Funciones que contiene:                                             %
%   modoE_consola(): función con la que se recogen los datos en el    %
%                    modo consola                                     %
%   modoE_archivo(): función con la que se recogen los datos en el    %
%                    modo por archivo                                 %
%                                                                     %
%*********************************************************************%


function [tipoProblema,datEntrada,guardarCaso,nombreCaso]=modoE_consola()
%---------------------------------------------------------------------
% Pedir datos en el modo consola
%---------------------------------------------------------------------

  %PEDIR DATOS GEOMETRÍA
  fprintf('\nIntroducir tipo de sección:')
  fprintf('\nR: sección rectangular')
  fprintf('\nC: sección circular')
  fprintf('\nI: sección doble T')
  fprintf('\nH: sección H')
  fprintf('\nT: sección en T')
  fprintf('\nInb: sección doble T no bisimétrica')
  fprintf('\nRh: sección rectangular con hueco')
  fprintf('\nCh: sección circular con hueco')
  fprintf('\nRombo: sección rombo')
  fprintf('\nIPE: sección IPE')
  fprintf('\nIPN: sección IPN')
  fprintf('\n\n Introducir caracteres entre comillas simple\n')

  tipoSecc=input('Tipo de sección: ');
  if isnumeric(tipoSecc), tipoSecc=num2str(tipoSecc); end
  seccNames={'R','I','H','T','Inb','Rh','Ch','Rombo','IPE','IPN','cajon','0'};
  while ~ismember(tipoSecc,seccNames)
    fprintf(['\nTipo de sección no reconocido, por favor introduzca un valor'
        ,...
          '\nadecuado o pulse 0 para salir (caracteres entre comillas
             simples)\n'])
```

```
44      tipoSecc=input('Tipo de sección: ');
45      if isnumeric(tipoSecc), tipoSecc=num2str(tipoSecc);end
46    end
47
48    switch tipoSecc
49
50      case 'R'
51        fprintf('\nR: sección rectangular\n')
52        dim(1)=input('ancho, b (mm): ');
53        dim(2)=input('canto, h (mm): ');
54
55      case 'I'
56        fprintf('\nI: sección doble T\n')
57        dim(1)=input('altura total, h (mm): ');
58        dim(2)=input('ancho ala, bf (mm): ');
59        dim(3)=input('espesor ala, tf (mm): ');
60        dim(4)=input('espesor alma, tw (mm): ');
61
62      case 'C'
63        fprintf('\nC: sección circular\n')
64        dim(1)=input('radio, r(mm): ');
65
66      case 'T'
67        fprintf('\nT: sección en T\n')
68        dim(1)=input('altura total, h (mm): ');
69        dim(2)=input('ancho ala, bf (mm): ');
70        dim(3)=input('espesor ala, tf (mm): ');
71        dim(4)=input('espesor alma, tw (mm): ');
72
73      case 'Inb'
74        fprintf('\nInb: sección doble T no bisimétrica\n')
75        dim(1)=input('altura total sección, h (mm): ');
76        dim(2)=input('ancho ala superior, bf1 (mm): ');
77        dim(3)=input('espesor ala superior, tf1 (mm): ');
78        dim(4)=input('espesor alma, tw (mm): ');
79        dim(5)=input('ancho ala inferior, bf2 (mm): ');
80        dim(6)=input('espesor ala inferior, tf2 (mm): ');
81
82      case 'Ch'
83        fprintf('\nCh: sección circular con hueco\n')
84        dim(1)=input('radio, r (mm): ');
85        dim(2)=input('espesor, e (mm): ');
86
87      case 'Rh'
88        fprintf('\nRh: sección rectangular con hueco\n')
```

```
89          dim(1)=input('ancho, b (mm): ');
90          dim(2)=input('canto, h (mm): ');
91          dim(3)=input('espesor, e (mm): ');
92
93      case 'H'
94          fprintf('\nH: sección H\n')
95          dim(1)=input('ancho ala, hf(mm): ');
96          dim(2)=input('alura total, b (mm): ');
97          dim(3)=input('espesor ala, tf (mm): ');
98          dim(4)=input('espesor alma, tw (mm): ');
99
100     case 'Rombo'
101         fprintf('\nRombo: sección rombo\n')
102         dim(1)=input('diagonal horizontal, d1 (mm): ');
103         dim(2)=input('diagonal vertical, d2 (mm): ');
104
105     case 'IPN'
106         fprintf('\nIPN: sección IPN\n')
107         dim(1)=input('altura total perfil, h (mm): ');
108         dim(2)=input('ancho alas, b (mm): ');
109         dim(3)=input('radio entre ala y alma, r1 (mm): ');
110         dim(4)=input('espesor ala, tf (mm): ');
111         dim(5)=input('radio ala, r2 (mm): ');
112         dim(6)=input('altura alma, d (mm): ');
113         fprintf('\nNota: espesor alma (tw) cumple: tw=r1')
114
115     case 'IPE'
116         fprintf('\nIPE: sección IPE\n')
117         dim(1)=input('altura total perfil, h (mm): ');
118         dim(2)=input('ancho alas, bf (mm): ');
119         dim(3)=input('espesor alma, tw (mm): ');
120         dim(4)=input('espesor ala, tf (mm): ');
121         dim(5)=input('radio ala, r (mm): ');
122         dim(6)=input('altura alma, d (mm): ');
123
124
125     case 'cajon'
126         fprintf('\ncajon: sección en cajón\n')
127         dim(1)=input('distancia entre caras extremas de las almas, b (mm): ');
128         dim(2)=input('saliente alas, b1 (mm): ');
129         dim(3)=input('espesor ala superior, tf1 (mm): ');
130         dim(4)=input('altura almas, hw (mm): ');
131         dim(5)=input('espesor almas, tw (mm): ');
132         dim(6)=input('espesor ala inferior, tf2 (mm): ');
133
```

```
134     case '0'
135       fprintf('\n Ha elegido salir del programa\n')
136       tipoProblema=NaN; datEntrada=NaN; guardarCaso=NaN; nombreCaso=NaN;
137       return
138 
139     otherwise
140       error('Tipo de sección no reconocido')
141 
142   end
143 
144 
145   %PEDIR DATOS MATERIAL
146   fprintf('\nIntroducir tipo de comportamiento del material del modelo\n')
147   fprintf(['\nG1: Material perfectamente plástico con igual comportamiento', ...
148           ' a tracción y compresión'])
149   fprintf(['\nG2: Material perfectamente plástico con diferente', ...
150            'comportamiento a tracción y compresión'])
151   fprintf(['\nG3: Material con endurecimiento con igual comportamiento a', ...
152           'tracción y compresión\n'])
153 
154   tipoMat=input('Tipo de material:');
155   if isnumeric(tipoMat), tipoMat=num2str(tipoMat);end
156   matNames={'G1','G2','G3','0'};
157 
158   while ~ismember(tipoMat,matNames)
159     fprintf(['\nPor favor, introduza un valor de material', ...
160             ' \nadecuado G1, G2 o G3, o bien pulse 0 ', ...
161             'para salir\n'])
162     tipoMat=input('Tipo de material:');
163     if isnumeric(tipoMat), tipoMat=num2str(tipoMat);end
164   end
165 
166   switch tipoMat
167     case 'G1'
168 
169       valores(1)=input('Tensión en el límite elástico (MPa): ');
170       valores(2)=input('Módulo de Young (MPa): ');
171 
172     case 'G2'
173       valores(1)=input('Módulo de Young (MPa): ');
174       valores(2)=input('Tensión en el límite elástico a compresión (MPa): ');
175       valores(3)=input('Tensión en el límite elástico a tracción (MPa): ');
176 
177     case 'G3'
178       valores(1)=input('Deformación unitaria en el límite elástico: ');
```

```
179         valores(2)=input('Tensión de en el límite elástico (MPa): ');
180         valores(3)=input('Deformación unitaria en el inicio del endurecimiento: ')
                ;
181         valores(4)=input('Tensión en el inicio del endurecimiento (MPa): ');
182         valores(5)=input('Deformación unitaria en el final del endurecimiento: ');
183         valores(6)=input('Tensión máxima (MPa): ');
184
185      case '0'
186         fprintf('\n Ha elegido salir del programa\n')
187         tipoProblema=NaN; datEntrada=NaN; guardarCaso=NaN; nombreCaso=NaN;
188         return
189
190      otherwise
191         error('Tipo de material no reconocido.')
192        end
193
194    %ELEGIR TIPO DE PROBLEMA
195
196    fprintf('\n Escoger qué problema se desea realizar')
197    fprintf('\n Introducir:')
198    fprintf('\n  1: Cálculo de valores característicos del comportamiento')
199    fprintf('\n     de la sección en flexión pura, se calculará:')
200    fprintf('\n   * Diagrama Momento - Curvatura')
201    fprintf('\n   * Momento Elástico, Momento Plástico y factor de forma')
202    fprintf('\n  2: Cálculo del diagrama de interacción de la sección')
203    fprintf('\n     N p - M p')
204    fprintf('\n  3: Cálculo del diagrama de tensiones dada la')
205    fprintf('\n     solicitación de la sección (N,M)\n')
206
207    tipoProblema=input('Introducir número asociado al problema: ');
208    tipoProblNames={'1','2','3','0'};
209    if isnumeric(tipoProblema), tipoProblema=num2str(tipoProblema);end
210
211    while ~ismember(tipoProblema,tipoProblNames)
212      fprintf(['\nPor favor, introduza un tipo de problema', ...
213              ' \nadecuado 1, 2 o 3, o bien pulse 0 ', ...
214              'para salir\n'])
215      tipoProblema=input('Introducir número asociado al problema: ');
216      if isnumeric(tipoProblema), tipoProblema=num2str(tipoProblema);end
217    end
218
219    tipoProblema=str2num(tipoProblema);
220
221    switch tipoProblema
222
```

```
223   case 1
224   fprintf('\n  1: Cálculo de valores característicos del comportamiento')
225   fprintf('\n  de la sección en flexión pura')
226
227   datEntrada{1}=tipoSecc;
228   datEntrada{2}=dim;
229   datEntrada{3}=tipoMat;
230   datEntrada{4}=valores; % valores del material
231
232
233   case 2
234   fprintf('\n  2: Cálculo del diagrama de interacción de la sección')
235   fprintf('\n     N p - M p')
236   fprintf('\n Introduzca')
237   fprintf('\n   n: axil de compresión')
238   fprintf('\n   p: axil de tracción\n')
239   signoN=input('SignoN: ');
240   if isnumeric(signoN), signoN=num2str(signoN);end
241   valAdmisible={'n','p','0'};
242
243   while ~ismember(valAdmisible,signoN)
244     fprintf('\n Valor no admisible.')
245     fprintf('\nPor favor, introduzca n para axil de compresión o p para axil de
          tracción')
246     fprintf('\n o bien pulse 0, para salir\n')
247     signoN=input('SignoN: ');
248     if isnumeric(signoN), signoN=num2str(signoN);end
249   end
250
251
252   datEntrada{1}=tipoSecc;
253   datEntrada{2}=dim;
254   datEntrada{3}=tipoMat;
255   datEntrada{4}=valores;
256   datEntrada{5}=signoN;
257
258
259   case 3
260   fprintf('\n  3: Cálculo del diagrama de tensiones dada la')
261   fprintf('\n     solicitación de la sección (N,M)\n')
262
263   fprintf('\n Introducir valores de la solicitación: \n')
264   N=input(' Valor del axil, N (kN): ');
265   M=input(' Valor del momento, M (kN*m): ');
266
```

```
267     datEntrada{1}=tipoSecc;
268     datEntrada{2}=dim;
269     datEntrada{3}=tipoMat;
270     datEntrada{4}=valores;
271     datEntrada{5}=N;
272     datEntrada{6}=M;
273 
274     case 0
275       fprintf('\n Ha elegido salir del programa\n')
276       tipoProblema=NaN; datEntrada=NaN; guardarCaso=NaN; nombreCaso=NaN;
277       return
278 
279     otherwise
280       error('\n Tipo de problema no reconocido')
281 
282     end
283     fprintf('\n¿Desea guardar el caso?')
284     fprintf('\n Pulse 1 si sí y 0 en caso negativo\n')
285     guardarCaso=input('Guardar resultados:');
286 
287     if guardarCaso
288       fprintf('\nAsignar un nombre al caso(entre comillas simples:)\n')
289       nombreCaso=input('Nombre:');
290     else
291       nombreCaso='sinnombre';
292     end
293 end
294 %
295 %=========================================================================
296 
297 
298 function [tipoProblema,nombreCaso,datEntrada]=modoE_archivo(i,C,P,S,M)
299 %-------------------------------------------------------------------------
300 %Entrada de datos por medio de archivo con textscan (sin tablas)
301 %-------------------------------------------------------------------------
302 
303 %Entrada de datos por archivo
304 %C: cell que contiene los casos que se quiere simular
305 %P: cell que contiene los problemas y sus valores asociados
306 %S: cell que contiene las secciones
307 %M: cell que contiene los materiales
308 
309 
310   nombreCaso=C{1}(i);
311   nombreProblema=C{2}(i);
```

```
312   nombreSecc=C{3}(i);
313   nombreMaterial=C{4}(i);
314
315
316   nombreCaso=char(nombreCaso);
317   nombreProblema=char(nombreProblema);
318   nombreSecc=char(nombreSecc);
319   nombreMaterial=char(nombreMaterial);
320
321   %Determinar tipo de sección y sus dimensiones
322   for i=1:length(S{1})
323     ss=[char(S{1}(i)),' '];
324     S{1}(i)={ss};
325   end
326
327   nombreSecc=[nombreSecc,' '];
328   b=strfind(S{1},nombreSecc);
329   iSecc=find(~(cellfun('isempty',b)));
330   tipoSecc=S{2}(iSecc);
331   tipoSecc=char(tipoSecc);
332   dimSecc=[S{3}(iSecc),S{4}(iSecc),S{5}(iSecc),S{6}(iSecc),S{7}(iSecc),S{8}(
          iSecc)];
333   dimSecc=dimSecc(~isnan(dimSecc));
334
335
336   %Determinar tipo de material y sus propiedades
337   for i=1:length(M{1})
338     mm=[char(M{1}(i)),' '];
339     M{1}(i)={mm};
340   end
341
342   nombreMaterial=[nombreMaterial,' '];
343   b=strfind(M{1},nombreMaterial);
344   iMat=find(~(cellfun('isempty',b)));
345   tipoMat=M{2}(iMat);
346   tipoMat=char(tipoMat);
347   valMat=[M{3}(iMat),M{4}(iMat),M{5}(iMat),M{6}(iMat),M{7}(iMat),M{8}(iMat)];
348   valMat=valMat(find(~isnan(valMat)));
349
350
351   %Determinar tipo de problema
352
353   for i=1:length(P{1})
354     pp=[char(P{1}(i)),' '];
355     P{1}(i)={pp};
```

```
356  end
357
358  nombreProblema=[nombreProblema,' '];
359  b=strfind(P{1},nombreProblema);
360  iProblema=find(~(cellfun('isempty',b)));
361  tipoProblema=P{2}(iProblema);
362  tipoProblema=char(tipoProblema);
363
364  %GENERACIÓN DEL MODELO
365  switch tipoProblema
366
367  case 1
368
369    datEntrada{1}=tipoSecc;
370    datEntrada{2}=dimSecc;
371    datEntrada{3}=tipoMat;
372    datEntrada{4}=valMat;
373
374  case 2
375
376    signoN=P{3}(iProblema);
377
378    datEntrada{1}=tipoSecc;
379    datEntrada{2}=dimSecc;
380    datEntrada{3}=tipoMat;
381    datEntrada{4}=valMat;
382    datEntrada{5}=char(signoN);
383
384
385  case 3
386
387    N=P{4}(iProblema);
388    M_=P{5}(iProblema);
389
390    datEntrada{1}=tipoSecc;
391    datEntrada{2}=dimSecc;
392    datEntrada{3}=tipoMat;
393    datEntrada{4}=valMat;
394    datEntrada{5}=N;
395    datEntrada{6}=M_;
396
397  otherwise
398    error('\n Tipo de problema no reconocido')
399  end
400
```

```
401 end
402 %
403 %=========================================================================
404
405 end
406 end
```

### B.2.2. *salidaDatos*

```
1  classdef salidaDatos
2  methods(Static)
3
4  %*************************************************************************%
5  %                                                                         %
6  % salidaDatos: módulo que contiene las funciones para la muestra         %
7  %              de resultados                                              %
8  %                                                                         %
9  % Funciones que contiene:                                                %
10 %   visualizacion_modelo(): mostrar el modelo de entrada (modelo          %
11 %                           geométrico y del material)                    %
12 %   mostrar_ley_mat(): mostrar ley constitutiva del material             %
13 %   mostrar_perfilConEjes(): mostrar perfil con sus ejes de referencia   %
14 %   mostrar_solicitacion(): mostrar solicitación (N,M) de la sección     %
15 %   visualizar_resultados(): mostrar resultados del programa de          %
16 %                            forma gráfica                                %
17 %   mostrar_diagr_MC(): mostrar diagrama Momento-Curvatura               %
18 %   mostrar_diagr_interacc(): mostrar diagrama interacción N'p-M'p con   %
19 %                             axil en valor absoluto                      %
20 %   mostrar_diagr_interacc2(): mostrar diagrama interacción N'p-M'p      %
21 %                              sin axil en valor absoluto                 %
22 %   mostrar_areasPlastificadas(): mostrar sección con áreas              %
23 %                                 plastificadas                           %
24 %   mostrar_ley_tensiones(): mostrar distribución de tensiones de        %
25 %                            la sección                                   %
26 %   mostrar_ley_deformaciones(): mostrar distribución de deformaciones   %
27 %                                de la sección                            %
28 %   fill_area(): rellenar un polígono(simple o compuesto) de un color    %
29 %   mostrar_perfil(): mostrar perfil                                      %
30 %   simbolo_esfuerzo(): generar símbolos de la solicitación (N, M)       %
31 %   generar_informeResultados(): generar informe de resultados           %
32 %   informe_diagramaInteraccion(): generar informe del problema de       %
33 %                                  determinar el diagrama de interacción %
34 %   informe_leyTensiones(): generar informe del problema de calcular     %
35 %                           la distribución de tensiones dada una solicitación %
36 %   dimSeccNames(): dado un tipo de sección, generar un cell con los     %
```

```
37 %            nombres de las dimensiones características de la sección  %
38 %    valNamesMaterial(): dado un tipo de modelo material, generar un    %
39 %    cell con los nombres de los valores característicos del material   %
40 %                                                                        %
41 %************************************************************************%
42
43
44 function visualizacion_modelo(tipo_probl,valEntrada,guardarCaso,nombreCaso)
45 %-------------------------------------------------------------------------+
46 %
47 % Descripción función:
48 %   Visualización del modelo de entrada (geometría,material y
49 % solicitación si fuera necesario)
50 %
51 % Inputs:
52 %   tipo_probl: tipo de problema
53 %   valEntrada: valores de entrada del problema
54 %   guardarCaso: valor booleano que indica si se quiere guardar el
55 % caso(1) o no(0)
56 %   nombreCaso: (string) nombre del caso
57 %
58 % Outputs:
59 %   Depende del tipo de problema.
60 %
61 % Dependencias:
62 %   salidaDatos.mostrar_perfilConEjes, salidaDatos.mostrar_ley_mat,
63 % salidaDatos.mostrar_solicitacion
64 %-------------------------------------------------------------------------+
65   if guardarCaso,pathname=fileparts('Output_Files/'); end
66   switch tipo_probl
67     case 1
68     %valEntrada{1}: vértices del polígono que define sección
69     %valSalida{2}: ley constitutiva del material
70       figure(1)
71       salidaDatos.mostrar_perfilConEjes(valEntrada{1})
72       if guardarCaso
73         figureName=[nombreCaso,'_perfil.png'];
74         figureOut=fullfile(pathname,figureName);
75         print(gcf,figureOut,'-dpng')
76       end
77       figure(2)
78       salidaDatos.mostrar_ley_mat(valEntrada{2})
79       if guardarCaso
80         figureName=[nombreCaso,'_leyConstitutiva.png'];
81         figureOut=fullfile(pathname,figureName);
```

```
82          print(gcf,figureOut,'-dpng')
83        end
84
85      case 2
86      %valEntrada{1}: vértices del polígono que define sección
87      %valSalida{2}: ley constitutiva del material
88        figure(1)
89        salidaDatos.mostrar_perfilConEjes(valEntrada{1})
90        if guardarCaso
91          figureName=[nombreCaso,'_perfil.png'];
92          figureOut=fullfile(pathname,figureName);
93          print(gcf,figureOut,'-dpng')
94        end
95        figure(2)
96        salidaDatos.mostrar_ley_mat(valEntrada{2})
97        if guardarCaso
98          figureName=[nombreCaso,'_leyConstitutiva.png'];
99          figureOut=fullfile(pathname,figureName);
100         print(gcf,figureOut,'-dpng')
101       end
102     case 3
103     %valEntrada{1}: vértices del polígono que define sección
104     %valSalida{2}: ley constitutiva del material
105     %valSalida{3}: solicitación, [N,M]
106       figure(1)
107       salidaDatos.mostrar_solicitacion(valEntrada{1},valEntrada{3},valEntrada
              {4})
108       if guardarCaso
109         figureName=[nombreCaso,'_solicitaciones.png'];
110         figureOut=fullfile(pathname,figureName);
111         print(gcf,figureOut,'-dpng')
112       end
113       figure(2)
114       salidaDatos.mostrar_ley_mat(valEntrada{2})
115       if guardarCaso
116         figureName=[nombreCaso,'_leyConstitutiva.png'];
117         figureOut=fullfile(pathname,figureName);
118         print(gcf,figureOut,'-dpng')
119       end
120     otherwise
121       error('Tipo de problema no reconocido.')
122   end
123 end
124 %
125 %==========================================================================
```

```
126
127
128
129 function mostrar_ley_mat(ley_mat)
130 %-------------------------------------------------------------------------+
131 %
132 %  mostrar_ley_mat(ley_mat)
133 %
134 %  Desripción función:
135 %    Mostrar ley constitutiva del material
136 %
137 %  Inputs:
138 %    ley_mat: ley constitutiva del material.
139 %             Siendo el vector fila i [eps_i,sig_i]
140 %
141 %  Outputs:
142 %    Gráfica de la ley_mat
143 %
144 %  Unidades:
145 %    Tensiones - MPa
146 %    z - mm
147 %
148 %-------------------------------------------------------------------------+
149   sig_pt=max(ley_mat(:,2));
150   i_eps_pt=find(ley_mat(:,2)==sig_pt,1);%índice de eps_pt
151   eps_pt=ley_mat(i_eps_pt,1);
152   sig_pc=min(ley_mat(:,2));
153   i_eps_pc=find(ley_mat(:,2)==sig_pc,2);
154   i_eps_pc=i_eps_pc(2); %índice elemento eps_pc(2nd elemento de find)
155   eps_pc=ley_mat(i_eps_pc,1);
156
157   n=2;
158   xmax=eps_pt*n;
159   xmin=eps_pc*n;
160   dx=(xmax-xmin)*0.10;
161   ymax=sig_pt;
162   ymin=sig_pc;
163   dy=(ymax-ymin)*0.10;
164
165   plot(ley_mat(:,1),ley_mat(:,2),'r')
166   hold on
167   plot([xmin-dx,xmax+dx],[0,0],'--k')
168   plot([0,0],[ymin-dy,ymax+dy],'--k')
169   set(gca,'Xlim',[xmin-dx,xmax+dx])
170
```

```
171     a={};
172     for i=1:length(ley_mat(:,2))
173       a{i}=sprintf('%.2f',ley_mat(i,2));
174     end
175     ley_mat(:,2)=str2double(a);
176 
177     set(gca,'XTick',unique(ley_mat(:,1)))
178     set(gca,'YTick',unique(ley_mat(:,2)))
179     xlabel('\epsilon')
180     ylabel('\sigma (Mpa)')
181     title('Diagrama Tensión-Deformación')
182     grid on
183 
184     set(gca,'FontSize',8)
185 end
186 %
187 %=========================================================================
188 
189 
190 
191 function mostrar_perfilConEjes(p)
192 %-------------------------------------------------------------------------+
193 %
194 % mostrar_perfilConEjes(p)
195 %
196 % Descripción función:
197 %   Función que muestra el perfil dados sus vértices y sus ejes de
198 % referencia
199 %
200 % Inputs:
201 %   p: conjunto vértices que determinan el polígono que encierra a la
202 % sección. Siendo el vector fila i: [yi,zi]
203 %
204 % Outputs:
205 %   Gráfica que muestra el perfil
206 %
207 % Unidades:
208 %   Distancias - mm
209 %
210 %-------------------------------------------------------------------------+
211 
212     fill(p(:,1),p(:,2),'w')
213     title('Perfil')
214 
215     xmin=min(p(:,1));
```

```
216     xmax=max(p(:,1));
217     ymin=min(p(:,2));
218     ymax=max(p(:,2));
219
220     dx=(xmax-xmin)*0.1;
221     dy=(ymax-ymin)*0.1;
222
223     % Mostrar ejes
224     hold on
225     % Eje horizontal
226     plot([0,xmax+dx],[0,0],'--b')
227     plot(xmax+dx,0,'>b')
228
229     % Eje vertical
230     plot([0,0],[0,ymax+dy],'--b')
231     plot(0,ymax+dy,'^b')
232     axis('equal')
233
234     xlim([xmin-dx,xmax+dx]);
235     ylim([ymin-dy,ymax+dy]);
236
237     % ejes
238
239     xlabel('y (mm)')
240     ylabel('z (mm)')
241
242     set(gca,'FontSize',8)
243 end
244 %
245 %=========================================================================
246
247
248
249 function mostrar_solicitacion(p,N,M)
250 %-------------------------------------------------------------------------+
251 %
252 % Descripción función:
253 %   Mostrar perfil con la solicitación
254 %
255 % Inputs:
256 %   p: conjunto vértices que determinan el polígono que encierra a la
257 % sección. Siendo el vector fila i: [yi,zi]
258 %   N: valor del axil (en kN)
259 %   M: valor del momento (en kN*m)
260 %
```

```
261 % Outputs:
262 %   Gráfica que muestra el perfil con la solicitación
263 % <<: momento positivo. >>: momento negativo
264 % círculo negro: axil de tracción. Círculo con x: axil de compresión
265 %
266 % Dependencias:
267 %   salidaDatos.simbolo_esfuerzo
268 %
269 %-------------------------------------------------------------------------+
270   xmax=max(p(:,1));
271   xmin=min(p(:,1));
272   ymax=max(p(:,2));
273   ymin=min(p(:,2));
274
275   dx=(xmax-xmin)*0.1;
276   dx2=dx/2;
277   dy=(ymax-ymin)*0.1;
278
279   Nstr=sprintf('%.2f kN',N);
280   Mstr=sprintf('%.2f kN*m',M);
281   salidaDatos.mostrar_perfilConEjes(p)
282
283   ancho=xmax-xmin;
284   alto=ymax-ymin;
285   r=max(ancho,alto)/30;
286   hold on
287
288   if N>0
289     salidaDatos.simbolo_esfuerzo('N_positivo',[0,0,r])
290     text(0,dy,Nstr)
291   elseif N<0
292     salidaDatos.simbolo_esfuerzo('N_negativo',[0,0,r])
293     text(0,dy,Nstr)
294   else
295     %%No hay axil que mostrar
296   end
297
298   if M>0
299     salidaDatos.simbolo_esfuerzo('M_positivo',[0,0,1.5*r])
300     text(0,-dy,Mstr)
301   elseif M<0
302     salidaDatos.simbolo_esfuerzo('M_negativo',[0,0,1.5*r])
303     text(0,-dy,Mstr)
304   else
305     %No hay momento que mostrar
```

```
306     end
307     title('Solicitacion','FontSize',8)
308
309 end
310 %
311 %=========================================================================
312
313
314
315 function visualizar_resultados(tipo_probl,valEntrada,valSalida,guardarCaso,
        nombreCaso)
316     if guardarCaso,pathname=fileparts('Output_Files/'); end
317
318     switch tipo_probl
319
320       case 1
321         %Problema: Calcular Diagrama Momento - Curvatura
322
323         Me=valSalida{1}; %momento elástico
324         chie=valSalida{2}; %curvatura correspondiente al momento elástico
325         Mp=valSalida{3}; %momento plástico
326         d_MC=valSalida{7}; %diagrama momento - curvatura
327
328         figure
329         salidaDatos.mostrar_diagr_MC(d_MC)
330         hold on
331         plot(chie,Me,'sg')
332         plot([min(d_MC(:,1)),max(d_MC(:,1))],[Mp,Mp],'--k')
333         legend('M - \chi','(M_e,\chi_e)','M_p')
334
335         if guardarCaso
336           figureName=[nombreCaso,'_dMC.png'];
337           figureOut=fullfile(pathname,figureName);
338           print(gcf,figureOut,'-dpng')
339         end
340
341       case 2
342         %Problema: Calcular diagrama interacción(N-M)
343
344         p=valEntrada{1}; %vértices polígono sección
345         d_int=valSalida{1}; %diagrama interacción
346
347         figure
348         salidaDatos.mostrar_diagr_interacc(p,d_int)
349
```

```
350         if guardarCaso
351           figureName=[nombreCaso,'_dInteracc.png'];
352           figureOut=fullfile(pathname,figureName);
353           print(gcf,figureOut,'-dpng')
354         end
355
356       case 3
357         %Problema: calcular distribución de tensiones y deformaciones
358
359         p=valEntrada{1}; %vértices del polígono de la sección
360         N=valEntrada{3}; %valor del axil
361         M=valEntrada{4}; %valor del momento
362         z_=valSalida{1}; %vector con distancias al f.n de los puntos
363                          %característicos de la distribución de tensiones
364         sigz_=valSalida{2}; %valores de tensión de la distribución de
365                             %tensiones
366         epsz_=valSalida{3}; %valores de deformación de la distribución
367                             %de deformaciones
368         Ap1=valSalida{5}; %polígono del área plastificada superior
369         Ap2=valSalida{6}; %polígono del área plastificada inferior
370
371         if isnan(z_(1))
372           %En este caso, zfn=NaN, Sección agotada
373         else
374           figure
375
376           subplot(2,2,1)
377           salidaDatos.mostrar_solicitacion(p,N,M)
378
379           subplot(2,2,2)
380           salidaDatos.mostrar_areasPlastificadas(p,z_,Ap1,Ap2);
381
382           subplot(2,2,3)
383           ancho=max(p(:,1))-min(p(:,1));
384           alto=max(p(:,2))-min(p(:,2));
385
386           salidaDatos.mostrar_ley_deformaciones([epsz_,z_])
387           if alto/ancho<=1.05, pbaspect([ancho, alto, 1]); end
388
389           subplot(2,2,4)
390           salidaDatos.mostrar_ley_tensiones([sigz_,z_])
391           if alto/ancho<=1.05, pbaspect([ancho, alto, 1]); end
392
393           if guardarCaso
394             figureName=[nombreCaso,'_leyTenAndDef.png'];
```

```
395             figureOut=fullfile(pathname,figureName);
396             print(gcf,figureOut,'-dpng')
397           end
398         end
399
400       otherwise
401         error('No se reconoce tipo de problema.')
402     end
403
404 end
405 %
406 %=========================================================================
407
408
409
410 function mostrar_diagr_MC(d_MC)
411 %-------------------------------------------------------------------------+
412 %
413 %  mostrar_diagr_MC(d_MC)
414 %
415 %  Descripción función:
416 %    Mostrar el diagrama Momento - Curvatura
417 %
418 %  Inputs:
419 % d_MC: diagrama Momento - Curvatura
420 %         d_MC(:,1): valores curvatura
421 %         d_MC(:,2): valores momento
422 %
423 %  Outputs:
424 %    Muestra diagrama Momento - Curvatura
425 %
426 %  Unidades:
427 %    Curvatura - rad/m
428 %    Momentos - kN*m
429 %
430 %-------------------------------------------------------------------------+
431
432   plot([0;d_MC(:,1)],[0;d_MC(:,2)],'*-')
433   title('Diagrama Momento-Curvatura')
434   xlabel('\chi(rad/m)')
435   ylabel('M (kN*m)')
436   grid on
437 end
438 %
439 %=========================================================================
```

```
440
441
442
443 function mostrar_diagr_interacc(p,d_int)
444 %+-----------------------------------------------------------------------+
445 %
446 % mostrar_diagr_interacc(d_int)
447 %
448 % Descripción función:
449 %   Mostrar el diagrama Interacción Np'-Mp' con los valores del axil
450 % en valor absoluto
451 %
452 % Inputs:
453 % d_int: diagrama momento interacción
454 %       d_int(:,1): valores Np'
455 %       d_int(:,2): valores Mp'
456 %
457 % Outputs:
458 %   Muestra diagrama interacción Np'-Mp'
459 %
460 % Unidades:
461 %   Fuerzas - kN
462 %   Momentos - kN*m
463 %
464 % Dependencias:
465 %   salidaDatos.mostrar_perfil, salidaDatos.simbolo_esfuerzo
466 %
467 %+-----------------------------------------------------------------------+
468
469   Np_abs=abs(d_int(:,1));
470
471   ymin=min(d_int(:,2));
472   ymax=max(d_int(:,2));
473   dy=(ymax-ymin)*0.1;
474
475   xmin=min(Np_abs);
476   xmax=max(Np_abs);
477   dx=(xmax-xmin)*0.1;
478
479   subplot(6,6,[1:5,7:11,13:17,19:23,25:29,31:35])
480   plot(Np_abs,d_int(:,2),'-')
481   title('Diagrama interacción N_p-M_p')
482   xlabel('|N_p| (kN)')
483   ylabel('M_p (kN*m)')
484   hold on
```

```
485     plot([0,0],[ymin,ymax+dy],':k')
486     plot([xmin,xmax+dx],[0,0],':k')
487
488     subplot(6,6,6)
489     ancho=max(p(:,1))-min(p(:,1));
490     alto=max(p(:,2))-min(p(:,2));
491     r=max(ancho,alto)/10;
492     salidaDatos.mostrar_perfil(p)
493     title(' ')
494     hold on
495     salidaDatos.simbolo_esfuerzo('M_positivo',[0,0,1.5*r])
496     if d_int(5,1)>0
497       salidaDatos.simbolo_esfuerzo('N_positivo',[0,0,r])
498     else
499       salidaDatos.simbolo_esfuerzo('N_negativo',[0,0,r])
500     end
501     xlim([min(p(:,2))-r/2,max(p(:,1))+r/2])
502     axis('off')
503
504     set(gca,'FontSize',8)
505
506 end
507 %
508 %=========================================================================
509
510
511
512 function mostrar_diagr_interacc2(d_int)
513 %-------------------------------------------------------------------------+
514 %
515 %  mostrar_diagr_interacc(d_int)
516 %
517 %  Descripción función:
518 %    Mostrar el diagrama Interacción Np'-Mp'
519 %
520 %  Inputs:
521 %  d_int: diagrama momento interacción
522 %         d_int(:,1): valores Np'
523 %         d_int(:,2): valores Mp'
524 %
525 %  Outputs:
526 %    Muestra diagrama interacción Np'-Mp'
527 %
528 %  Unidades:
529 %    Fuerzas - kN
```

```
530 %     Momentos - kN*m
531 %
532 %-------------------------------------------------------------------------+
533
534     plot(d_int(:,1),d_int(:,2),'-')
535     title('Diagrama interacción N_p-M_p')
536     xlabel('N_p (kN)')
537     ylabel('M_p (kN*m)')
538     hold on
539
540     ymin=min(d_int(:,2));
541     ymax=max(d_int(:,2));
542     dy=(ymax-ymin)*0.1;
543
544     xmin=min(d_int(:,1));
545     xmax=max(d_int(:,1));
546     dx=(xmax-xmin)*0.1;
547
548     plot([0,0],[ymin,ymax+dy],':k')
549     plot([xmin,xmax+dx],[0,0],':k')
550
551     set(gca,'FontSize',8)
552
553 end
554 %
555 %=========================================================================
556
557
558 function mostrar_areasPlastificadas(p,z_,Ap1,Ap2)
559 %-------------------------------------------------------------------------+
560 %
561 %  mostrar_areasPlastificadas(p,z_,Ap1,Ap2)
562 %
563 %  Descripción función:
564 %    Función que muestra las áreas plastificadas
565 %
566 %  Inputs:
567 %    p: sección de la que queremos representar sus áreas plastificadas
568 %    z_: valores z_ del diagrama de tensiones(referidos a la posición de
569 %        la fibra neutra)
570 %    Ap1: área plastificada que queda por encima de la f.n.
571 %    Ap2: área plastificada que queda por debajo de la f.n.
572 %
573 %  Dependencias:
574 %    salidaDatos.mostrar_perfil, salidaDatos.fill_area
```

```
575 %
576 %-------------------------------------------------------------------------+
577   if size(z_,1)<2, Ap1=[]; Ap2=[]; return; end
578   zmin_p=min(p(:,2));
579   ymin_p=min(p(:,1));
580
581   %Calcular ze1 y ze2
582   ze1=z_(find(z_==0)+1);
583   ze2=z_(find(z_==0)-1);
584
585   % Calcular zfn
586   if length(z_)==2, zfn=1E16; else zfn=min(z_);end
587   if zfn>0; zfn=-zfn; else zfn=abs(zfn);end
588
589  %Ponemos (0,0) en esquina inferior izquierda
590   p(:,1)=p(:,1)-ymin_p;
591   p(:,2)=p(:,2)-zmin_p;
592
593   salidaDatos.mostrar_perfil(p)
594   hold on
595   salidaDatos.fill_area(Ap1,'y')
596   salidaDatos.fill_area(Ap2,'y')
597   z=z_+zfn;
598
599   z=unique(z);
600   z_show=[z(1),z(end),zfn];
601   if length(Ap1)~=0, z_show=[z_show,ze1+zfn]; end
602   if length(Ap2)~=0, z_show=[z_show,ze2+zfn]; end
603
604 % Ponemos cifras con 2 cifras decimales
605   for i=1:length(z_show)
606     a{i}=sprintf('%.2f',z_show(i));
607   end
608   z_show=str2double(a);
609
610   set(gca,'YTick',unique(z_show))
611   %Representar eje fibra neutra
612   xmin=min(p(:,1));
613   xmax=max(p(:,1));
614   plot([xmin,xmax],[zfn,zfn],'--')
615   dx=(xmax-xmin)*0.1;
616   text(xmax-dx,zfn,'f.n.')
617
618   title('Perfil con areas plastificadas')
619
```

```
620     grid on
621     set(gca,'FontSize',8)
622     hold off
623 
624 end
625 %
626 %=========================================================================
627 
628 
629 function mostrar_ley_tensiones(sigma_z)
630 %-------------------------------------------------------------------------+
631 %
632 %  mostrar_ley_tensiones(sigma_z)
633 %
634 %  Desripción función:
635 %
636 %  Función que muestra la distribución de tensiones
637 %
638 %  Inputs:
639 %    sigma_z: matriz que representa los puntos característicos de
640 %             la ley de tensiones, de la forma: [sigma_i,z_i]
641 %
642 %  Outputs:
643 %    Diagrama de tensiones del problema
644 %
645 %  Unidades:
646 %    Tensiones - MPa
647 %
648 %-------------------------------------------------------------------------+
649     if size(sigma_z,1)<2, return; end
650     z_=sigma_z(:,2);
651 
652     zmin_=min(z_);
653     sigma_z(:,2)=z_-zmin_; %Trasladamos el 0 en el extremo inferior
654 
655     plot(sigma_z(:,1),sigma_z(:,2),'b')
656     hold on;
657     %Ejes
658     ylabel('z (mm)')
659     ymax=max(sigma_z(:,2));
660     ymin=min(sigma_z(:,2));
661     plot([0,0],[ymin,ymax],'b')
662     xlabel('\sigma (MPa)')
663     xmax=max(sigma_z(:,1));
664     xmin=min(sigma_z(:,1));
```

```
665
666   dy=(ymax-ymin)*0.1;
667   if (length(z_)==2 && sigma_z(1,1)==sigma_z(2,1)), dx=abs(sigma_z(1,1))*0.1;
          else dx=(xmax-xmin)*0.1; end
668
669
670   %Eje fibra neutra
671   z_fn=sigma_z(find(sigma_z(:,1)==0),2);
672   %Si length(z_fn)==0, z_fn no cae sobre sección
673   if length(z_fn)==1
674     plot([xmax,xmin],[z_fn,z_fn],'--g')
675     text(xmax-dx,z_fn,'f.n.')
676   end
677   %Representar vectores sigma(z)
678   for i=1:length(sigma_z)
679     plot([0,sigma_z(i,1)],[sigma_z(i,2),sigma_z(i,2)],'b')
680     if sigma_z(i,1)<0
681       plot(sigma_z(i,1),sigma_z(i,2),'<b')
682     elseif sigma_z(i,1)>0
683       plot(sigma_z(i,1),sigma_z(i,2),'>b')
684     else
685       %Si es ==0 no se representa punta de flecha
686     end
687   end
688
689   % Ponemos cifras con 2 cifras decimales
690   for i=1:length(sigma_z(:,1))
691     a{i}=sprintf('%.2f',sigma_z(i,1));
692   end
693   sigma_z(:,1)=str2double(a);
694   a={};
695     for i=1:length(sigma_z(:,2))
696     a{i}=sprintf('%.2f',sigma_z(i,2));
697   end
698   sigma_z(:,2)=str2double(a);
699
700   %%Casos en los ue la f.n. cae fuera de la sección
701   if xmin<0 && xmax<0, xmax=0; end
702   if xmin>0 && xmax>0, xmin=0; end
703   %%
704
705   if xmin~=xmax, xlim([xmin-dx,xmax+dx]), end
706   if ymin~=ymax, ylim([ymin-dy,ymax+dy]), end
707
708   set(gca,'XTick',unique(sigma_z(:,1)))
```

```
709     set(gca,'YTick',unique(sigma_z(:,2)))
710     title('Diagrama de tensiones')
711     grid on
712 
713     set(gca,'FontSize',8)
714 end
715 %
716 %=========================================================================
717 
718 
719 function mostrar_ley_deformaciones(epsilon_z)
720 %-------------------------------------------------------------------------+
721 %
722 %  mostrar_ley_deformaciones(epsilon_z)
723 %
724 %  Desripción función:
725 %
726 %  Función que muestra el diagrama de deformaciones
727 %
728 %  Inputs:
729 %    epsilon_z: matriz que representa los puntos característicos de
730 %               la ley de deformaciones, de la forma: [epsilon_i,z_i]
731 %
732 %  Outputs:
733 %    Diagrama de deformaciones del problema
734 %
735 %  Unidades:
736 %    z - mm
737 %
738 %-------------------------------------------------------------------------+
739     if size(epsilon_z,1)<2, return; end
740     z_=epsilon_z(:,2);
741 
742     if isnan(epsilon_z(1,1))
743       plot([-1,1],[0,0],'m')
744       title('Diagrama Deformaciones')
745       return
746     end
747     zmin_=min(z_);
748     epsilon_z(:,2)=z_-zmin_; %Trasladamos el 0 en el extremo inferior
749 
750     plot(epsilon_z(:,1),epsilon_z(:,2),'m')
751     hold on;
752     %Ejes
753     ylabel('z (mm)')
```

```
754   ymax=max(epsilon_z(:,2));
755   ymin=min(epsilon_z(:,2));
756   plot([0,0],[ymin,ymax],'m')
757   xlabel('\epsilon')
758   xmax=max(epsilon_z(:,1));
759   xmin=min(epsilon_z(:,1));
760
761
762   dy=(ymax-ymin)*0.1;
763   if (length(z_)==2 && epsilon_z(1,1)==epsilon_z(2,1)), dx=abs(epsilon_z(1,1))
          *0.1; else dx=(xmax-xmin)*0.1; end
764
765   % Eje fibra neutra
766   z_fn=epsilon_z(find(epsilon_z(:,1)==0),2);
767   % Si length(z_fn)==0, z_fn no cae sobre sección
768   if length(z_fn)==1
769     plot([xmax,xmin],[z_fn,z_fn],'--g')
770     text(xmax-dx,z_fn,'f.n.')
771     epsilon_z=[epsilon_z(1,:);[0,z_fn];epsilon_z(end,:)];
772   else
773     % si f.n. no cae sobre la sección
774     epsilon_z=[epsilon_z(1,:);epsilon_z(end,:)];
775   end
776   % Representar vectores epsilon(z)
777   for i=1:length(epsilon_z)
778     plot([0,epsilon_z(i,1)],[epsilon_z(i,2),epsilon_z(i,2)],'m')
779     if epsilon_z(i,1)<0
780       plot(epsilon_z(i,1),epsilon_z(i,2),'<m')
781     elseif epsilon_z(i,1)>0
782       plot(epsilon_z(i,1),epsilon_z(i,2),'>m')
783     else
784       % Si es ==0 no se representa punta de flecha
785     end
786   end
787
788 %% Casos en los que la f.n. cae fuera de la sección
789   if xmin<0 && xmax<0, xmax=0; end
790   if xmin>0 && xmax>0, xmin=0; end
791 %%
792
793   if xmin~=xmax, xlim([xmin-dx,xmax+dx]), end
794   if ymin~=ymax, ylim([ymin-dy,ymax+dy]), end
795
796   if ~isnan(epsilon_z(1))
797   set(gca,'XTick',unique(epsilon_z(:,1)))
```

```
798   end
799   set(gca,'YTick',unique(epsilon_z(:,2)))
800   title('Diagrama de deformaciones')
801   grid on
802
803   set(gca,'FontSize',8)
804 end
805 %
806 %=========================================================================
807
808
809
810 function fill_area(A,color)
811 %-------------------------------------------------------------------------+
812 %
813 % fill_area(A,color)
814 %
815 % Descripción función:
816 %   Función que muestra un área coloreada
817 %
818 % Inputs:
819 %   A: polígono que encierra el área que queremos rellenar (puede
820 % ser simple o compuesto)
821 %   color: color del área
822 %
823 % Outputs:
824 %   Muestra área coloreada
825 %
826 %-------------------------------------------------------------------------+
827   if iscell(A)
828     for i=1:length(A)
829       if length(A{i})==0
830         %no hay nada que mostrar
831       else
832         fill(A{i}(:,1),A{i}(:,2),color);hold on
833       end
834     end
835   else
836     if length(A)==0
837       %No hay nada que mostrar
838     else
839       fill(A(:,1),A(:,2),color)
840     end
841   end
842 end
```

```
843 %
844 %=========================================================================
845
846
847
848 function mostrar_perfil(p)
849 %-------------------------------------------------------------------------+
850 %
851 %  mostrar_perfil(p)
852 %
853 %  Descripción función:
854 %    Función que muestra el perfil dados sus puntos
855 %
856 %  Inputs:
857 %    p: conjunto vértices que determinan el polígono que encierra a la
858 %  sección. Siendo el vector fila i: [yi,zi]
859 %
860 %  Outputs:
861 %    Representación del perfil
862 %
863 %  Unidades:
864 %    Distancias - mm
865 %
866 %-------------------------------------------------------------------------+
867
868    fill(p(:,1),p(:,2),'w')
869    title('Perfil')
870
871    xmin=min(p(:,1));
872    xmax=max(p(:,1));
873    ymin=min(p(:,2));
874    ymax=max(p(:,2));
875
876    axis('equal')
877
878    dx=(xmax-xmin)*0.1;
879    dy=(ymax-ymin)*0.1;
880
881    xlim([xmin-dx,xmax+dx]);
882    ylim([ymin-dy,ymax+dy]);
883
884    xlabel('y (mm)')
885    ylabel('z (mm)')
886
887    set(gca,'FontSize',8)
```

```
888 end
889 %
890 %=========================================================================
891
892
893 function simbolo=simbolo_esfuerzo(esfuerzo,valores)
894 %-------------------------------------------------------------------------+
895 %
896 % simbolo=simbolo_esfuerzo(esfuerzo,valores)
897 %
898 % Descripción función:
899 %   Función que genera el símbolo de un esfuerzo de la solicitación
900 %
901 % Inputs:
902 %   esfuerzo: (string) que puede ser: <N_positivo: axil de tracción,
903 % N_negativo: axil de compresión, M_positivo: M>0,
904 % M_negativo: M<0 >
905 %   valores: array
906 %     valores(1): Cx (coordenada x posición símbolo)
907 %     valores(2): Cy (coordenada y posición símbolo)
908 %     valores(3): tamaño del símbolo
909 %
910 % Outputs:
911 %   símbolo(plot)
912 %
913 %-------------------------------------------------------------------------+
914   switch esfuerzo
915
916     case 'N_positivo'
917       Cx=valores(1);
918       Cy=valores(2);
919       r=valores(3);
920       tita=linspace(0,2*pi,50);
921       p=([r*cos(tita)+Cx;r*sin(tita)+Cy])';
922       fill(p(:,1),p(:,2),'k')
923
924     case 'N_negativo'
925       Cx=valores(1);
926       Cy=valores(2);
927       r=valores(3);
928       tita=linspace(0,2*pi,50);
929       p=([r*cos(tita)+Cx;r*sin(tita)+Cy])';
930       fill(p(:,1),p(:,2),'w')
931       hold on
932       plot([Cx+r*cos(pi/4),Cx+r*cos(5*pi/4)],[Cy+r*sin(pi/4),Cy+r*sin(5*pi/4)],'
```

```
        k')
933     plot([Cx+r*cos(3*pi/4),Cx+r*cos(7*pi/4)],[Cy+r*sin(3*pi/4),Cy+r*sin(7*pi
            /4)],'k')
934
935   case 'M_positivo'
936     Cx=valores(1);
937     Cy=valores(2);
938     h=valores(3);
939     tita=20*pi/180;
940
941     p=[[Cx-4*h,Cy];[Cx+h-4*h,Cy-h*tan(tita)];[Cx+h-4*h,Cy];[2*h-4*h+Cx,Cy-h*
            tan(tita)];...
942       [Cx+2*h-4*h,Cy+h*tan(tita)];[Cx+h-4*h,Cy];[Cx+h-4*h,Cy+h*tan(tita)];[Cx
            -4*h,Cy]];
943
944     plot([Cx-4*h,Cx],[Cy,Cy],'k')
945     hold on
946     fill(p(:,1),p(:,2),'k')
947
948
949
950   case 'M_negativo'
951     Cx=valores(1);
952     Cy=valores(2);
953     h=valores(3);
954     tita=20*pi/180;
955     p=[[Cx+4*h,Cy];[Cx-h+4*h,Cy-h*tan(tita)];[Cx-h+4*h,Cy];[-2*h+4*h+Cx,Cy-h*
            tan(tita)];...
956       [Cx-2*h+4*h,Cy+h*tan(tita)];[Cx-h+4*h,Cy];[Cx-h+4*h,Cy+h*tan(tita)];[Cx
            +4*h,Cy]];
957
958     plot([Cx,Cx+4*h],[Cy,Cy],'k')
959     hold on
960     fill(p(:,1),p(:,2),'k')
961
962
963   otherwise
964     error('No se reconoce tipo de esfuerzo')
965  end
966 end
967 %
968 %=========================================================================
969
970
971 function generar_informeResultados(tipo_probl,datEntrada,valSalida,guardarCaso,
```

```
    nombreCaso)
%+-------------------------------------------------------------------------+
%    Función que genera el informe de resultados en función del
% tipo de problema, los valores de entrada y resultados
%+-------------------------------------------------------------------------+
  switch tipo_probl

  case 1

    tipo_secc=datEntrada{1};
    dim=datEntrada{2};
    tipo_mat=datEntrada{3};
    valores=datEntrada{4};

    signoM='p'; % por defecto, M>0

    Me=valSalida{1};
    chie=valSalida{2};
    Mp=valSalida{3};
    ff=valSalida{4};
    signo_zp=valSalida{5};
    condic=valSalida{6};
    d_MC=valSalida{7};

    if guardarCaso
      fileName=[nombreCaso,'_dMC.txt'];
      pathname=fileparts('Output_Files/');
      fileOut=fullfile(pathname,fileName);
      file=fopen(fileOut,'w');
      salidaDatos.informe_problemaMC(tipo_secc,dim,tipo_mat,valores,signoM,Me,
          chie,Mp,ff,d_MC,signo_zp,condic,file)
      fclose(file);
    end
    salidaDatos.informe_problemaMC(tipo_secc,dim,tipo_mat,valores,signoM,Me,chie
        ,Mp,ff,d_MC,signo_zp,condic,1)

  case 2

    tipo_secc=datEntrada{1};
    dim=datEntrada{2};
    tipo_mat=datEntrada{3};
    valores=datEntrada{4};
    signoN=datEntrada{5};
    signoM='p'; % por defecto, M>0

```

```
1014      d_int=valSalida{1};
1015      Np=valSalida{2};
1016      Mp=valSalida{3};
1017
1018      if guardarCaso
1019        fileName=[nombreCaso,'_dinteracc.txt'];
1020        pathname=fileparts('Output_Files/');
1021        fileOut=fullfile(pathname,fileName);
1022        file=fopen(fileOut,'w');
1023        salidaDatos.informe_diagramaInteraccion(tipo_secc,dim,tipo_mat,valores,
              signoN,signoM,Np,Mp,d_int,file)
1024        fclose(file);
1025      end
1026      salidaDatos.informe_diagramaInteraccion(tipo_secc,dim,tipo_mat,valores,
            signoN,signoM,Np,Mp,d_int,1)
1027
1028    case 3
1029
1030      tipo_secc=datEntrada{1};
1031      dim=datEntrada{2};
1032      tipo_mat=datEntrada{3};
1033      valores=datEntrada{4};
1034      N=datEntrada{5};
1035      M=datEntrada{6};
1036
1037      z_=valSalida{1};
1038      sigz_=valSalida{2};
1039      chi=valSalida{4};
1040
1041      if guardarCaso
1042        fileName=[nombreCaso,'_leyTensiones.txt'];
1043        pathname=fileparts('Output_Files/');
1044        fileOut=fullfile(pathname,fileName);
1045        file=fopen(fileOut,'w');
1046        salidaDatos.informe_leyTensiones(tipo_secc,dim,tipo_mat,valores,N,M,z_,
              sigz_,chi,file)
1047        fclose(file);
1048
1049      end
1050      salidaDatos.informe_leyTensiones(tipo_secc,dim,tipo_mat,valores,N,M,z_,sigz_
            ,chi,1)
1051
1052    otherwise
1053
1054      error('No se reconoce tipo de problema.')
```

```
1055
1056    end
1057
1058  end
1059  %
1060  %=========================================================================
1061
1062
1063  function informe_problemaMC(tipo_secc,dim,tipo_mat,valores,signoM,Me,chie,Mp,ff,
        d_MC,signo_zp,condic,f)
1064  %-------------------------------------------------------------------------+
1065  %    Generar el informe del problema de calcular el diagrama
1066  % momento - curvatura
1067  %-------------------------------------------------------------------------+
1068      %Entrada de valores
1069       fprintf(f,'\n +------- Cálculo Diagrama Momento - Curvatura -------+')
1070       fprintf(f,'\n *** VALORES ENTRADA ***')
1071       fprintf(f,'\n distancias(mm), tensiones(MPa), E(Mpa)')
1072       fprintf(f,'\nTipo de sección: %s',tipo_secc)
1073       dimNames=salidaDatos.dimSeccNames(tipo_secc);
1074       valNames=salidaDatos.valNamesMaterial(tipo_mat);
1075       fprintf(f,'\nDimensiones sección:')
1076       for i=1:length(dim)
1077         fprintf(f,'\n   %s: %.2f mm',dimNames{i},dim(i))
1078       end
1079
1080       fprintf(f,'\nTipo de Material: %s',tipo_mat)
1081       fprintf(f,'\nValores del material:')
1082       fprintf(f,'\nTensiones (MPa)\n')
1083       for i=1:length(valores)
1084         fprintf(f,'\n   %s: %.6f ',valNames{i},valores(i))
1085       end
1086
1087       %Presentar valores característicos por consola
1088       if condic==1
1089         cara='superior';
1090         if signo_zp=='c', tipo_zp='compresión';else, tipo_zp='tracción';end
1091       else
1092         cara='inferior';
1093         if signo_zp=='c', tipo_zp='compresión';else, tipo_zp='tracción';end
1094       end
1095
1096       fprintf(f,'\n *** RESULTADOS *** ')
1097       fprintf(f,'\nMomento elástico: %.4f kN*m',Me)
1098       fprintf(f,'\nCurvatura(M=Me): %.6f rad/m',chie)
```

```
1099      fprintf(f,'\nMomento plástico: %.4f kN*m',Mp)
1100      fprintf(f,'\nFactor de forma: %f',ff)
1101
1102      fprintf(['\nProfundidad de plastificación(zp) de %s ', ...
1103                       'respecto cara %s\r\n'],tipo_zp,cara)
1104      fprintf(f,'\n Diagrama Momento-Curvatura')
1105      fprintf(f,'\n%12s    %12s    %12s\r\n','zp(mm)', ...
1106                       'chi(rad/m)','M(kN*m)')
1107     d_MC=[d_MC(:,3),d_MC(:,1),d_MC(:,2)]; %seleccionamos filas zp, chi,M
1108      for i=1:length(d_MC)
1109        fprintf(f,'%12.4f    %12.4f    %12.4f \r\n',d_MC(i,1),d_MC(i,2),d_MC(i,3))
1110      end
1111
1112 end
1113 %
1114 %=========================================================================
1115
1116
1117
1118
1119 function informe_diagramaInteraccion(tipo_secc,dim,tipo_mat,valores,signoN,
      signoM,Np,Mp,d_int,f)
1120 %-------------------------------------------------------------------------+
1121 %    Generar el informe del problema de calcular el diagrama
1122 %  interacción axil - momento
1123 %-------------------------------------------------------------------------+
1124      %Entrada de valores
1125      fprintf(f,'\n +------- Cálculo Diagrama Interacción N p-M p -------+')
1126      fprintf(f,'\n *** VALORES ENTRADA ***')
1127      fprintf(f,'\n distancias(mm), tensiones(MPa), E(Mpa)')
1128      fprintf(f,'\nTipo de sección: %s',tipo_secc)
1129      dimNames=salidaDatos.dimSeccNames(tipo_secc);
1130      valNames=salidaDatos.valNamesMaterial(tipo_mat);
1131      fprintf(f,'\nDimensiones sección:\n')
1132      for i=1:length(dim)
1133        fprintf(f,'  %s: %.2f mm\r\n',dimNames{i},dim(i))
1134      end
1135
1136      fprintf(f,'\nTipo de Material: %s',tipo_mat)
1137      fprintf(f,'\nValores del material:')
1138      fprintf(f,'\nTensiones (MPa)\n')
1139      for i=1:length(valores)
1140        fprintf(f,'  %s: %.6f \r\n',valNames{i},valores(i))
1141      end
1142      fprintf(f,'\nSigno del axil: %s',signoN)
```

```
1143      fprintf(f,'\nSigno del momento: %s',signoM)
1144      fprintf(f,'\n *** RESULTADOS *** ')
1145      fprintf(f,'\nAxil plástico (Mp=0): %.4f kN',Np)
1146      fprintf(f,'\nMomento plástico (Np=0): %.4f kN*m',Mp)
1147
1148      fprintf(f,'\nDiagrama Interacción N p-M p')
1149      fprintf(f,'\n %12s  %12s  %12s\r\n','zfnp(mm)','N p(kN)','M p(kn*m)')
1150      for i=1:length(d_int)
1151        fprintf(f,'%12.4f  %12.4f  %12.4f\r\n',d_int(i,3),d_int(i,1),d_int(i,2))
1152      end
1153
1154  end
1155  %
1156  %=========================================================================
1157
1158
1159
1160  function informe_leyTensiones(tipo_secc,dim,tipo_mat,valores,N,M,z_,sigz_,chi,f)
1161  %-------------------------------------------------------------------------+
1162  %   Generar el informe del problema de calcular la distribución de
1163  % tensiones y deformaciones
1164  %-------------------------------------------------------------------------+
1165
1166      fprintf(f,'\n +------ Cálculo Ley de Tensiones ------+')
1167      fprintf(f,'\n *** VALORES ENTRADA ***')
1168      fprintf(f,'\n distancias(mm), tensiones(MPa), E(Mpa)')
1169      fprintf(f,'\nTipo de seción: %s',tipo_secc)
1170      dimNames=salidaDatos.dimSeccNames(tipo_secc);
1171      valNames=salidaDatos.valNamesMaterial(tipo_mat);
1172      fprintf(f,'\nDimensiones sección:\n')
1173      for i=1:length(dim)
1174        fprintf(f,'  %s: %.2f\r\n',dimNames{i},dim(i))
1175      end
1176
1177      fprintf(f,'\nTipo de Material: %s',tipo_mat)
1178      fprintf(f,'\nValores del material:')
1179      fprintf(f,'\nTensiones (MPa)\n')
1180      for i=1:length(valores)
1181        fprintf(f,'  %s: %.6f\r\n',valNames{i},valores(i))
1182      end
1183      fprintf(f,'\nSolicitación de la sección')
1184      fprintf(f,'\nAxil: %.2f kN*m',N)
1185      fprintf(f,'\nMomento: %.2f kN*m',M)
1186      fprintf(f,'\n *** RESULTADOS *** ')
1187      fprintf(f,'\nLey de Tensiones\n')
```

```
1188      fprintf(f,'%12s    %12s   \r\n','z(mm)', ...
1189                        'sigma(MPa)')
1190      zmin_=min(z_);
1191      z=z_-zmin_; %Ponemos el 0 en la fibra inferior
1192      %sigz=[z,sigz_];
1193      for i=1:length(z)
1194        fprintf(f,'%12.4f    %12.4f\r\n',z(i),sigz_(i))
1195      end
1196      fprintf(f,'\n Curvatura= %f rad/m',chi)
1197  end
1198  %
1199  %=========================================================================
1200
1201
1202
1203  function dimNames=dimSeccNames(tipo)
1204  %-------------------------------------------------------------------------+
1205  %    Dado un tipo de sección, la función genera un cell que contiene los
1206  % nombres de las dimensiones características de dicha sección
1207  %-------------------------------------------------------------------------+
1208    switch tipo
1209      case 'R'
1210        dimNames={'b','h'};
1211      case 'C'
1212        dimNames={'r'};
1213      case 'I'
1214        dimNames={'h','bf','tf','tw'};
1215      case 'H'
1216        dimNames={'hf','b','tf','tw'};
1217      case 'T'
1218        dimNames={'h','bf','tf','tw'};
1219      case 'Inb'
1220        dimNames={'h','bf1','tf1','tw','bf2','tf2'};
1221      case 'Rh'
1222        dimNames={'b','h','e'};
1223      case 'Ch'
1224        dimNames={'r','e'};
1225      case 'IPN'
1226        dimNames={'h','b','r1','tf','r2','d'};
1227      case 'IPE'
1228        dimNames={'h','bf','tw','tf','r','d'};
1229      case 'Rombo'
1230        dimNames={'dv','dh'};
1231      case 'cajon'
1232        dimNames={'b','b1','tf1','hw','tw','tf2'};
```

```
1233      otherwise
1234        error('No existe ese tipo de sección.')
1235    end
1236
1237  end
1238  %
1239  %=========================================================================
1240
1241
1242  function valNames=valNamesMaterial(tipo_mat)
1243  %-------------------------------------------------------------------------+
1244  %    Dado un tipo de material, la función genera un cell que contiene los
1245  % nombres de las propiedades características del material
1246  %-------------------------------------------------------------------------+
1247    switch tipo_mat
1248      case 'G1'
1249        valNames={'sig_p(MPa)','E(MPa)'};
1250      case 'G2'
1251        valNames={'E(MPa)','sig_pc(Mpa)','sig_pt(MPa)'};
1252      case 'G3'
1253        valNames={'eps_e','sig_e(MPa)','eps_p1','sig_p1(MPa)','eps_p2','sig_p2(MPa
              )'};
1254      otherwise
1255        error('No se reconoce ese tipo de material.')
1256     end
1257
1258  end
1259  %
1260  %=========================================================================
1261
1262
1263  end
1264  end
```

## B.3. *logic/*

Módulo de análisis.

### B.3.1. *main_logic()*

```
1  function [valEntrada,valSalida]=main_logic(tipoProblema,datEntrada)
2
3    %% Crear modelo geométrico
4    % Siendo:
5    %   datEntrada{1}: tipo de sección (T, I, etc)
```

```
6   %   datEntrada{2}: dimensiones características de la sección
7   valEntrada{1}=generacion_Modelo.verticesPoly(datEntrada{1},datEntrada{2});
8
9   %% Crear modelo material
10  % Siendo:
11  %    datEntrada{3}: tipo de modelo del material
12  %    datEntrada{4}: valores característicos del modelo de material
13  valEntrada{2}=generacion_Modelo.comportMaterial(datEntrada{3},datEntrada{4});
14
15  %% Acondicionar los datos de entrada para el Análisis
16  % Cambiar sistema de coorderadas (referir puntos al c.d.g)
17  % Cambiar sistema de unidades : tesiones (MPa --> kPa),
18  % distancias(m --> mm)
19  valEntrada=acondicionamientoDatos.datosEntrada(tipoProblema,valEntrada);
20
21  switch tipoProblema
22
23  case 1
24    % Problema: Calcular Diagrama Momento - Curvatura
25
26    p=valEntrada{1}; % vértices del polígono de la sección
27    ley_mat=valEntrada{2}; % ley constituiva del material
28    signoM='p'; % por defecto, M>0
29
30    %Cálculo de valores característicos de la curva (M-Chi)
31    N=0; % por defecto, flexión pura
32    [Mp,zfnp,zpmax,signo_zp,condic]=problFlexion.N_Mp(p,ley_mat,N,signoM);
33    [Me,~,chie]=problFlexion.N_Me(p,ley_mat,N,signoM);
34    ff=Mp/Me;
35
36    %Cálculo del diagrama M-Chi
37    d_MC=problFlexion.N_diagramaMC(p,ley_mat,0,signoM); %N=0
38
39    % Especificar valores de salida
40    valSalida{1}=Me; % momento elástico
41    valSalida{2}=chie; % curvatura correspondiente al momento elástico
42    valSalida{3}=Mp; % momento plástico
43    valSalida{4}=ff; % factor de forma
44    valSalida{5}=signo_zp; % carácter de la profundidad de plastificación
45                           % de referencia(compresión/tracción)
46    valSalida{6}=condic; % desde qué extremo (superior/inferior) se mide
47                         % la profundidad de plastificación de referencia
48    valSalida{7}=d_MC; % diagrama momento - curvatura
49
50    % Cambio de unidades a la salida
```

```
51      [valEntrada,valSalida]=acondicionamientoDatos.datosSalida(tipoProblema,
            valEntrada,valSalida);
52
53    case 2
54      %Problema: Calcular el Diagrama de Interacción N-M de la sección
55
56      p=valEntrada{1}; %vértices del polígono de la sección
57      ley_mat=valEntrada{2}; %ley constitutiva del material
58      signoN=datEntrada{5}; %indica si el axil del diagrama es de
59                            %tracción (N>0) o compresión(N<0)
60
61      valEntrada{3}=signoN;
62
63      signoM='p'; %por defecto, se establece M>0
64
65      %% Calcular diagrama de interacción N'p-M'p, axil y momento plástico (Np,Mp)
66      [d_int,Np,Mp]=problFlexion.diagrama_interaccionNM(p,ley_mat,signoN,signoM);
67
68      %Especificar valores de salida
69      valSalida{1}=d_int; %diagrama de interacción
70      valSalida{2}=Np; %axil de agotamiento (axil puro)
71      valSalida{3}=Mp; %momento de agotamiento (flexión pura)
72
73      %Cambio de unidades a la salida
74      [valEntrada,valSalida]=acondicionamientoDatos.datosSalida(tipoProblema,
            valEntrada,valSalida);
75
76    case 3
77      %Problema: Calcular distribución de tensiones y deformaciones
78
79      p=valEntrada{1}; %vértices del polígono de la sección
80      ley_mat=valEntrada{2}; %ley constitutiva del material
81      N=datEntrada{5}; %valor del axil
82      M=datEntrada{6}; %valor del momento
83
84      valEntrada{3}=N;
85      valEntrada{4}=M;
86
87      %Calcular posición de la fibra neutra y curvatura (parámetros que
88      %caracterizan la ley de deformaciones)
89      [zfn,chi]=problFlexion.NAndM_zfnAndChi(p,ley_mat,N,M);
90
91      %Cálculo del diagrama de tensiones
92        if isnan(zfn)
93          %Caso de sección fuera de superficie de interacción
```

```
94          z_=NaN;
95          epsz_=NaN;
96          sigz_=NaN;
97        elseif zfn>=1E16
98          [z_,epsz_,sigz_]=problFlexion.EpszAndSigz_AxilPuro(p,ley_mat,N);
99        else
100         [z_,epsz_,sigz_]=problFlexion.zfnAndChi_EpszAndSigz(p,ley_mat,zfn,chi);
101       end
102
103       %Calcular superficies de plastificación
104       [Ap1,Ap2]=problFlexion.calcular_areasPlastificadas(p,z_);
105
106       %Especificar valores de salida
107       valSalida{1}=z_; %distancias medidas respecto posición de f.n.
108           %de los puntos característicos de la distribución de tensiones
109       valSalida{2}=sigz_; %valores de tensión de la distribución de
110                           %tensiones
111       valSalida{3}=epsz_; %valores de deformación de la distribución
112                           %de deformaciones
113       valSalida{4}=chi; %valor de la curvatura
114       valSalida{5}=Ap1; %polígono del área plastificada superior
115       valSalida{6}=Ap2; %polígono del área plastificada inferior
116
117       %Cambio de unidades a la salida
118       [valEntrada,valSalida]=acondicionamientoDatos.datosSalida(tipoProblema,
              valEntrada,valSalida);
119   otherwise
120     error('Tipo de problema no reconocido.')
121   end
122
123 end
```

### B.3.2. *generacion_Modelo*

```
1  classdef generacion_Modelo
2  methods(Static)
3
4  %*********************************************************************%
5  %                                                                     %
6  % generacion_Modelo: módulo que recoge las funciones para la         %
7  %                    generación del modelo                            %
8  %                                                                     %
9  %   verticesPoly(): función para la generación de la geometría       %
10 %   comportMaterial(): función para generar la ley de constitutiva   %
11 %                      del material                                   %
```

```
12 %                                                                      %
13 %**********************************************************************%
14
15
16 function ley_mat=comportMaterial(tipo,valores)
17 %-------------------------------------------------------------------------+
18 %
19 % ley_mat=comportMaterial(tipo,valores)
20 %
21 % Descripción función:
22 %   Generación de la ley constitutiva del material a partir del tipo
23 % de modelo y sus valores característicos del modelo
24 %
25 % Inputs:
26 %   tipo: tipo de modelo de material
27 %         <'G1' | 'G2' | 'G3' | 'G4'>
28 %   valores: valores característicos del modelo material(es función
29 %            del tipo)
30 %
31 % Outputs:
32 %   ley_mat: ley constitutiva del material. Vector fila i, [eps_i,sig_i]
33 %
34 %-------------------------------------------------------------------------+
35
36   switch tipo
37
38     case 'G1'
39     %G1: elástico - material perfectamente plástico con igual
40     %comportamiento a tracción que a compresión
41     sig_p=valores(1); %tensión en el límite de fluencia
42     E=valores(2); %módulo de Young
43
44     e_inf=100; %valor suficientemente grande
45     e_p=sig_p/E; %deformación unitaria en el límite elástico
46     ley_mat=[[-e_inf,-sig_p];[-e_p,-sig_p];[0,0];[e_p,sig_p];[e_inf,sig_p]];
47
48     case 'G2'
49     %G2: material elástico - perfectamente plástico con diferente
50     %comportamiento a tracción que compresión
51     E=valores(1); %módulo de Young
52     sig_pc=valores(2); %tensión del límite elástico a tracción
53     sig_pt=valores(3); %tensión del límite elástico a compresión
54
55     e_inf=100; %valor lo suficientemente grande
56     e_pt=sig_pt/E; %deformación unitaria en el límite elástico a tracc.
```

```
57      e_pc=sig_pc/E; %deformaión unitaria en el límite elástico a compr.
58      ley_mat=[[-e_inf,sig_pc];[e_pc,sig_pc];[0,0];...
59               [e_pt,sig_pt];[e_inf,sig_pt]];
60
61      case 'G3'
62      %G3: material con endurecimiento con igual comportamiento a
63      %tracción que a compresión
64      e_inf=100; %valor lo suficientemente grande
65
66      eps_e=valores(1); %def. unitaria en el límite elástico
67      sig_e=valores(2); %tensión en el límite elástico
68      eps_p1=valores(3); %def. unitaria en el primer límite de fluencia
69      sig_p1=valores(4); %tensión en el primer límite de fluencia
70      eps_p2=valores(5); %def. unitaria en el segundo límite de fluencia
71      sig_p2=valores(6); %tensión en el 2ndo límite de fluencia
72
73      eps_inf=100;
74      ley_mat=[[-e_inf,-sig_p2];[-eps_p2,-sig_p2];[-eps_p1,-sig_p1]; ...
75               [-eps_e,-sig_e];[0,0];[eps_e,sig_e];[eps_p1,sig_p1];...
76               [eps_p2,sig_p2];[eps_inf,sig_p2]];
77
78      case 'G4'
79      %G4: material con endurecimiento con diferente comportamiento a
80      %tracción que a compresión (caso más general)
81      e_inf=100; %valor lo suficientemente grande
82
83      E=valores(1);
84      sig_ec=valores(2);
85      eps_p1c=valores(3);
86      sig_p1c=valores(4);
87      eps_p2c=valores(5);
88      sig_p2c=valores(6);
89      sig_et=valores(7);
90      eps_p1t=valores(8);
91      sig_p1t=valores(9);
92      eps_p2t=valores(10);
93      sig_p2t=valores(11);
94
95      eps_ec=sig_ec/E;
96      eps_et=sig_et/E;
97
98      ley_mat=[[-e_inf,sig_p2c];[eps_p2c,sig_p2c];[eps_p1c,sig_p1c]; ...
99               [eps_ec,sig_ec];[0,0];[eps_et,sig_et];[eps_p1t,sig_p1t];...
100              [eps_p2t,sig_p2t];[e_inf,sig_p2t]];
101     otherwise
```

```
102         error(['Por favor, introduza un valor de material', ...
103                ' adecuado <G1 | G2 | G3 | G4>'])
104     end
105 end
106 %=========================================================================
107
108
109 function ptos=verticesPoly(tipo,dim)
110 %-------------------------------------------------------------------------+
111 %
112 % ptos=verticesPoly(tipo,dim)
113 %
114 % Descripción función:
115 %   Función que determina los puntos de una sección indicando el tipo y
116 % sus dimensiones
117 %
118 % Inputs:
119 %   tipo: tipo de dimensión
120 %     'R': sección rectangular
121 %     'C': sección circular
122 %     'I': sección doble T
123 %     'T': sección T
124 %     'Inb': sección I no bisimétrica
125 %     'Rh': sección rectangular con hueco
126 %     'Ch': sección tubular
127 %
128 %   dim: dimensiones de la sección(mm)
129 %       Sección R: [b,h] (b: base, h: altura)
130 %       Sección C: [r] (r: radio)
131 %       Sección I: [h,bf,tf,tw] (h: altura total, bf: ancho ala,
132 %                                tf: espesor ala, tw: espesor alma)
133 %       Sección T: [h,bf,tf,tw] (h: altura total, bf: ancho ala,
134 %                                tf: espesor ala, tw: espesor alma)
135 %       Sección Inb: [h,bf1,tf1,tw,bf2,tf2] (h:altura total,
136 %                    bf1:ancho ala superior, tf1: espesor ala superior,
137 %                    tw: espesor alma, bf2: ancho ala inferior,
138 %                    tf2: espesor ala inferior)
139 %       Sección Rh: [b,h,e] (b:base, h: altura, e:espesor)
140 %       Sección Ch: [r,e] (r:radio, e: espesor)
141 %
142 % Dependencias:
143 %   Ch_ptos.m, Inb_ptos.m, I_ptos.m,Rh_ptos.m, T_ptos.m, C_ptos.m,
144 %   IPE_ptos.m, Rombo_ptos.m, H_ptos.m, IPN_ptos.m, R_ptos.m
145 %
146 %-------------------------------------------------------------------------+
```

```
147   switch tipo
148     case 'R'
149       b=dim(1); %base
150       h=dim(2); %altura
151       ptos=[[0,0];[b,0];[b,h];[0,h]];
152
153     case 'I'
154       h=dim(1); %altura total del perfil
155       bf=dim(2); %ancho alas
156       tf=dim(3); %espesor alas
157       tw=dim(4); %espesor alma
158       ptos=[[0,0];[bf,0];[bf,tf];[(bf+tw)/2,tf];[(bf+tw)/2,h-tf];...
159             [bf,h-tf];[bf,h];[0,h];[0,h-tf];[(bf-tw)/2,h-tf];...
160             [(bf-tw)/2,tf];[0,tf]];
161     case 'C'
162       r=dim(1); %radio
163       n=100; %por defecto num. ptos n=100;
164       tita=linspace(3*pi/2,2*pi+3*pi/2,n);
165       ptos=([r*(cos(tita)+1);r*(sin(tita)+1)])';
166
167     case 'T'
168       h=dim(1); %altura total
169       bf=dim(2); %ancho alas
170       tf=dim(3); %espesor alas
171       tw=dim(4); %espesor alma
172       hw=h-tf; %Var. auxiliar (altura alma)
173       ptos=[[(bf-tw)/2,0];[(bf+tw)/2,0];[(bf+tw)/2,hw];[bf,hw];[bf,h];[0,h];...
174             [0,hw];[(bf-tw)/2,hw]];
175
176     case 'Inb'
177       h=dim(1); %Altura total perfil
178       bf1=dim(2); %Ancho ala superior
179       tf1=dim(3); %Espesor ala superior
180       tw=dim(4); %Espesor alma
181       bf2=dim(5); %Ancho ala inferior
182       tf2=dim(6); %Espesor ala inferior
183       hw=h-tf1-tf2; %Var. auxiliar(altura alma)
184       bmax=max(bf1,bf2); %Var. auxiliar(ancho máx. de las alas)
185       ptos=[[(bmax-bf2)/2,0];[(bmax+bf2)/2,0];[(bmax+bf2)/2,tf2];...
186             [(bmax+tw)/2,tf2];[(bmax+tw)/2,h-tf1];[(bmax+bf1)/2,h-tf1];...
187             [(bmax+bf1)/2,h];[(bmax-bf1)/2,h];[(bmax-bf1)/2,h-tf1];...
188             [(bmax-tw)/2,h-tf1];[(bmax-tw)/2,tf2];[(bmax-bf2)/2,tf2]];
189
190     case 'Ch'
191       R=dim(1); %radio exterior
```

```
192         espesor=dim(2); % espesor
193         r=R-espesor; % radio interior
194         n=100; % por defecto num. ptos n=100
195         % lím. sup. de tita~=lim.inf. de tita
196         tita=linspace(0+6*pi/4,6*pi/4+2*pi,n);
197         ptos_r=([r*cos(tita)+R;r*sin(tita)+R])';
198         ptos_r=ptos_r(length(ptos_r):-1:1,:); %s.a.r. ya que es un agujero
199         ptos_R=([R*(cos(tita)+1);R*(sin(tita)+1)])';
200         ptos=[ptos_R;ptos_r];
201
202       case 'Rh'
203         b=dim(1); % ancho (exterior)
204         h=dim(2); % canto (exterior)
205         e=dim(3); % espesor
206         p_ext=[[e,0];[b,0];[b,h];[0,h];[0,0];[e,0]];
207         p_int=[[e,e];[e,h-e];[b-e,h-e];[b-e,e];[e,e]];
208         ptos=[p_ext;p_int];
209
210       case 'H'
211         hf=dim(1); % altura ala
212         b=dim(2); % ancho total
213         tf=dim(3); % espesor ala
214         tw=dim(4); % espesor alma
215         bw=b-2*tf; %Var. aux.(ancho del alma)
216
217         ptos=[[0,0];[tf,0];[tf,(hf-tw)/2];[b-tf,(hf-tw)/2];[b-tf,0];[b,0];[b,hf
                ];...
218             [b-tf,hf];[b-tf,(hf+tw)/2];[tf,(hf+tw)/2];[tf,hf];[0,hf]];
219
220       case 'Rombo'
221         d1=dim(1); % diagonal horizontal
222         d2=dim(2); % diagonal vertical
223
224         ptos=[[d1/2,0];[d1,d2/2];[d1/2,d2];[0,d2/2]];
225
226       case 'IPN'
227         h=dim(1); % altura total
228         b=dim(2); % ancho alas
229         r1=dim(3); % tw=r1 (espesor alma=radio entre alma-ala)
230         tf=dim(4); % espesor alas
231         r2=dim(5); % radio - ala
232         d=dim(6); % altura alma
233
234         ntita=5; % chaflán
235         %Chaflán A
```

```
236         titaA=linspace(0,pi/2,ntita);
237         CA=[b-r2,r2];
238         rA=r2;
239         pA=([[rA*cos(titaA)+CA(1)];[rA*sin(titaA)+CA(2)]])';
240
241         %Chaflán B
242         titaB=linspace(-pi/2,-pi,ntita);
243         yB=(h-d)/2;
244         xB=(b+r1)/2+r1;
245         CB=[xB,yB];
246         rB=r1;
247         pB=([[rB*cos(titaB)+CB(1)];[rB*sin(titaB)+CB(2)]])';
248
249         %Chaflán C
250         titaC=linspace(-pi,-3*pi/2,ntita);
251         xC=xB;
252         yC=(h+d)/2;
253         CC=[xC,yC];
254         rC=r1;
255
256         pC=([[rC*cos(titaC)+CC(1)];[rC*sin(titaC)+CC(2)]])';
257
258         %Chaflán D
259         titaD=linspace(3*pi/2,2*pi,ntita);
260         CD=[b-r2,h-r2];
261         rD=r2;
262         pD=([[rD*cos(titaD)+CD(1)];[rD*sin(titaD)+CD(2)]])';
263
264         %Chaflán E
265         titaE=linspace(pi,3*pi/2,ntita);
266         CE=[r2,h-r2];
267         rE=r2;
268         pE=([[rE*cos(titaE)+CE(1)];[rE*sin(titaE)+CE(2)]])';
269
270         %Chaflán F
271         titaF=linspace(-3*pi/2,-2*pi,ntita);
272         CF=[(b-r1)/2-r1,yC];
273         rF=r1;
274         pF=([[rF*cos(titaF)+CF(1)];[rF*sin(titaF)+CF(2)]])';
275
276         %Chaflán G
277         titaG=linspace(0,-pi/2,ntita);
278         yG=CB(2); CG=[CF(1),yG]; rG=r1;
279         pG=([[rG*cos(titaG)+CG(1)];[rG*sin(titaG)+CG(2)]])';
280
```

```
281         %Chaflán H
282         titaH=linspace(pi/2,pi,ntita);
283         CH=[r2,r2]; rH=r2;
284         pH=([[rH*cos(titaH)+CH(1)];[rH*sin(titaH)+CH(2)]])';
285
286         ptos=[[0,0];[b,0];pA;pB;pC;pD;[b,h];[0,h];...
287                 pE;pF;pG;pH];
288
289     case 'IPE'
290         h=dim(1); %altura total
291         b=dim(2); %ancho alas
292         tw=dim(3); %espesor alma
293         tf=dim(4); %espesor alas
294         r=dim(5); %radio
295         d=dim(6); %altura alma
296
297         ntita=5;
298         %Chaflán B
299         titaB=linspace(-pi/2,-pi,ntita);
300         yB=tf+r;
301         xB=(b+tw)/2+r;
302         CB=[xB,yB];
303         rB=r;
304         pB=([[rB*cos(titaB)+CB(1)];[rB*sin(titaB)+CB(2)]])';
305
306         %Chaflán C
307         titaC=linspace(-pi,-3*pi/2,ntita);
308         xC=xB;
309         yC=h-tf-r;
310         CC=[xC,yC];
311         rC=r;
312         pC=([[rC*cos(titaC)+CC(1)];[rC*sin(titaC)+CC(2)]])';
313
314
315         %Chaflán F
316         titaF=linspace(-3*pi/2,-2*pi,ntita);
317         xF=(b-tw)/2-r;
318         yF=yC;
319         CF=[xF,yF];
320         rF=r;
321         pF=([[rF*cos(titaF)+CF(1)];[rF*sin(titaF)+CF(2)]])';
322
323
324         %Chaflán G
325         titaG=linspace(0,-pi/2,ntita);
```

```
326         xG=xF;
327         yG=yB;
328         CG=[xG,yG];
329         rG=r;
330         pG=([[rG*cos(titaG)+CG(1)];[rG*sin(titaG)+CG(2)]])';
331
332         ptos=[[0,0];[b,0];[b,tf];pB;pC;[b,h-tf];...
333               [b,h];[0,h];[0,h-tf];pF;pG;[0,tf]];
334
335       case 'cajon'
336         %Definición de vértices
337         b=dim(1); %ancho agujero
338         b1=dim(2); %saliente ala
339         tf1=dim(3); %espesor ala superior
340         hw=dim(4); %altura almas
341         tw=dim(5); %ancho almas
342         tf2=dim(6); %espesor ala inferior
343
344         h=hw+tf1+tf2;
345         b2=b1;
346
347         v1=[0,0]; v2=[0,tf2]; v3=[0,h-tf1];v4=[b-2*tw,h-tf1];
348         v5=[b-2*tw,tf2]; v6=[b-tw+b2,0]; v7=[b-tw+b2,tf2];
349         v8=[b-tw,tf2]; v9=[b-tw,h-tf1]; v10=[b-tw+b1,h-tf1];
350         v11=[b-tw+b1,h]; v12=[-b1-tw,h]; v13=[-b1-tw,h-tf1];
351         v14=[-tw,h-tf1]; v15=[-tw,tf2]; v16=[-tw-b2,tf2];
352         v17=[-tw-b2,0];
353
354         ptos=[v1;v2;v3;v4;v5;v2;v1;v6;v7;v8;v9;v10;v11;v12;v13;v14;v15;v16;v17];
355
356       otherwise
357         error(['Por favor introduce un tipo de perfil apropiado.',...
358                'Opciones: R, I, H, C, T, Inb, Ch, Rh, IPE, IPN, cajon'])
359     end
360 end
361 %
362 %=========================================================================
363
364 end
365 end
```

### B.3.3. *acondicionamientoDatos*

```
1 classdef acondicionamientoDatos
2 methods(Static)
```

```
3
4  %*****************************************************************%
5  %                                                                 %
6  % acondicionamientoDatos: módulo que contiene las funciones para  %
7  %   preparar los datos a la entrada de la resolución del problema %
8  %   y para la presentación de los resultados                      %
9  %                                                                 %
10 % Funciones que contiene:                                        %
11 %                                                                 %
12 %   datosEntrada(): función que prepara los datos a la entrada de los %
13 % cálculos                                                       %
14 %   datosSalida(): función que prepara los datos a la salida de la %
15 % resolución del problema para su posterior representación       %
16 %                                                                 %
17 %*****************************************************************%
18
19
20 function valEntrada=datosEntrada(tipo_problema,valEntrada)
21 %-----------------------------------------------------------------+
22 % Acondicionamiento de los datos a la entrada de los cálculos:
23 % Dimensiones: mm --> m
24 % Tensiones: MPa --> kPa
25 % Origen del sistema de referencia de los vértices de la sección: c.d.g
26 %-----------------------------------------------------------------+
27   p=valEntrada{1};
28   yG=propSeccion.propGeom(p,'yG');
29   zG=propSeccion.propGeom(p,'zG');
30   p=problGeom.cambioSistCoord(p,-yG,-zG,0);
31   p=p*1E-3; %mm-->m
32   valEntrada{1}=p;
33
34   ley_mat=valEntrada{2};
35   ley_mat(:,2)=ley_mat(:,2)*1E3; %MPa --> kPa
36   valEntrada{2}=ley_mat;
37
38 end
39 %
40 %=================================================================
41
42
43
44 function [valEntrada,valSalida]=datosSalida(tipo_problema,valEntrada,valSalida)
45 %-----------------------------------------------------------------+
46 % Acondicionamiento de los datos a la salida de los cálculos:
47 % Dimensiones: m --> mm
```

```
48  %  Tensiones: kPa --> MPa
49  %-------------------------------------------------------------------------+
50
51    valEntrada{1}=valEntrada{1}*1E3;  %m --> mm
52    ley_mat=valEntrada{2};
53    ley_mat(:,2)=ley_mat(:,2)*1E-3;  %kPa --> MPa
54    valEntrada{2}=ley_mat;
55
56    switch tipo_problema
57
58    case 1
59      d_MC=valSalida{7};
60      d_MC(:,3:5)=d_MC(:,3:5)*1E3;  %m --> mm
61       valSalida{7}=d_MC;
62
63    case 2
64      d_int=valSalida{1};
65      d_int(:,3)=d_int(:,3)*1E3;  %m --> mm
66      valSalida{1}=d_int;
67
68    case 3
69      valSalida{1}=valSalida{1}*1E3;  %m --> mm
70      valSalida{2}=valSalida{2}*1E-3;  %kPa --> MPa
71      if iscell(valSalida{5})
72        for i=1:length(valSalida{5})
73          valSalida{5}{i}=valSalida{5}{i}*1E3;  %m --> mm
74        end
75      else
76        valSalida{5}=valSalida{5}*1E3;  %m --> mm
77      end
78
79      if iscell(valSalida{6})
80        for i=1:length(valSalida{6})
81          valSalida{6}{i}=valSalida{6}{i}*1E3;  %m --> mm
82        end
83      else
84        valSalida{6}=valSalida{6}*1E3;  %m --> mm
85      end
86
87    otherwise
88      error('No se reconoce tipo de problema')
89    end
90  end
91  %
92  %=========================================================================
```

```
93
94 end
95 end
```

### B.3.4. *problFlexion*

```
1  classdef problFlexion
2  methods(Static)
3
4  %*********************************************************************%
5  %                                                                     %
6  % problFlexion: módulo que recoge las funciones para la resolución de %
7  %               problemas de flexión en régimen elasto-plástico       %
8  %   zfnAndChi_N(): cálculo del axil a partir de la posición de la     %
9  %                  de la fibra neutra y la curvatura                  %
10 %   zfnAndChi_M(): cálculo del momento a partir de la posición de la  %
11 %                  fibra neutra, curvatura y axil                     %
12 %   sigma(): cálculo de la tensión de una fibra dada su deformación   %
13 %            unitaria                                                 %
14 %   zfnAndChi_EpszAndSigz(): cálculo de la ley de tensiones y         %
15 %                            deformaciones a partir de la posición    %
16 %                            de la f.n. y curvatura                   %
17 %   EpszAndSigz_AxilPuro(): cálculo de la ley de tensiones y          %
18 %                           deformaciones para axil puro              %
19 %   calculo_factorForma(): cálculo del factor de forma de una sección %
20 %   N_Me(): cálculo del momento elástico asociado a un valor de axil  %
21 %   N_Mp(): cálculo del momento plástico asociado a un valor de axil  %
22 %   N_diagramaMC(): diagrama Momento-Curvatura para un axil determinado%
23 %   N_diagramaMC2():diagrama Momento-Curvatura para un axil determinado%
24 %   diagrama_interaccionNM(): diagrama interacción (N-M)              %
25 %   zfnp_NpMp(): cálculo del par de agotamiento de la sección (Np,Mp) %
26 %                asociado a una posición de fibra neutra (zfnp)       %
27 %                                                                     %
28 %   zpAndN_momento(): cálculo del momento a partir de la profundidad  %
29 %                     de plastificación y valor del axil              %
30 %   zpAndN_zfn(): cálculo de la posición de la fibra neutra a partir  %
31 %                 de la profundidad de plastificación y axil          %
32 %   zpAndzfn_N(): cálculo del axil a partir de la profundidad de      %
33 %                 plastificación y la posición de la fibra neutra     %
34 %   chiAndN_momento(): cálculo del momento a partir de la curvatura   %
35 %                      y valor del axil                               %
36 %   chiAndN_zfn(): cálculo de la posición de la fibra neutra a partir %
37 %                  de la curvatura y el valor del axil                %
38 %   NAndM_zfnAndChi(): cálculo de la posición de la fibra neutra y la %
39 %                      curvatura a partir de la solicitación (N,M)    %
```

```
40 %    NM_NeMe(): calcular el momento-axil máximo que se encuentra en      %
41 %                régimen elástico dada una recta de carga determinada    %
42 %                por N y M                                               %
43 %    NM_NpMp(): calcular el momento-axil de agotamiento dada una recta   %
44 %                de carga determinada por N y M                          %
45 %    calcular_areasPlastificadas(): determinar los polígonos que         %
46 %                encierran las regiones plastificadas de la sección      %
47 %                                                                        %
48 %************************************************************************%
49
50
51 function N=zfnAndChi_N(P,ley_mat,zfn,chi)
52 %+----------------------------------------------------------------------+
53 %
54 % N=zfnAndChi_N(P,ley_mat,zfn,chi)
55 %
56 % Descripción función:
57 %   Calcular el axil dada la posición de la fibra neutra y la curvatura
58 % de la distribución de tensiones
59 %
60 % Inputs:
61 %   P: conjunto vértices que determinan el polígono que encierra a la
62 %      sección. Siendo el vector fila i: [yi,zi]
63 %   ley_mat: ley constitutiva del material.
64 %            Siendo el vector fila i [eps_i,sig_i]
65 %   zfn: posición de la fibra neutra respecto sist. ref. global(OYZ)
66 %   chi: curvatura
67 %
68 % Outputs:
69 %   N: axil que es estáticamente equivalente al diagrama de tensiones
70 %     con zfn y chi.
71 %
72 % Dependencias:
73 %   problFlexion.zfnAndChi_EpszAndSigz, problGeom.cambioSistCoord,
74 % problGeom.calculo_cortarArea, propSeccion.propGeom
75 %
76 %+----------------------------------------------------------------------+
77   z1=max(P(:,2));
78   z2=min(P(:,1));
79
80   %Calcular la distribución de tensiones y deformaciones
81   [z_,epsz_,sigz_]=problFlexion.zfnAndChi_EpszAndSigz(P,ley_mat,zfn,chi);
82
83   nz_=length(z_); %z_ está ordenado de menor a mayor
84
```

```
85    %Referimos los puntos de la sección al sistema de referencia O_Y_Z_
86    %O_Y_Z_ sist. referencia local, donde O_=(0,zfn)
87    P_=problGeom.cambioSistCoord(P,0,-zfn,0);
88
89    %Cálculo del axil(N)
90    N=0; %inicializamos variable
91    for i=1:nz_-1
92       %Para el tramo i, zi_<=zx<=zj_ --> epsi_<=epsx<=epsj_ -->
93       %sigi_<=sigx<=sigj_
94       Pi_=problGeom.calculo_cortarArea(P_,[0,z_(i+1),1],2);
95       P_=problGeom.calculo_cortarArea(P_,[0,z_(i+1),1],1);
96       Ai_=propSeccion.propGeom(Pi_,'area');
97       Wyi_=propSeccion.propGeom(Pi_,'Wy');
98       if length(Pi_)==0, Ai_=0;Wyi_=0;end
99       Ex=(sigz_(i+1)-sigz_(i))/(epsz_(i+1)-epsz_(i));
100      Nx=(sigz_(i)-Ex*epsz_(i))*Ai_-Ex*chi*Wyi_;
101      N=N+Nx;
102   end
103 end
104 %=========================================================================
105
106
107 function M=zfnAndChi_M(P,ley_mat,zfn,chi,N)
108 %-------------------------------------------------------------------------+
109 %
110 % M=zfnAndChi_M(P,ley_mat,zfn,chi,N)
111 %
112 % Descripción función:
113 %   Calcular el momento estáticamente equivalente a la distribución
114 % de tensiones, dada la posición de la fibra neutra y la curvatura
115 %
116 % Inputs:
117 %   P: conjunto vértices que determinan el polígono que encierra a la
118 %      sección. Siendo el vector fila i: [yi,zi]
119 %   ley_mat: ley constitutiva del material.
120 %            Siendo el vector fila i [eps_i,sig_i]
121 %   zfn: posición de la fibra neutra respecto sist. ref. global(OYZ)
122 %   chi: curvatura
123 %   N: axil(si es flexión pura, N=0)
124 %
125 % Outputs:
126 %   M: momento que es estáticamente equivalente al diagrama de tensiones
127 %      con zfn y chi.
128 %
129 % Dependencias:
```

```
%    problFlexion.zfnAndChi_EpszAndSigz, problGeom.cambioSistCoord,
% propSeccion.propGeom, problGeom.calculo_cortarArea
%
%-------------------------------------------------------------------------+

  %Cálculo de la distribución de tensiones y deformaciones
  [z_,epsz_,sigz_]=problFlexion.zfnAndChi_EpszAndSigz(P,ley_mat,zfn,chi);

  nz_=length(z_);
  %z_ está ordenado de menor a mayor

  %Referimos los puntos de la sección al sistema de referencia O_Y_Z_
  %O_Y_Z_ sist. referencia local, donde O_=(0,zfn)
  P_=problGeom.cambioSistCoord(P,0,-zfn,0);
  dist=propSeccion.propGeom(P_,'zG'); %Distancia del cdg a la f.n.

  %Cálculo del momento(M)
  M=0; %inicializamos variable
for i=1:nz_-1
    %Para el tramo i, zi_<=zx<=zj_ --> epsi_<=epsx<=epsj_ -->
    %sigi_<=sigx<=sigj_
    Pi_=problGeom.calculo_cortarArea(P_,[0,z_(i+1),1],2);
    P_=problGeom.calculo_cortarArea(P_,[0,z_(i+1),1],1);
    Wyi_=propSeccion.propGeom(Pi_,'Wy');
    Ex=(sigz_(i+1)-sigz_(i))/(epsz_(i+1)-epsz_(i));
    Wyi_=propSeccion.propGeom(Pi_,'Wy');
    Iyi_=propSeccion.propGeom(Pi_,'Iy');
    Ex=(sigz_(i+1)-sigz_(i))/(epsz_(i+1)-epsz_(i));
    Mx=-(sigz_(i)-Ex*epsz_(i))*Wyi_+Ex*chi*Iyi_;
    M=M+Mx;
  end
  M=M+dist*N;
end
%
%=========================================================================


function sig_x=sigma(e_x,ley_constitutiva)
%-------------------------------------------------------------------------+
%
% sig_x=sigma(e_x,ley_constitutiva)
%
% Descripción función:
%   Función que calcula la tensión de una fibra que tiene una
% deformación(e_x) y una ley constitutiva(ley_constitutiva)
```

```
175 %
176 % Inputs:
177 %   e_x: deformación de la fibra donde queremos calcular la tensión
178 %   ley_constitutiva: ley constitutiva del material
179 %
180 % Outputs:
181 %   sig_x: tensión que soporta la fibra en cuestión
182 %
183 % Variables Auxiliares:
184 %   E: módulo de Young en el tramo considerado de la ley constitutiva
185 %
186 % Dependencias:
187 %   Ninguna
188 %
189 %-------------------------------------------------------------------------+
190
191   i=1; %inicializamos variable
192
193   while e_x>ley_constitutiva(i,1)
194     i=i+1;
195   end
196
197   E=(ley_constitutiva(i,2)-ley_constitutiva(i-1,2))/...
198     (ley_constitutiva(i,1)-ley_constitutiva(i-1,1));
199   sig_x=ley_constitutiva(i-1,2)+E*(e_x-ley_constitutiva(i-1,1));
200 end
201 %=========================================================================
202
203
204 function [z_,epsz_,sigz_]=zfnAndChi_EpszAndSigz(P,ley_mat,zfn,chi)
205 %-------------------------------------------------------------------------+
206 %
207 % [z_,epsz_,sigz_]=zfnAndChi_EpszAndSigz(P,ley_mat,zfn,chi)
208 %
209 % Descripción función:
210 %   Función que calcula el diagrama de tensiones y deformaciones
211 % a partir de la posición de la fibra neutra y la curvatura.
212 %
213 % Inputs:
214 %   P: conjunto vértices que determinan el polígono que encierra a la
215 %      sección. Siendo el vector fila i: [yi,zi]
216 %   ley_mat: ley constitutiva del material.
217 %            Siendo el vector fila i [eps_i,sig_i]
218 %   zfn: posición de la f.n. respecto sistema de referencia global(OYZ)
219 %   chi: curvatura
```

```
220 %
221 % Outputs:
222 %   z_: posiciones referidas al sist. de referencia local(asociado
223 %       a la posición de la f.n., O_Y_Z_)
224 %   epsz_: deformaciones asociadas a las diferentes posiciones z_
225 %         calculadas mediante hipótesis Navier, eps(z_)=chi*z_
226 %   sigz_: tensiones asociadas a las diferentes posiciones z_
227 %
228 % Dependencias:
229 %   problGeom.cambioSistCoord, propSeccion.propMaterial,
230 % problFlexion.sigma
231 %
232 %-------------------------------------------------------------------------+
233
234   delta=1E-6;
235
236   P_=problGeom.cambioSistCoord(P,0,-zfn,0);
237
238   z1_=max(P_(:,2)); %posición cara sup. respecto f.n.
239   z2_=min(P_(:,2)); %posición cara sup. respecto f.n.
240
241   if abs(chi)>1E3
242
243     %Sección agotada
244     epsz_=[nan; -inf;inf; inf; nan];
245     chi=100;
246     z_=[z1_;-1E-16;0;1E-16;z2_];
247
248     epsz1_=sign(-chi*z1_)*50;
249     epsz2_=sign(-chi*z2_)*50;
250
251     sig_pc=propSeccion.propMaterial(ley_mat,'sig_pc');
252     sig_pt=propSeccion.propMaterial(ley_mat,'sig_pt');
253     if chi>0, sigz1_=sig_pc;sigz2_=sig_pt;
254     else, sigz1_=sig_pt;sigz2_=sig_pc; end
255     sigz_=[sigz1_;sigz1_;0;sigz2_;sigz2_];
256     return
257   end
258
259   %Cálculo de posiciones(z_)
260   z_=ley_mat(:,1)/(-chi);
261   z_=z_(find(z_>=z2_ & z_<=z1_));
262   z_=unique(z_);
263
264   if abs(min(z_)-z2_)<=delta, z_=z_(2:end); end
```

```
265     if abs(max(z_)-z1_)<=delta, z_=z_(1:end-1); end
266     z_=[z2_;z_;z1_];
267
268     z_=unique(z_); %Garantizar que no hay puntos repetidos
269
270     %Cálculo diagrama de deformaciones
271     epsz_=-chi*z_;
272
273     %Cálculo diagrama de tensiones
274     nz_=length(z_);
275     sigz_=nan(nz_,1);
276     for i=1:nz_
277       sigz_(i)=problFlexion.sigma(epsz_(i),ley_mat);
278     end
279
280 end
281 %
282 %=========================================================================
283
284
285 function [z_,epsz_,sigz_]=EpszAndSigz_AxilPuro(p,ley_mat,N)
286 %+-----------------------------------------------------------------------+
287 %
288 % [z_,epsz_,sigz_]=EpszAndSigz_AxilPuro(p, ley_mat, N)
289 %
290 % Descripción función: función que calcula el diagrama de tensiones
291 %   y deformaciones cuando la sección está sometida a axil puro
292 %
293 % Inputs:
294 %   p: conjunto vértices que determinan el polígono que encierra a la
295 %      sección. Siendo el vector fila i: [yi,zi]
296 %   ley_mat: ley constitutiva del material.
297 %            Siendo el vector fila i [eps_i,sig_i]
298 %   N: axil de la sección
299 %
300 % Outputs:
301 %   z_: posiciones referidas al sist. de referencia local(asociado
302 %       a la posición de la f.n., O_Y_Z_)
303 %   epsz_: deformaciones asociadas a las diferentes posiciones z_
304 %         calculadas mediante hipótesis Navier, eps(z_)=chi*z_
305 %   sigz_: tensiones asociadas a las diferentes posiciones z_
306 %
307 % Dependencias:
308 %   propSeccion.propMaterial, propSeccion.propGeom
309 %
```

```
310 %-------------------------------------------------------------------------+
311
312   E=propSeccion.propMaterial(ley_mat,'E');
313   Atot=propSeccion.propGeom(p,'area');
314
315   z_=[min(p(:,2)),max(p(:,2))]';
316   sigma=N/Atot;
317   epsilon=sigma/E;
318
319   epsz_=[epsilon,epsilon]';
320   sigz_=[sigma,sigma]';
321
322 end
323 %=========================================================================
324
325
326 function ff=calculo_factorForma(p,ley_mat,signoM)
327 %-------------------------------------------------------------------------+
328 %
329 %  ff=calculo_factorForma(p,ley_mat,signoM)
330 %
331 %  Descripción función:
332 %    Función que calcula el factor de forma de una sección
333 %
334 %  Inputs:
335 %    p: conjunto vértices que determinan el polígono que encierra a la
336 %       sección. Siendo el vector fila i: [yi,zi]
337 %    ley_mat: ley constitutiva del material.
338 %             Siendo el vector fila i [eps_i,sig_i]
339 %    signoM: signo del momento <'p': positivo | 'n': negativo>
340 %      (Momento positivo: cara superior comprimida)
341 %
342 %  Outputs:
343 %    ff: factor de forma
344 %
345 %  Se asume:
346 %    M>0: cara superior comprimida
347 %    M<0: cara superior traccionada
348 %
349 %  Dependencias:
350 %    problFlexion.N_Me, problFlexion.N_Mp
351 %
352 %-------------------------------------------------------------------------+
353   N=0;
354   Me=problFlexion.N_Me(p,ley_mat,N,signoM);
```

```
355   Mp=problFlexion.N_Mp(p,ley_mat,N,signoM);
356   ff=Mp/Me;
357 end
358 %
359 %=========================================================================
360
361
362 function   [Me,zfne,chie,condic]=N_Me(p,ley_mat,N,signoM)
363 %-------------------------------------------------------------------------+
364 %
365 %  [Me,zfne,chie,condic]=N_Me(p,ley_mat,N,signoM)
366 %
367 %  Descripción función:
368 %    Función que calcula el momento elástico asociado a un valor de axil
369 %
370 %  Inputs:
371 %    p: conjunto vértices que determinan el polígono que encierra a la
372 %       sección. Siendo el vector fila i: [yi,zi]
373 %    ley_mat: ley constitutiva del material.
374 %             Siendo el vector fila i [eps_i,sig_i]
375 %    signoM: signo del momento <'p': positivo | 'n': negativo>
376 %      (Momento positivo: cara superior comprimida)
377 %
378 %  Outputs:
379 %    Me: máximo momento asociado a un valor de axil, con el que la
380 %        sección permanece en el régimen elástico
381 %    zfne: posición de la fibra neutra cuando la sección está sometida
382 %          por la solicitación (N,Me)
383 %    chie: curvatura de la sección cuando está sometida a (N,Me)
384 %    condic: indica qué cabeza de la sección plastifica primero
385 %            <1: si cabeza superior | 2: si cabeza inferior>
386 %  Se asume:
387 %    M>0: cara superior comprimida
388 %    M<0: cara superior traccionada
389 %
390 %  Dependencias:
391 %    propSeccion.propGeom, propSeccion.propMaterial
392 %
393 %-------------------------------------------------------------------------+
394
395   zG=propSeccion.propGeom(p,'zG');
396   Iy=propSeccion.propGeom(p,'IGy');
397   Atot=propSeccion.propGeom(p,'area');
398   z1=max(p(:,2));
399   z2=min(p(:,2));
```

```
400
401   z1_=z1-zG; %Referimos posición de la cara superior a la
402                        %posición de la f.n.
403   z2_=z2-zG; %Referimos posición de la cara inferior a la
404                        %posición de la f.n.
405
406   %Propiedades del material
407   sig_et=propSeccion.propMaterial(ley_mat,'sig_et');
408   sig_ec=propSeccion.propMaterial(ley_mat,'sig_ec');
409   E=propSeccion.propMaterial(ley_mat,'E');
410
411   if N<0, sig_e=sig_ec; else sig_e=sig_et;end
412   Ny=sig_e*Atot;
413   sigN=N/Atot;
414   if abs(N)>=abs(Ny) && N~=0
415     Me=0;
416     z1=1E16;z2=-1E16;
417     if signoM=='p'
418       if N>0 condic=1; zfne=z1;else, condic=2; zfne=z2;end
419     else
420       if N>0 condic=2; zfne=z2; else, condic=1; zfne=z1;end
421     end
422     chie=0;
423   else
424   if signoM=='p'
425     %Caso en el que el momento es positivo
426     sigp1=sig_ec;
427     sigp2=sig_et;
428   else
429     %Caso en el que el momento es negativo
430     sigp1=sig_et;
431     sigp2=sig_ec;
432   end
433
434   %Cálculo Me
435   if abs((sigp1-sigN)/z1_)<abs((sigp2-sigN)/z2_)
436     condic=1;
437     Me=-(sigp1-sigN)*Iy/z1_;
438   else
439     condic=2;
440     Me=-(sigp2-sigN)*Iy/z2_;
441   end
442
443   sig1=sigN-Me/Iy*z1_;
444   sig2=sigN-Me/Iy*z2_;
```

```
445
446     %Cálculo de la posición de la fibra neutra y curvatura
447     zfne=(z1*sig2-z2*sig1)/(sig2-sig1);
448     chie=Me/(E*Iy);
449    end
450 end
451 %=========================================================================
452
453
454 function [Mp,zfnp,zpmax,signo_zpmax,condic]=N_Mp(p,ley_mat,N,signoM)
455 %-------------------------------------------------------------------------+
456 %
457 % [Mp,zfnp,zpmax,signo_zpmax,condic]=N_Mp(p,ley_mat,N,signoM)
458 %
459 % Descripción función:                                                   %
460 %   Función que calcula el momento plástico asociado a un
461 % axil de una sección
462 %
463 % Inputs:
464 %   p: conjunto vértices que determinan el polígono que encierra a la
465 %      sección. Siendo el vector fila i: [yi,zi]
466 %   ley_mat: ley constitutiva del material.
467 %            Siendo el vector fila i [eps_i,sig_i]
468 %   signoM: signo del momento <'p': positivo | 'n': negativo>
469 %
470 % Outputs:
471 %   Mp: momento plástico de la sección p asociado al axil N
472 %   zfnp: posición de la fibra neutra cuando solicitación es: (N,Mp)
473 %   zpmax: valor de la profundidad de plastificación de la sección
474 %          medida desde el extrem ue plastifica primero cuando
475 %          solicitación es (N,Mp)
476 %   signo_zpmax: carácter de la profundidad de plastificación (zpmax)
477 %                <'t':tracción | 'c':compresión >
478 %   condic: indica ué extremo(superior/inferior) plastifica primero
479 %           <1: extremo superior | 2: extremo inferior >
480 %
481 % Se asume:
482 %   M>0: cara superior comprimida
483 %   M<0: cara superior traccionada
484 %
485 % Dependencias:
486 %   problFlexion.N_Me, propSeccion.propMaterial, problFlexion.zfnp_NpMp,
487 % metNum.zero, problGeom.calculo_cambioSistCoord,
488 % problGeom.calculo_cortarArea, propSeccion.propGeom
489 %
```

```
490 %-------------------------------------------------------------------------+
491
492     tolerancia=1E-6;
493
494     z1=max(p(:,2));
495     z2=min(p(:,2));
496
497     [~,zfne,~,condic]=problFlexion.N_Me(p,ley_mat,N,signoM); %Saber qué cara
            plastifica antes
498
499     sig_pt=propSeccion.propMaterial(ley_mat,'sig_pt');
500     sig_pc=propSeccion.propMaterial(ley_mat,'sig_pc');
501
502     if signoM=='p'
503       %Caso en el que el momento es positivo
504       sigp1=sig_pc;
505       sigp2=sig_pt;
506     else
507       %Caso en el que el momento es negativo
508       sigp1=sig_pt;
509       sigp2=sig_pc;
510     end
511
512     f=@(zfnp) N-problFlexion.zfnp_NpMp(p,ley_mat,zfnp,signoM);
513
514     zfnp1=z2;
515     zfnp2=z1;
516
517     %Cálculo de la posición de la f.n. cuando la sección está agotada
518     zfnp=metNum.zero(zfnp1,zfnp2,eps,tolerancia,f);
519
520     %Cálculo Mp
521     p_=problGeom.calculo_cambioSistCoord(p,0,-zfnp,0);
522     p1_=problGeom.calculo_cortarArea(p_,[0,0,1],1);
523     W1_=propSeccion.propGeom(p1_,'Wy');
524     p2_=problGeom.calculo_cortarArea(p_,[0,0,1],2);
525     W2_=propSeccion.propGeom(p2_,'Wy');
526     dist=propSeccion.propGeom(p_,'zG');
527     Mp=-sigp1*W1_-sigp2*W2_+dist*N;
528
529     if condic==1
530       zpmax=z1-zfnp;
531       if sigp1<0, signo_zpmax='c';
532       else, signo_zpmax='t'; end
533     else
```

```
534       zpmax=zfnp-z2;
535       if sigp2<0, signo_zpmax='c';
536       else, signo_zpmax='t'; end
537     end
538   end
539   %
540   %=========================================================================
541
542
543   function [D_MC,Me,chie,Mp]=N_diagramaMC(p,ley_mat,N,signoM)
544   %-------------------------------------------------------------------------+
545   %
546   % [D_MC,Me,chie,Mp]=N_diagramaMC(p,ley_mat,N,signoM)
547   %
548   % Descripción función:
549   %   Función que calcula el diagrama Momento-Curvatura de una sección
550   % con el mínimo número de puntos
551   %
552   % Inputs:
553   %   p: conjunto vértices que determinan el polígono que encierra a la
554   %      sección. Siendo el vector fila i: [yi,zi]
555   %   ley_mat: ley constitutiva del material.
556   %            Siendo el vector fila i [eps_i,sig_i]
557   %   signoM: signo del momento <'p': positivo | 'n': negativo>
558   %     (Momento positivo: cara superior comprimida)
559   %
560   % Outputs:
561   %   D_MC: diagrama momento curvatura. Array
562   %     D_MC(:,1): curvatura
563   %     D_MC(:,2): momento
564   %     D_MC(:,3): profundidad de plastificación
565   %     D_MC(:,4): posición fibra neutra
566   %     D_MC(:,5): posición primera fibra plastificada respecto
567   %                posición f.n.
568   %   Me: momento elástico
569   %   chie: curvatura cuando el momento es elástico
570   %   Mp: momento plástico
571   %
572   % Se asume:
573   %   M>0: cara superior comprimida
574   %   M<0: cara superior traccionada
575   %
576   % Limitación:
577   %   Esta función no es válida para material con endurecimiento con
578   % valores de axil próximos a Ny. Para estos casos utilizar
```

```
% N_diaramaMC2()
%
% Dependencias:
%   problFlexion.N_Mp, problFlexion.N_Me, problFlexion.zpAndN_momento
%
%-------------------------------------------------------------------------+

% delta=1E-3;
  n=20; %Número de puntos del diagrama

  [Mp,zfnp,zpmax,signo_zpmax,condic]=problFlexion.N_Mp(p,ley_mat,N,signoM);
  [Me,zfne,chie]=problFlexion.N_Me(p,ley_mat,N,signoM);

  zp=linspace(0,zpmax*0.95,n);

  signo_zp=signo_zpmax;

 D_MC=[];
  for i=1:length(zp)
    [M,chi,zfn,ze]=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,zp(i),
        signo_zp,condic,N);
    D_MC=[D_MC;[chi,M,zp(i),zfn,ze]];
  end
end
%=========================================================================


function [D_MC,Me,chie,Mp]=N_diagramaMC2(p,ley_mat,N,signoM)
%-------------------------------------------------------------------------+
%
% [D_MC,Me,chie,Mp]=N_diagramaMC2(p,ley_mat,N,signoM)
%
% Descripción función:
%   Función que calcula el diagrama Momento-Curvatura de una sección
%
% Inputs:
%   p: conjunto vértices que determinan el polígono que encierra a la
%      sección. Siendo el vector fila i: [yi,zi]
%   ley_mat: ley constitutiva del material.
%            Siendo el vector fila i [eps_i,sig_i]
%   signoM: signo del momento <'p': positivo | 'n': negativo>
%     (Momento positivo: cara superior comprimida)
%
% Outputs:
%   D_MC: diagrama momento curvatura. Array
```

```
623 %       D_MC(:,1): curvatura
624 %       D_MC(:,2): momento
625 %    Me: momento elástico
626 %    chie: curvatura cuando el momento es elástico
627 %    Mp: momento plástico
628 %
629 %  Se asume:
630 %    M>0: cara superior comprimida
631 %    M<0: cara superior traccionada
632 %
633 %  Limitación:
634 %    Para el caso de material con endurecimiento no ofrece solución
635 %  para axiles |Ny|<=|N|<=|Np|
636 %
637 %  Dependencias:
638 %    propSeccion.propGeom, propSeccion.propMaterial,
639 %  problFlexion.N_Mp, problFlexion.N_Me, problFlexion.zpAndN_momento,
640 %  problFlexion.chiAndN_zfn, problFlexion.zfnAndChi_M
641 %
642 %-------------------------------------------------------------------------+
643
644    delta=1E-3;
645    n=100
646
647    Atot=propSeccion.propGeom(p,'area');
648    sig_et=propSeccion.propMaterial(ley_mat,'sig_et');
649    sig_ec=propSeccion.propMaterial(ley_mat,'sig_ec');
650    if N<0, sig_e=sig_ec; else, sig_e=sig_et;end
651    Ny=sig_e*Atot;
652
653 %Determinar rango de valores de profundidad de plastificación(zp),
654 %condición y si es de compresión o tracción
655    [Mp,zfnp,zpmax,signo_zpmax,condic]=problFlexion.N_Mp(p,ley_mat,N,signoM);
656    [Me,zfne,chie]=problFlexion.N_Me(p,ley_mat,N,signoM);
657
658    if N>0, Ny=Atot*sig_et; else, Ny=Atot*sig_ec; end
659
660    if abs(N)>=abs(Ny), D_MC=[]; fprintf('No hay diagrama M-chi para |N|>|%.2f|',
          Ny), return, end
661    signo_zp=signo_zpmax;
662    [~,chip]=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,zpmax*0.95,
          signo_zpmax,condic,N);
663
664    if size(ley_mat,1)>5
665      %Caso de no ser un material perfectamente plástico (ley constitutiva
```

```
        determinada por más de 5 puntos)
666     z1=max(p(:,2));
667     z2=min(p(:,2));
668     h=z1-z2;
669     if signo_zpmax=='c',signo_zpmax2='t';else, signo_zpmax2='c';end
670     if condic==1,condic2=2;else, condic2=1;end
671     zpmax2=h-zpmax;
672     [~,chip2]=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,zpmax2*0.8,
            signo_zpmax2,condic2,N);
673     chip=sign(chip)*max(abs(chip),abs(chip2));
674   end
675
676   chi=linspace(chie,chip,n);
677   D_MC=[];
678
679   for i=1:length(chi)
680     fprintf('\n**chi= %f',chi(i))
681     zfn=problFlexion.chiAndN_zfn(p,ley_mat,zfne,zfnp,chi(i),N);
682     M=problFlexion.zfnAndChi_M(p,ley_mat,zfn,chi(i),N);
683     D_MC=[D_MC;[chi(i),M],zfn];
684   end
685
686 end
687 %
688 %=========================================================================
689
690
691 function [d_int,Np,Mp]=diagrama_interaccionNM(p,ley_mat,signoN,signoM)
692 %-------------------------------------------------------------------------+
693 %
694 %  [d_int,Np,Mp]=diagrama_interaccionNM(p,ley_mat,signoN,signoM)
695 %
696 %  Descripción función:
697 %    Calcular el diagrama de interacción axil - momento
698 %
699 %  Inputs:
700 %    p: conjunto vértices que determinan el polígono que encierra a la
701 %       sección. Siendo el vector fila i: [yi,zi]
702 %    ley_mat: ley constitutiva del material.
703 %             Siendo el vector fila i [eps_i,sig_i]
704 %    signoN: signo del axil <'p': tracción | 'n': compresión>
705 %    signoM: signo del momento <'p': positivo | 'n': negativo>
706 %      (Momento positivo: cara superior comprimida)
707 %
708 %  Outputs:
```

```
709 %    d_int: diagrama de interacción axil - momento
710 %      d_int(:,1): Np
711 %      d_int(:,2): Mp
712 %      d_int(:,3): m, recta de carga (m_i=Np_i/Mp_i)
713 %    Np: axil de plastificación (axil puro)
714 %    Mp: momento de plasifiación (flexión pura)
715 %
716 % Dependencias:
717 %    problFlexion.zfnAndChi_M, propSeccion.propMaterial, problFlexion.N_Mp,
718 % problFlexion.zfnp_NpMp
719 %
720 %-------------------------------------------------------------------------+
721
722   d_int=[]; %nicializamos variable
723   Atot=propSeccion.propGeom(p,'area');
724
725   z1=max(p(:,2));
726   z2=min(p(:,2));
727
728   sig_pc=propSeccion.propMaterial(ley_mat,'sig_pc');
729   sig_pt=propSeccion.propMaterial(ley_mat,'sig_pt');
730
731   [Mp,zfnp]=problFlexion.N_Mp(p,ley_mat,0,signoM);
732   if signoN=='p',sig_p=sig_pt;else sig_p=sig_pc; end
733
734   Np=sig_p*Atot;
735
736   %Límites de zfn
737   if signoM=='p'
738     if signoN=='p'
739       zfn0=zfnp;
740       zfnN=z1;
741     else
742       zfn0=z2;
743       zfnN=zfnp;
744     end
745   else
746     if signoN=='p'
747       zfn0=z2;
748       zfnN=zfnp;
749     else
750       zfn0=zfnp;
751       zfnN=z1;
752     end
753   end
```

```
754
755     zfn=linspace(zfn0,zfnN,100);
756     for i=1:length(zfn)
757         [Np_,Mp_]=problFlexion.zfnp_NpMp(p,ley_mat,zfn(i),signoM);
758         m=Mp_/Np_;
759         d_int=[d_int;[Np_,Mp_,zfn(i),m]];
760     end
761 end
762 %
763 %=========================================================================
764
765
766
767 function [Np,Mp]=zfnp_NpMp(p,ley_mat,zfnp,signoM)
768 %+-----------------------------------------------------------------------+
769 %
770 % [Np,Mp]=zfnp_NpMp(p,ley_mat,zfnp,signoM)
771 %
772 % Descripción función:
773 %   Función que calcula el par de agotamiento Np, Mp dada una posición
774 % de fibra neutra
775 %
776 % Inputs:
777 %   p: conjunto vértices que determinan el polígono que encierra a la
778 %      sección. Siendo el vector fila i: [yi,zi]
779 %   ley_mat: ley constitutiva del material.
780 %            Siendo el vector fila i [eps_i,sig_i]
781 %   zfnp: posición de la f.n. respecto sistema de referencia global(OYZ)
782 %   signoM: signo del momento. <'p':positivo | 'n': negativo>
783 %
784 % Outputs:
785 %   Np: axil de agotamiento
786 %   Mp: momento de agotamiento
787 %
788 % Dependencias:
789 %   propSeccion.propMaterial, problGeom.cambioSistCoord,
790 % problGeom.calculo_cortarArea, propSeccion.propGeom
791 %
792 %+-----------------------------------------------------------------------+
793
794     %Propiedades del material
795     sig_pt=propSeccion.propMaterial(ley_mat,'sig_pt');
796     sig_pc=propSeccion.propMaterial(ley_mat,'sig_pc');
797
798     if signoM=='p'
```

```
799       %Caso en el que el momento es positivo
800       sigp1=sig_pc;
801       sigp2=sig_pt;
802     else
803       %Caso en el que el momento es negativo
804       sigp1=sig_pt;
805       sigp2=sig_pc;
806     end
807
808     p_=problGeom.cambioSistCoord(p,0,-zfnp,0);
809     p1_=problGeom.calculo_cortarArea(p_,[0,0,1],1);
810     p2_=problGeom.calculo_cortarArea(p_,[0,0,1],2);
811     A1_=propSeccion.propGeom(p1_,'area');
812     A2_=propSeccion.propGeom(p2_,'area');
813     W1_=propSeccion.propGeom(p1_,'Wy');
814     W2_=propSeccion.propGeom(p2_,'Wy');
815     zG_=propSeccion.propGeom(p_,'zG');
816     dist=zG_;
817
818     Np= sigp1*A1_+sigp2*A2_;
819     Mp=-sigp1*W1_-sigp2*W2_+dist*Np;
820
821 end
822 %
823 %=========================================================================
824
825
826 function [M,chi,zfn,ze_]=zpAndN_momento(p,ley_mat,zfne,zfnp,zp,signo_zp,condic,N
      )
827 %-------------------------------------------------------------------------+
828 %
829 % [M,chi,zfn,ze_]=zpAndN_momento(p,ley_mat,zfne,zfnp,zp,signo_zp,condic,N)
830 %
831 % Descripción función:
832 %   Función que calcula el momento, dado una profundidad de
833 % plastificación
834 %
835 % Inputs:
836 %   p: conjunto vértices que determinan el polígono que encierra a la
837 %      sección. Siendo el vector fila i: [yi,zi]
838 %   ley_mat: ley constitutiva del material.
839 %            Siendo el vector fila i [eps_i,sig_i]
840 %   zfne: posición de la fibra neutra cuando sección en régimen elástico
841 %         y el axil es N
842 %   zfnp: posición de la fibra neutra cuando sección está plastificada
```

```
%          y el axil es N
%    zp: profundidad de plastificación (medida desde extremo
%        determinado por condic)
%    signo_zp: indica si la profundidad de plastificación es de
%              compresión o tracción <'c': compresión | 't': tracción>
%    condic: indica si la profundidad de plastificación es respecto
%            a la cara superior o inferior
%           <1: cara superior | 2: cara inferior>
%    N: valor del axil de la sección
%
% Outputs:
%    M: momento de la sección
%    chi: curvatura
%    zfn: posición de la fibra neutra referido al sistema de
%         ejes global (OYZ)
%    ze_: posición de la primera fibra plastificada en referencia a la
%        posición de la fibra neutra(O_Y_Z_)
%
% Se asume:
%    M>0: cara superior comprimida
%    M<0: cara superior traccionada
%
% Dependencias:
%    propSeccion.propMaterial, problFlexion.zpAndN_zfn,
% problGeom.cambioSistCoord, problFlexion.zfnAndChi_M
%
%-------------------------------------------------------------------------+


  z1=max(p(:,2));
  z2=min(p(:,2));

  sig_et=propSeccion.propMaterial(ley_mat,'sig_et');
  sig_ec=propSeccion.propMaterial(ley_mat,'sig_ec');
  eps_et=propSeccion.propMaterial(ley_mat,'eps_et');
  eps_ec=propSeccion.propMaterial(ley_mat,'eps_ec');

  if signo_zp=='c'
    sig_e=sig_ec;
    eps_e=eps_ec;
  else
    sig_e=sig_et;
    eps_e=eps_et;
  end

```

```
888     %Cálculo de la posición de la fibra neutra(zfn)
889     zfn=problFlexion.zpAndN_zfn(p,ley_mat,zfne,zfnp,zp,signo_zp,condic,N);
890
891     p_=problGeom.cambioSistCoord(p,0,-zfn,0);
892
893     z1_=max(p_(:,2));
894     z2_=min(p_(:,2));
895
896     %ze_: posición de la primera fibra plastificada considerada respecto
897     %posición de la f.n.
898     if condic==1
899       %zp==zpc
900       ze_=z1_-zp;
901     else
902       %zp==zpt
903       ze_=z2_+zp;
904     end
905
906   %%f.n. cae fuera de la sección(flexión compuesta)
907     %Caso en el que posición de la fibra neutra cae por encima de la
908     %cara superior sección
909     if zfn>z1, ze_=z2_+zp; end
910     %Caso en el que posición de la fibra neutra cae por debajo de la
911     %cara inferior sección
912     if zfn<z2, ze_=z1_-zp;end
913
914     chi=-eps_e/ze_;
915
916     M=problFlexion.zfnAndChi_M(p,ley_mat,zfn,chi,N);
917   end
918   %
919   %=========================================================================
920
921
922
923   function zfn=zpAndN_zfn(p,ley_mat,zfne,zfnp,zp,signo_zp,condic,N)
924   %-------------------------------------------------------------------------+
925   %
926   %  zfn=zpAndN_zfn(p,ley_mat,zfne,zfnp,zp,signo_zp,condic,N)
927   %
928   %  Desripción función:
929   %    Cálculo posición de la fibra neutra dada una profundidad de
930   %  plastificación y el valor del axil
931   %
932   %  Inputs:
```

```
933 %    p: conjunto vértices que determinan el polígono que encierra a la
934 %       sección. Siendo el vector fila i: [yi,zi]
935 %    ley_mat: ley constitutiva del material.
936 %             Siendo el vector fila i [eps_i,sig_i]
937 %    zfnp: posición de la fibra neutra cuando sección está plastificada
938 %          y el axil es N
939 %    zp: profundidad de plastificación (medida desde extremo
940 %        determinado por condic)
941 %    signo_zp: indica si la profundidad de plastificación es de
942 %              compresión o tracción <'c': compresión | 't': tracción>
943 %    condic: indica si la profundidad de plastificación es respecto
944 %            a la cara superior o inferior
945 %           <1: cara superior | 2: cara inferior>
946 %    N: valor del axil de la sección
947 %
948 % Outputs:
949 %    zfn: distancia de la f.n. respecto sistema ejes global (OYZ)
950 %
951 % Dependencias:
952 %    problFlexion.zpAndzfn_N, metNum.approot, metNum.secant, metNum.zero
953 %
954 %-------------------------------------------------------------------------+
955
956   tolerancia=1E-6;
957   delta=1E-3;
958
959
960   z1=max(p(:,2));
961   z2=min(p(:,2));
962
963   h=z1-z2;
964
965   %Cálculo de la posición de fibra neutra en sección agotada
966   if condic==1
967     %Si la profundidad de plástificación es en la cara superior
968     %y es de compresión, está asociado a un momento positivo.
969     %Si la profundidad de plastificación es de tracción, el
970     %momento es negativo
971     if signo_zp=='c', signoM='p';else, signoM='n';end
972   else
973     %Si la profundidad de plastificación es en
974     %la cara inferior, compresiones abajo --> M<0, tracciones -->M>0
975     if signo_zp=='c', signoM='n';else, signoM='p';end
976   end
977
```

```
978     if abs(zfne-zfnp)<1E-3
979     %Como la f.n. se mueve entre zG y zfnp, en este caso es
980     %directo su valor
981         zfn=zfne;
982     else
983
984     f= @(zfn) N-problFlexion.zpAndzfn_N(p,ley_mat,zp,signo_zp,condic,zfn);
985
986
987     if condic==2, b=max(zfne,zfnp)+delta;a=z2+zp+delta;
988     else, b=min(zfne,zfnp)+delta; a=z1-zp-delta; end
989
990     n=5;
991     [R,x]=metNum.approot(f,a,b,n,tolerancia);
992     if length(x)==0
993       p0=b;
994       p1=a;
995       zfn=metNum.secant(f,p0,p1,tolerancia);
996     else
997       p0=x(1,1);
998       p1=x(1,2);
999       zfn=metNum.zero(p0,p1,eps,tolerancia,f);
1000    end %length(x)==0
1001
1002    end
1003  end
1004  %=========================================================================
1005
1006
1007  function N=zpAndzfn_N(p,ley_mat,zp,signo_zp,condic,zfn)
1008  %-------------------------------------------------------------------------+
1009  %
1010  % N=zpAndzfn_N(p,ley_mat,zp,signo_zp,condic,zfn)
1011  %
1012  % Descripción función:
1013  %   Función que calcula el axil dada la profundidad de plastificación
1014  % y la posición de la fibra neutra
1015  %
1016  % Inputs:
1017  %   p: conjunto vértices que determinan el polígono que encierra a la
1018  %      sección. Siendo el vector fila i: [yi,zi]
1019  %   ley_mat: ley constitutiva del material.
1020  %            Siendo el vector fila i [eps_i,sig_i]
1021  %   zp: profundidad de plastificación
1022  %   signo_zp: tipo de plastificación
```

```
1023 %              <'c': compresión | 't': tracción>
1024 %    condic: indica si la profundidad de plastificación es en referencia
1025 % a la cara superior o inferior
1026 % <1: cara superior | 2: cara inferior>
1027 %
1028 % Outputs:
1029 %    N: axil estáticamente equivalente al diagrama de tensiones normales
1030 %       con una posición de la fibra neutra y profundidad de
1031 %       plástificación especificadas.
1032 %
1033 % Dependencias:
1034 %    propSeccion.propMaterial, problGeom.cambioSistCoord,
1035 % problFlexion.zfnAndChi_N
1036 %
1037 %+-------------------------------------------------------------------------+
1038
1039   z1=max(p(:,2));
1040   z2=min(p(:,2));
1041
1042   eps_et=propSeccion.propMaterial(ley_mat,'eps_et');
1043   eps_ec=propSeccion.propMaterial(ley_mat,'eps_ec');
1044
1045   if signo_zp=='c'
1046     eps_e=eps_ec;
1047   else
1048     eps_e=eps_et;
1049   end
1050
1051   p_=problGeom.cambioSistCoord(p,0,-zfn,0);
1052
1053   z1_=max(p_(:,2));
1054   z2_=min(p_(:,2));
1055
1056   %ze: posición de la primera fibra plástificada considerada respecto
1057   %posición de la f.n.
1058   if condic==1
1059     ze=z1_-zp;
1060   else
1061     ze=z2_+zp;
1062   end
1063
1064 %%%Caso en que f.n. cae fuera de la sección
1065   %Caso en el que posición de la fibra neutra cae por encima de la
1066   %cara superior sección
1067   if zfn>z1, ze=z2_+zp; end
```

```
1068
1069    %Caso en el que posición de la fibra neutra cae por debajo de la
1070    %cara inferior sección
1071    if zfn<z2, ze=z1_-zp; end
1072  %%%%
1073
1074    chi=-eps_e/ze;
1075    N=problFlexion.zfnAndChi_N(p,ley_mat,zfn,chi);
1076
1077  end
1078  %=========================================================================
1079
1080
1081  function [M,zfn]=chiAndN_momento(p,ley_mat,zfne,zfnp,chi,N)
1082  %+-----------------------------------------------------------------------+
1083  %
1084  % [M,zfn]=chiAndN_momento(p,ley_mat,zfne,zfnp,chi,N)
1085  %
1086  % Descripción función:
1087  %   Calcular el momento a partir del valor de la curvatura y
1088  % el axil de la sección
1089  %
1090  % Inputs:
1091  %   p: conjunto vértices que determinan el polígono que encierra a la
1092  %      sección. Siendo el vector fila i: [yi,zi]
1093  %   ley_mat: ley constitutiva del material.
1094  %            Siendo el vector fila i [eps_i,sig_i]
1095  %   zfne: posición de la fibra neutra cuando sección en régimen elástico
1096  %         y el axil es N
1097  %   zfnp: posición de la fibra neutra cuando sección está plastificada
1098  %         y el axil es N
1099  %   chi: curvatura
1100  %   condic: indica si la profundidad de plastificación es respecto
1101  %           a la cara superior o inferior
1102  %          <1: cara superior | 2: cara inferior>
1103  %   N: valor del axil de la sección
1104  %
1105  % Outputs:
1106  %   M: momento
1107  %   zfn: posición de la fibra neutra referido al sistema de ejes
1108  %        global (OYZ)
1109  %
1110  % Dependencias:
1111  %   problFlexion.chiAndN_zfn, problFlexion.zfnAndChi_M
1112  %
```

```
%-------------------------------------------------------------------------+

    %Calcular posición de la fibra media
    zfn=problFlexion.chiAndN_zfn(p,ley_mat,zfne,zfnp,chi,N);

    %Calcular momento
    M=problFlexion.zfnAndChi_M(p,ley_mat,zfn,chi,N);
end
%=========================================================================


function zfn=chiAndN_zfn(p,ley_mat,zfne,zfnp,chi,N)
%-------------------------------------------------------------------------+
%
% zfn=zpAndN_zfn(p,ley_mat,zfnp,zp,signo_zp,condic)
%
% Desripción función:
%   Cálculo posición de la fibra neutra dada una profundidad de
% plastificación en flexión pura(ha de cumplir condición N=0)
%
% Inputs:
%   p: conjunto vértices que determinan el polígono que encierra a la
%      sección. Siendo el vector fila i: [yi,zi]
%   ley_mat: ley constitutiva del material.
%            Siendo el vector fila i [eps_i,sig_i]
%   zfne: posición de la fibra neutra cuando sección en régimen elástico
%         y el axil es N
%   zfnp: posición de la fibra neutra cuando sección está plastificada
%         y el axil es N
%   chi: curvatura
%   condic: indica si la profundidad de plastificación es respecto
%           a la cara superior o inferior
%          <1: cara superior | 2: cara inferior>
%   N: valor del axil de la sección
%
% Outputs:
%   zfn: distancia de la f.n. respecto sistema ejes global (OYZ)
%
% Dependencias:
%   problFlexion.zfnAndChi_N, metNum.approot, metNum.secant, metNum.zero
%
%-------------------------------------------------------------------------+

  tolerancia=1E-6;
  delta=1E-3;
```

```
1158
1159    z1=max(p(:,2));
1160    z2=min(p(:,2));
1161
1162    if zfne>zfnp, a=zfne+delta;b=z2-delta;
1163    else, a=zfne-delta; b=z1+delta;
1164    end
1165
1166    if abs(zfne-zfnp)<1E-3
1167        zfn=zfne;
1168    else
1169      f= @(zfn) N-problFlexion.zfnAndChi_N(p,ley_mat,zfn,chi);
1170      n=5;
1171      [R,x]=metNum.approot(f,a,b,n,tolerancia);
1172      if length(x)==0
1173          p0=a;
1174          p1=b;
1175         zfn=metNum.secant(f,p0,p1,tolerancia);
1176      else
1177        p0=x(1,1);
1178        p1=x(1,2);
1179        zfn=metNum.zero(p0,p1,eps,tolerancia,f);
1180      end %length(x)==0
1181    end
1182 end
1183 %
1184 %=========================================================================
1185
1186
1187
1188 function [zfn,chi]=NAndM_zfnAndChi(p,ley_mat,N,M)
1189 %-------------------------------------------------------------------------+
1190 %
1191 % [zfn,chi]=NAndM_zfnAndChi(p,ley_mat,N,M)
1192 %
1193 % Descripción función:
1194 %   Calcular los parámetros que caracteriza una distribución de
1195 % tensiones(posición de f.n. y curvatura) asociado a un axil y momento
1196 %
1197 % Inputs:
1198 %   p: conjunto vértices que determinan el polígono que encierra a la
1199 %      sección. Siendo el vector fila i: [yi,zi]
1200 %   ley_mat: ley constitutiva del material.
1201 %            Siendo el vector fila i [eps_i,sig_i]
1202 %   N: valor del axil
```

```
%   M: valor del momento
%
% Outputs:
%   zfn: posición de la fibra neutra en referencia al sistema de
%        ejes global (OYZ)
%   chi: curvatura de la sección asociada a la solicitación (N,M)
%
% Dependencias:
%   propSeccion.propGeom, propSeccion.propMaterial, problFlexion.N_Mp,
% problFlexion.N_Me, problFlexion.zpAndN_momento,
% problGeom.calculo_cambioSistCoord, problGeom.calculo_cambioSistCoord,
% metNum.approot, metNum.secant, metNum.zero,
% problFlexion.chiAndN_momento
%
%+-------------------------------------------------------------------------+

  Atot=propSeccion.propGeom(p,'area');

  sig_ec=propSeccion.propMaterial(ley_mat,'sig_ec');
  sig_et=propSeccion.propMaterial(ley_mat,'sig_et');

  sig_pc=propSeccion.propMaterial(ley_mat,'sig_pc');
  sig_pt=propSeccion.propMaterial(ley_mat,'sig_pt');

  if N<0, sig_e=sig_ec; else, sig_e=sig_et;end
  if N<0, sig_p=sig_pc; else, sig_p=sig_pt;end

  Ny=sig_e*Atot;
  Np=sig_p*Atot;

  tita=0;

  if abs(N)>abs(0.99*Np)
    disp('Sección agotada')
    zfn=NaN;
    chi=NaN;
    return

  elseif abs(N)>=abs(Ny)
    disp('|Np|>=|N|>=|Ny|, no se dispone de solución para este rango de valores
        del axil.')
    zfn=NaN;
    chi=NaN;
    return
```

```
1247    elseif M==0 %axil puro
1248      zfn=1E16;
1249      chi=0;
1250    else
1251
1252      if M>0, signoM='p'; else, signoM='n'; end
1253
1254      [Mp,zfnp,zpmax,signo_zpmax,condic]=problFlexion.N_Mp(p,ley_mat,N,signoM);
1255
1256      signo_zp=signo_zpmax;
1257
1258      [Me,zfne,chie,~]=problFlexion.N_Me(p,ley_mat,N,signoM);
1259      [Mmax,chimax]=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,zpmax*0.95,
              signo_zpmax,condic,N);
1260
1261      if size(ley_mat,1)>5
1262        %Caso de no ser un material perfectamente plástico
1263        %(ley constitutiva determinada por más de 5 puntos)
1264        z1=max(p(:,2));
1265        z2=min(p(:,2));
1266        h=z1-z2;
1267        if signo_zpmax=='c',signo_zpmax2='t';else, signo_zpmax2='c';end
1268        if condic==1, condic2=2; else, condic2=1; end
1269        zpmax2=h-zpmax;
1270        [Mmax2,chimax2]=problFlexion.zpAndN_momento(p,ley_mat,zfne,zfnp,zpmax2
              *0.95,signo_zpmax2,condic2,N);
1271        if abs(chimax2)>abs(chimax)
1272          chimax=chimax2;
1273          Mmax=Mmax2;
1274        end
1275      end
1276
1277      if abs(M)<=abs(Me)
1278        disp('Régimen elástico')
1279        Iy=propSeccion.propGeom(p,'IGy');
1280        zG=propSeccion.propGeom(p,'zG');
1281        z1=max(p(:,2));
1282        z2=min(p(:,2));
1283        E=propSeccion.propMaterial(ley_mat,'E');
1284        chi=M/(E*Iy);
1285        p_=problGeom.calculo_cambioSistCoord(p,0,-zG,0);
1286        %Cálculo tensión en cara superior e inferior
1287        z1_=max(p_(:,2));
1288        z2_=min(p_(:,2));
1289        sig1=N/Atot-M/Iy*z1_;
```

```
1290          sig2=N/Atot-M/Iy*z2_;
1291          zfn=(z1*sig2-z2*sig1)/(sig2-sig1);
1292
1293        elseif abs(M)<=abs(Mmax)
1294         disp('Régimen elasto-plástico')
1295
1296          delta=1E-6;
1297          tolerancia=1E-6;
1298
1299          f=@(chi) M-problFlexion.chiAndN_momento(p,ley_mat,zfne,zfnp,chi,N);
1300
1301          a=chie;
1302          b=chimax;
1303
1304          n=5;
1305          [R,x]=metNum.approot(f,a,b,n,tolerancia);
1306
1307          if length(x)==0
1308            chi0=a;
1309            chi1=b;
1310            chi=metNum.secant(f,chi0,chi1,tolerancia);
1311          else
1312               chi0=x(1,1);
1313               chi1=x(1,2);
1314               chi=metNum.zero(chi0,chi1,eps,tolerancia,f);
1315          end
1316          [M,zfn]=problFlexion.chiAndN_momento(p,ley_mat,zfne,zfnp,chi,N);
1317        elseif abs(M)<=abs(Mp)
1318          disp('Sección agotada')
1319            zfn=zfnp;
1320            chi=1E16;
1321        else
1322          disp('No se puede dar el caso. Punto (N,M) cae fuera de la superficie de
                 interacción.')
1323          zfn=NaN;
1324          chi=NaN;
1325        end
1326     end
1327  end
1328  %
1329  %=========================================================================
1330
1331
1332
1333  function [Ne,Me,zfne,chie,condic]=NM_NeMe(p,ley_mat,N,M)
```

```
1334 %+------------------------------------------------------------------------+
1335 %
1336 % [Ne,Me,zfne,chie,condic]=NM_NeMe(p,ley_mat,N,M)
1337 %
1338 % Descripción función:
1339 %   Calcular el momento-axil máximo que se encuentra en régimen
1340 % elástico dada una recta de carga determinada por N y M
1341 %
1342 % Inputs:
1343 %   p: conjunto vértices que determinan el polígono que encierra a la
1344 %      sección. Siendo el vector fila i: [yi,zi]
1345 %   ley_mat: ley constitutiva del material.
1346 %            Siendo el vector fila i [eps_i,sig_i]
1347 %   N: valor del axil
1348 %   M: valor del momento
1349 %
1350 % Outputs:
1351 %   Ne: valor del axil
1352 %   Me: valor del momento
1353 %   zfne: posición de la f.n. cuando solicitación es (Ne,Me)
1354 %   chie: curvatura cuando solicitación es (Ne,Me)
1355 %   condic: indica el extremo que plastifica primero
1356 %           <1: extremo superior | 2: extremo inferior>
1357 %
1358 % Dependencias:
1359 %   propSeccion.propGeom, problGeom.calculo_cambioSistCoord,
1360 % propSeccion.propMaterial
1361 %
1362 %+------------------------------------------------------------------------+
1363
1364   z1=max(p(:,2));
1365   z2=min(p(:,2));
1366
1367   Iy=propSeccion.propGeom(p,'IGy');
1368   zG=propSeccion.propGeom(p,'zG');
1369   Atot=propSeccion.propGeom(p,'area');
1370   p_=problGeom.calculo_cambioSistCoord(p,0,-zG,0);
1371   z1_=max(p_(:,2));
1372   z2_=min(p_(:,2));
1373
1374   E=propSeccion.propMaterial(ley_mat,'E');
1375   sig_ec=propSeccion.propMaterial(ley_mat,'sig_ec');
1376   sig_et=propSeccion.propMaterial(ley_mat,'sig_et');
1377
1378   if N<0
```

```
1379        if M>0
1380          sig_e1=sig_ec;
1381          sig_e2=sig_et;
1382        else
1383          sig_e1=sig_et;
1384          sig_e2=sig_ec;
1385        end
1386      else
1387        if M<0
1388          sig_e1=sig_et;
1389          sig_e2=sig_ec;
1390        else
1391          sig_e1=sig_ec;
1392          sig_e2=sig_et;
1393        end
1394      end
1395      m=M/N;
1396      %Suponemos que plastifica primero cara superior
1397      %Por tanto, sig1==sige1 y sig2<=sig_e2
1398      condic=1;
1399      Ne=sig_e1/(1/Atot-z1_*m/Iy);
1400      Me=m*Ne;
1401      sig2=Ne/Atot-Me*z2_/Iy;
1402      if sig2>sig_e2
1403        condic=2;
1404        Ne=sig_e2/(1/Atot-z2_*m/Iy);
1405        Me=Ne*m;
1406      end
1407
1408      sig1=Ne/Atot-Me*z1_/Iy;
1409      sig2=Ne/Atot-Me*z2_/Iy;
1410
1411      zfne=(sig2*z1-z2*sig1)/(sig2-sig1);
1412      chie=-Me/(E*Iy);
1413
1414  end
1415  %=========================================================================
1416
1417
1418
1419  function [Np,Mp,zfnp,zpmax,signo_zpmax,condic]=NM_NpMp(p,ley_mat,N,M)
1420  %-------------------------------------------------------------------------+
1421  %
1422  %  [Np,Mp,zfnp,zpmax,signo_zpmax,condic]=NM_NpMp(p,ley_mat,N,M)
1423  %
```

```
1424 % Descripción función:
1425 %   Calcular el momento-axil de agotamiento dada una recta de carga
1426 % determinada por N y M
1427 %
1428 % Inputs:
1429 %   p: conjunto vértices que determinan el polígono que encierra a la
1430 %      sección. Siendo el vector fila i: [yi,zi]
1431 %   ley_mat: ley constitutiva del material.
1432 %            Siendo el vector fila i [eps_i,sig_i]
1433 %   N: valor del axil
1434 %   M: valor del momento
1435 %
1436 % Outputs:
1437 %   Np: axil plástico de la sección p asociado al momento Mp
1438 %   Mp: momento plástico de la sección p asociado al axil Np
1439 %   zfnp: posición de la fibra neutra cuando solicitación es: (N,Mp)
1440 %   zpmax: valor de la profundidad de plastificación de la sección
1441 %          medida desde el extrem ue plastifica primero cuando
1442 %          solicitación es (N,Mp)
1443 %   signo_zpmax: carácter de la profundidad de plastificación (zpmax)
1444 %                <'t':tracción | 'c':compresión >
1445 %   condic: indica ué extremo(superior/inferior) plastifica primero
1446 %           <1: extremo superior | 2: extremo inferior >
1447 % Dependencias:
1448 %   propSeccion.propMaterial, problFlexion.NM_NeMe,
1449 % problFlexion.diagrama_interaccionNM, problFlexion.zfnp_NpMp
1450 %
1451 %-------------------------------------------------------------------------+
1452
1453   z1=max(p(:,2));
1454   z2=min(p(:,2));
1455
1456   sig_pt=propSeccion.propMaterial(ley_mat,'sig_pt');
1457   sig_pc=propSeccion.propMaterial(ley_mat,'sig_pc');
1458
1459   if N<0, signoN='n'; else, signoN='p';end
1460   if M<0, signoM='n'; else, signoM='p';end
1461
1462   if signoM=='p'
1463     %Caso en el que el momento es positivo
1464     sigp1=sig_pc;
1465     sigp2=sig_pt;
1466   else
1467     %Caso en el que el momento es negativo
1468     sigp1=sig_pt;
```

```
1469        sigp2=sig_pc;
1470      end
1471
1472      [~,~,~,~,condic]=problFlexion.NM_NeMe(p,ley_mat,N,M);
1473
1474      m=M/N;
1475      d_int=problFlexion.diagrama_interaccionNM(p,ley_mat,signoN,signoM);
1476      %Como ma<m<mb
1477      index_a=find(abs(d_int(:,4))<=abs(m));
1478      if m>0, index_a=min(index_a); else index_a=max(index_a); end
1479      ma=d_int(index_a,4);
1480      index_b=find(abs(d_int(:,4))>abs(m));
1481      if m>0, index_b=max(index_b); else, index_b=min(index_b); end
1482      mb=d_int(index_b,4);
1483      zfnp_a=d_int(index_a,3);
1484      zfnp_b=d_int(index_b,3);
1485
1486      zfnp=((m-mb)*zfnp_a+(ma-m)*zfnp_b)/(ma-mb);
1487
1488      [Np,Mp]=problFlexion.zfnp_NpMp(p,ley_mat,zfnp,signoM);
1489
1490      %Cálculo de zpmax
1491      if condic==1
1492        zpmax=z1-zfnp;
1493        if sigp1<0, signo_zpmax='c';
1494        else, signo_zpmax='t'; end
1495      else
1496        zpmax=zfnp-z2;
1497        if sigp2<0, signo_zpmax='c';
1498        else, signo_zpmax='t'; end
1499      end
1500  end
1501  %
1502  %=========================================================================
1503
1504  function [Ap1,Ap2]=calcular_areasPlastificadas(p,z_)
1505  %-------------------------------------------------------------------------+
1506  %
1507  %  [Ap1,Ap2]=calcular_areasPlastificadas(p,z_)
1508  %
1509  %  Descripción función:
1510  %    Función que muestra las áreas plastificadas
1511  %
1512  %  Inputs:
1513  %    p: conjunto vértices que determinan el polígono que encierra a la
```

```
%       sección. Siendo el vector fila i: [yi,zi]
%    z_: valores z_ del diagrama de tensiones(referidos a la posición de
%        la fibra neutra)
%
% Outputs:
%    Ap1: área plastificada que queda por encima de la f.n.
%    Ap2: área plastificada que queda por debajo de la f.n.
%
% Dependencias:
%    problGeom.calculo_cortarArea, problGeom.cambioSistCoord
%
%-------------------------------------------------------------------------+
   if size(z_,1)<2, Ap1=[]; Ap2=[]; return; end
   zmin_p=min(p(:,2));
   ymin_p=min(p(:,1));
   p=problGeom.cambioSistCoord(p,-ymin_p,-zmin_p,0); %Ponemos (0,0) en
                                                     %la esquina inferior izquierda

   %Calcular ze1 y ze2
   ze1=z_(find(z_==0)+1);
   ze2=z_(find(z_==0)-1);

   %Calcular zfn
   if length(z_)==2, zfn=1E16; else zfn=min(z_);end
   if zfn>0; zfn=-zfn; else zfn=abs(zfn);end

   p_=problGeom.cambioSistCoord(p,0,-zfn,0);

   if length(ze1)==0
      Ap1=[];
   else
      Ap1=problGeom.calculo_cortarArea(p_,[0,ze1,1],1);
      Ap1=problGeom.calculo_cambioSistCoord(Ap1,0,zfn,0);
   end

   if length(ze2)==0
      Ap2=[];
   else
      Ap2=problGeom.calculo_cortarArea(p_,[0,ze2,1],2);
      Ap2=problGeom.calculo_cambioSistCoord(Ap2,0,zfn,0);
   end

   if length(ze1)==0 && length(ze2)==0
      z_=unique(z_);
      if length(z_)>2
```

```
1559        if z_(1)>0
1560          %zfn cae por debajo de la sección
1561          ze1=z_(end-1);
1562          Ap1=problGeom.calculo_cortarArea(p_,[0,z_(end-1),1],1);
1563          Ap1=problGeom.calculo_cambioSistCoord(Ap1,0,zfn,0);
1564        else
1565          ze2=z_(2);
1566          Ap2=problGeom.calculo_cortarArea(p_,[0,z_(2),1],2);
1567          Ap2=problGeom.calculo_cambioSistCoord(Ap2,0,zfn,0);
1568        end
1569      end
1570    end
1571  end
1572  %
1573  %=========================================================================
1574
1575  end
1576  end
```

### B.3.5. *propSeccion*

```
1   classdef propSeccion
2   methods(Static)
3
4   %*************************************************************************%
5   %                                                                         %
6   %  propSeccion: módulo que recoge las funciones para la                   %
7   %                    generación de las propiedades de la sección          %
8   %    propMaterial(): función para hallar la propiedad del material        %
9   %                    escogida                                             %
10  %    propGeom(): calcular las propiedades geométricas de la sección       %
11  %      perimetro(): calcular el perímetro                                 %
12  %      calculo_area(): calcular el área de un polígono compuesto/simple   %
13  %      Area(): calcular el área de un polígono simple                     %
14  %      calculo_Gn(): calcular el cdg de la coordenada n de un polígono    %
15  %                    compuesto                                            %
16  %      G_n(): calcular el cdg de la coordenada n de un polígono simple    %
17  %      calculo_Inn(): calcular el momento de inercia respecto eje n       %
18  %                    de un polígono compuesto/simple                      %
19  %      I_nn(): calcular el momento de inercia respecto el eje n de un     %
20  %              polígono simple                                            %
21  %      I_nm(): calcular el producto de inercia de un polígono simple      %
22  %      calculo_W_n(): calcular el momento estático de un polígono         %
23  %                    simple/compuesto                                     %
24  %      W_n(): calcular el momento estático de un polígono simple          %
```

```
25 %                                                                      %
26 %                                                                      %
27 %  Se Asume:                                                          %
28 %    polígono simple: región determinada por un solo polígono(array)   %
29 %    polígono compuesto: región determinada por un conjunto de         %
30 %                        polígonos (cell)                              %
31 %                                                                      %
32 %**********************************************************************%
33
34
35 function propVal=propMaterial(ley_mat,prop)
36 %----------------------------------------------------------------------+
37 %
38 %  propVal=propMaterial(ley_mat,prop)
39 %
40 %  Descripción:
41 %    Función que calcula la propiedad del material escogida
42 %
43 %  Inputs:
44 %    ley_mat: ley constitutiva del material. Array(Nx2). N puntos
45 %             característicos de la ley constitutiva. (eps,sig)i i={1..N}
46 %
47 %  Outputs:
48 %    propVal: valor de la propiedad escogida. String.
49 %             Opciones: {'E', 'sig_pc', 'sig_pt', 'eps_pc', 'eps_pt',
50 %                        'sig_ec', 'sig_et', 'sig_ec', 'eps_et'}
51 %
52 %             Siendo:
53 %               E: módulo de Young
54 %               sig_pc: tensión plástica a compresión
55 %               sig_pt: tensión plástica a tracción
56 %               eps_pc: deformación de plastificación a compresión
57 %               eps_pt: deformación de plastificación a tracción
58 %               sig_ec: tensión en el límite elástico a compresión
59 %               sig_et: tensión en el límite elástico a tracción
60 %               eps_ec: deformación en el límite elástico a compresión
61 %               eps_et: deformación en el límite elástico a tracción
62 %
63 %----------------------------------------------------------------------+
64
65   switch prop
66     case 'E'
67       sig_et=ley_mat(find(ley_mat(:,1)==0)+1,2);
68       eps_et=ley_mat(find(ley_mat(:,1)==0)+1,1);
69       E=sig_et/eps_et;
```

```
70          propVal=E;
71      case 'sig_et'
72          sig_et=ley_mat(find(ley_mat(:,1)==0)+1,2);
73          propVal=sig_et;
74      case 'sig_ec'
75          sig_ec=ley_mat(find(ley_mat(:,1)==0)-1,2);
76          propVal=sig_ec;
77      case 'eps_et'
78          eps_et=ley_mat(find(ley_mat(:,1)==0)+1,1);
79          propVal=eps_et;
80      case 'eps_ec'
81          eps_ec=ley_mat(find(ley_mat(:,1)==0)-1,1);
82          propVal=eps_ec;
83      case 'sig_pt'
84          sig_pt=max(ley_mat(:,2));
85          propVal=sig_pt;
86      case 'sig_pc'
87          sig_pc=min(ley_mat(:,2));
88          propVal=sig_pc;
89      case 'eps_pt'
90          eps_pt=ley_mat(length(ley_mat)-1,1);
91          propVal=eps_pt;
92      case 'eps_pc'
93          eps_pc=ley_mat(2,1);
94          propVal=eps_pc;
95      otherwise
96          error(['Introducir un valor de propiedad de material válida.',...
97                 ' Opciones: E, sig_et, sig_ec, eps_et, eps_ec',...
98                  'sig_pt, sig_pc, eps_pt, eps_pc'])
99    end
100 end
101 %
102 %=========================================================================
103
104
105 function propVal=propGeom(p,prop)
106 %-------------------------------------------------------------------------+
107 %
108 %  propVal=propGeom(p,prop)
109 %
110 %  Descripción función:
111 %    Calcular el valor de una propiedad geométrica, dado los vértices
112 %    de un polinomio.
113 %
114 %  Inputs:
```

```
115 %    p: vértices que definen polinomio. Array (Nx2), N: núm. de vértices
116 %    prop: propiedad que se desea calcular. String.
117 %          opciones:{'area','IGy','IGz','Iy','Iz','zG','yG','Wy','Wz'}
118 %
119 % Outputs:
120 %    propVal: valor de la propiedad escogida
121 %
122 % Dependencias:
123 %    propSeccion.perimetro, propSeccion.calculo_area, propSeccion.G_n,
124 % propSeccion.calculo_I_nn, problGeom.cambioSistCoord,
125 % propSeccion.calculo_W_n
126 %
127 %-------------------------------------------------------------------------+
128
129   switch prop
130     case 'perimetro'
131       propVal=propSeccion.perimetro(p);
132     case 'area'
133       propVal=propSeccion.calculo_area(p);
134     case 'yG'
135       propVal=propSeccion.G_n(p,1);
136     case 'zG'
137       propVal=propSeccion.calculo_G_n(p,2);
138     case 'Iy'
139       propVal=propSeccion.calculo_I_nn(p,1);
140     case 'IGy'
141       zG=propSeccion.G_n(p,2);
142       p1=problGeom.cambioSistCoord(p,0,-zG,0);
143       propVal=propSeccion.calculo_I_nn(p1,1);
144     case 'Iz'
145       propVal=propSeccion.calculo_I_nn(p,2);
146     case 'IGz'
147       yG=propSeccion.G_n(p,1);
148       p1=problGeom.cambioSistCoord(p,-yG,0,0);
149       propVal=propSeccion.calculo_I_nn(p1,2);
150     case 'Wy'
151       propVal=propSeccion.calculo_W_n(p,1);
152     case 'Wz'
153       propVal=propSeccion.calculo_W_n(p,2);
154     otherwise
155       error(['Opción de propiedad no válida. Opciones válidas:',...
156              ' area, yG, zG, Iy, Iz, IGy, IGz, Wy, Wz'])
157   end
158 end
159 %
```

```
160 %=========================================================================
161
162 function per=perimetro(p)
163 %-------------------------------------------------------------------------+
164 %
165 % Descripción función:
166 %   Calcular el perímetro de un polígono
167 %
168 % Inputs:
169 %   p: conjunto de vértices(array) que definen el polígono
170 %
171 % Outputs:
172 %   per: perímetro
173 %
174 %-------------------------------------------------------------------------+
175   nver=length(p);
176   per=0; %inicializamos variable
177     for i=1:nver
178       j=i+1;
179       if i==nver
180         j=1;
181       end
182       per=per+sqrt((p(i,1)-p(j,1))^2+(p(i,2)-p(j,2))^2);
183     end
184 end
185 %
186 %=========================================================================
187
188 function A=calculo_area(p)
189 %-------------------------------------------------------------------------+
190 %
191 % Descripción función:
192 %   Calcular el área de un polígono compuesto(cell) o simple(array)
193 %
194 % Inputs:
195 %   p: polígono simple(array) o compuesto(array)
196 %
197 % Outputs:
198 %   A=área de p
199 %
200 % Dependencias:
201 %   propSeccion.Area
202 %
203 %-------------------------------------------------------------------------+
204   if iscell(p)
```

```
205       A=0;
206       for i=1:length(p)
207         A=A+propSeccion.Area(p{i});
208       end
209     else
210       A=propSeccion.Area(p);
211     end
212   end
213   %
214   %=========================================================================
215
216   function A=Area(p)
217   %-------------------------------------------------------------------------+
218   %
219   % Descripción función:
220   %   Calcular el área de un polígono simple(array)
221   %
222   % Inputs:
223   %   p: vértices del polígono simple(array)
224   %
225   % Outputs:
226   %   A=área de p
227   %
228   %-------------------------------------------------------------------------+
229     A=0;
230     nver=length(p);
231     for i=1:nver
232       j=i+1;
233       if i==nver
234         j=1;
235       end
236       A=A+(p(i,1)-p(j,1))*(p(i,2)+p(j,2))/2;
237     end
238   end
239   %
240   %=========================================================================
241
242
243   function Gn=calculo_G_n(p,n)
244   %-------------------------------------------------------------------------+
245   %
246   % Descripción función:
247   %   Calcular la coordenada del centro de gravedad respecto eje n de un
248   % polígono compuesto(cell) o simple(array)
249   %
```

```
250 %  Inputs:
251 %    p: polígono simple(array) o compuesto(array)
252 %    n: eje respecto al cual ueremos calcular la coordenada del c.d.g.
253 %       <1: eje de abscisas | 2: eje ordenado>
254 %  Outputs:
255 %    Gn= coordenada del c.d.g.
256 %
257 %  Dependencias:
258 %    propSeccion.G_n
259 %-------------------------------------------------------------------------+
260   if iscell(p)
261     p=conjuntoSimpleArrays(p);
262     A=0;
263     Gn=0;
264     for i=1:length(p)
265       Ai=Area(p{i});
266       A=A+Ai;
267       Gn=Gn+propSeccion.G_n(p{i},n)*Ai;
268     end
269     Gn=Gn/A;
270   else
271     Gn=propSeccion.G_n(p,n);
272   end
273 end
274 %
275 %=========================================================================
276
277
278 function Gn=G_n(p,n)
279 %-------------------------------------------------------------------------+
280 %
281 %  Descripción función:
282 %    Calcular la coordenada del centro de gravedad respecto eje n de un
283 %  polígono simple(array)
284 %
285 %  Inputs:
286 %    p: polígono simple(array)
287 %    n: eje respecto al cual queremos calcular la coordenada del c.d.g.
288 %       <1: eje de abscisas | 2: eje ordenado>
289 %  Outputs:
290 %    Gn= coordenada del c.d.g.
291 %  Dependencias:
292 %    propSeccion.W_n, propSeccion.Area
293 %
294 %-------------------------------------------------------------------------+
```

```
295     if n==1
296       m=2;
297     else
298       m=1;
299     end
300     Gn=propSeccion.W_n(p,m)/propSeccion.Area(p);
301   end
302   %
303   %=========================================================================
304
305
306   function Inn=calculo_I_nn(p,n)
307   %-------------------------------------------------------------------------+
308   %
309   % Descripción función:
310   %   Calcular el momento de inercia respecto el eje n de un polígono
311   % simple o copuesto
312   %
313   % Inputs:
314   %   p: polígono simple(array) o compuesto(array)
315   %   n: eje respecto al cual queremos calcular la coordenada del c.d.g.
316   %      <1: eje de abscisas | 2: eje ordenado>
317   %
318   % Outputs:
319   %   Inn: momento de inercia respecto eje n
320   %
321   % Dependencias:
322   %   propSeccion.I_nn
323   %
324   %-------------------------------------------------------------------------+
325     if iscell(p)
326       Inn=0;
327       for i=1:length(p)
328         Inn=Inn+propSeccion.I_nn(p{i},n);
329       end
330     else
331       Inn=propSeccion.I_nn(p,n);
332     end
333   end
334   %
335   %=========================================================================
336
337   function Inn=I_nn(p,n)
338   %-------------------------------------------------------------------------+
339   %
```

```
340 % Descripción función:
341 %   Calcular el momento de inercia de un polígono simple respecto el
342 % eje n
343 %
344 % Inputs:
345 %   p: polígono simple(array)
346 %   n: eje respecto al cual queremos calcular el momento de inercia
347 %      < 1: eje de abscisas | 2: eje ordenado >
348 %
349 % Outputs:
350 %   Inn: momento de inercia de p respecto de n
351 %
352 %-------------------------------------------------------------------------+
353   Inn=0;
354   if n==1
355     m=2;
356   else
357     m=1;
358   end
359   nver=length(p);
360   for i=1:nver
361     j=i+1;
362     if i==nver
363       j=1;
364     end
365     Inn=Inn+(-1)^n*(p(j,n)-p(i,n))*(p(i,m)^3+ ...
366     p(i,m)^2*p(j,m)+p(i,m)*p(j,m)^2+p(j,m)^3)/12;
367   end
368 end
369 %
370 %=========================================================================
371
372
373 function Inm=I_nm(p)
374 %-------------------------------------------------------------------------+
375 %
376 % Descripción función:
377 %   Calcular producto de inercia de un polígono simple
378 %
379 % Inputs:
380 %   p: polígono simple
381 %
382 % Outputs:
383 %   Inm: producto de inercia
384 %
```

```
385 %-------------------------------------------------------------------------+
386   Inm=0;
387   nver=length(p);
388   for i=1:nver
389     j=i+1;
390     if i==nver
391       j=1;
392     end
393     w_A=9*p(i,2)^2+6*p(i,2)*p(j,2)+3*p(j,2)^2; %var.auxiliar
394     w_B=3*p(i,2)^2+6*p(i,2)*p(j,2)+9*p(j,2)^2; %var.auxiliar
395     Inm=Inm+(p(i,1)-p(j,1))*(p(i,1)*w_A+p(j,1)*w_B)/72;
396   end
397 end
398 %
399 %=========================================================================
400
401
402 function Wn=calculo_W_n(p,n)
403 %-------------------------------------------------------------------------+
404 %
405 % Descripción función:
406 %   Calcular el momento estático de un polígono simple/compuesto
407 % respecto eje n
408 %
409 % Inputs:
410 %   p: polígono simple  compuesto
411 %   n: eje respecto al cual calculamos momento estático
412 %      <1: eje de abscisas | 2: eje ordenado>
413 %
414 % Outputs:
415 %   wn: momento estático de p respecto el eje n
416 %
417 % Dependencias:
418 %   propSeccion.W_n
419 %
420 %-------------------------------------------------------------------------+
421   if iscell(p)
422     Wn=0;
423     for i=1:length(p)
424       Wn=Wn+propSeccion.W_n(p{i},n);
425     end
426   else
427     Wn=propSeccion.W_n(p,n);
428   end
429 end
```

```
%
%=========================================================================
```

```
function Wn=W_n(p,n)
%-------------------------------------------------------------------------+
%
% Descripción función:
%   Calcular el momento estático de un polígono simple respecto un eje
%
% Inputs:
%   p: polígono simple
%   n: eje respecto al cual se calcula el momento estático
%      <1: eje de abscisas | 2: eje ordenado>
%
% Outputs:
%   Wn: momento estático de p respecto del eje n
%
%-------------------------------------------------------------------------+
  Wn=0;
  if n==1
    m=2;
  else
    m=1;
  end
  nver=length(p);
  for i=1:nver
    j=i+1;
    if i==nver
      j=1;
    end
    Wn=Wn+(-1)^n*(p(j,n)-p(i,n))*(p(i,m)^2+p(i,m)*p(j,m)+p(j,m)^2)/6;
  end
end
%
%=========================================================================
```

```
end
```

### B.3.6. *metNum*

```
classdef metNum
methods(Static)

```

```
%*****************************************************************%
%                                                                 %
% metNum: módulo para la resolución de problemas de Métodos Numéricos %
%         (Resolución de ecuaciones no lineales)                  %
%                                                                 %
% Funciones que contiene:                                         %
%   approot(): hallar el intervalo que contiene una raíz de f(x)=0 %
%   zero(): método de Brent                                       %
%   secant(): método de la secante                                %
%                                                                 %
%*****************************************************************%


function [R,x]=approot(f,a,b,n,epsilon)
%-----------------------------------------------------------------+
%
% R=approot(f,a,b,n,epsilon)
%
% Descripción función:
%   Función que estima la posición de una raíz dada una función y un
%   intervalo que contiene la raíz de dicha función
%
% Inputs:
%   f: función de la que queremos encontrar el intervalo
%   a,b: valores que encierran el intervalo donde se estima que existe
%        una raíz
%   n: número de puntos del intervalo donde se evaluará la función
%   epsilon: tolerancia
%
% Outputs:
%   R: raíz estimada
%   x: el intervalo que contiene una raíz
%
% Fuente:
%   Numerical Methods Using MATLAB. 3ra Edición, Cap.2 - John H. Mathews,
%   Kurtis D. Fink
%
%-----------------------------------------------------------------+

  R=[];
  x=[];
  X=linspace(a,b,n);
  fa=f(X(1));
  for i=2:n
    fb=f(X(i));
```

```
49        if  fb*fa<=0
50          R=(X(i-1)+X(i))/2;
51          x=[X(i-1),X(i)];
52          fa=fb;
53          return
54        end
55      end
56    end
57    %
58    %=============================================================================
59
60
61    function value = zero ( a, b, machep, t, f )
62    %*****************************************************************************80
63    %
64    %% ZERO seeks the root of a function F(X) in an interval [A,B].
65    %
66    %  Discussion:
67    %
68    %    The interval [A,B] must be a change of sign interval for F.
69    %    That is, F(A) and F(B) must be of opposite signs.  Then
70    %    assuming that F is continuous implies the existence of at least
71    %    one value C between A and B for which F(C) = 0.
72    %
73    %    The location of the zero is determined to within an accuracy
74    %    of 6 * MACHEPS * abs ( C ) + 2 * T.
75    %
76    %    Thanks to Thomas Secretin for pointing out a transcription error in the
77    %    setting of the value of P, 11 February 2013.
78    %
79    %  Licensing:
80    %
81    %    This code is distributed under the GNU LGPL license.
82    %
83    %  Modified:
84    %
85    %    11 February 2013
86    %
87    %  Author:
88    %
89    %    Original FORTRAN77 version by Richard Brent
90    %    MATLAB version by John Burkardt
91    %
92    %  Reference:
93    %
```

```
94  %    Richard Brent,
95  %    Algorithms for Minimization Without Derivatives,
96  %    Dover, 2002,
97  %    ISBN: 0-486-41998-3,
98  %    LC: QA402.5.B74.
99  %
100 %  Parameters:
101 %
102 %    Input, real A, B, the endpoints of the change of sign interval.
103 %
104 %    Input, real MACHEP, an estimate for the relative machine
105 %    precision.
106 %
107 %    Input, real T, a positive error tolerance.
108 %
109 %    Input, real value = F ( x ), the name of a user-supplied
110 %    function which evaluates the function whose zero is being sought.
111 %
112 %    Output, real VALUE, the estimated value of a zero of
113 %    the function F.
114 %
115
116 %
117 %  Make local copies of A and B.
118 %
119   sa = a;
120   sb = b;
121   fa = f ( sa );
122   fb = f ( sb );
123
124   c = sa;
125   fc = fa;
126   e = sb - sa;
127   d = e;
128
129   while ( 1 )
130
131     if ( abs ( fc ) < abs ( fb ) )
132
133       sa = sb;
134       sb = c;
135       c = sa;
136       fa = fb;
137       fb = fc;
138       fc = fa;
```

```
139
140     end
141
142     tol = 2.0 * machep * abs ( sb ) + t;
143     m = 0.5 * ( c - sb );
144
145     if ( abs ( m ) <= tol || fb == 0.0 )
146       break
147     end
148
149     if ( abs ( e ) < tol || abs ( fa ) <= abs ( fb ) )
150
151       e = m;
152       d = e;
153
154     else
155
156       s = fb / fa;
157
158       if ( sa == c )
159
160         p = 2.0 * m * s;
161         q = 1.0 - s;
162
163       else
164
165         q = fa / fc;
166         r = fb / fc;
167         p = s * ( 2.0 * m * q * ( q - r ) - ( sb - sa ) * ( r - 1.0 ) );
168         q = ( q - 1.0 ) * ( r - 1.0 ) * ( s - 1.0 );
169
170       end
171
172       if ( 0.0 < p )
173         q = - q;
174       else
175         p = - p;
176       end
177
178       s = e;
179       e = d;
180
181       if ( 2.0 * p < 3.0 * m * q - abs ( tol * q ) && p < abs ( 0.5 * s * q ) )
182         d = p / q;
183       else
```

```
184           e = m;
185           d = e;
186         end
187
188       end
189
190       sa = sb;
191       fa = fb;
192
193       if ( tol < abs ( d ) )
194         sb = sb + d;
195       elseif ( 0.0 < m )
196         sb = sb + tol;
197       else
198         sb = sb - tol;
199       end
200
201       fb = f ( sb );
202
203       if ( ( 0.0 < fb && 0.0 < fc ) || ( fb <= 0.0 && fc <= 0.0 ) )
204         c = sa;
205         fc = fa;
206         e = sb - sa;
207         d = e;
208       end
209
210     end
211
212     value = sb;
213
214     return
215   end
216   %
217   %=========================================================================
218
219
220   function [p,f,nit]=secant(fun,p0,p1,tolerancia)
221   %-------------------------------------------------------------------------+
222   %
223   %  [p,err,nit]=secant(fun,p0,p1,tolerancia)
224   %
225   %  Descripción función:
226   %    Función que calcula la raíz de una función(fun) por medio del
227   %  Método de la secate
228   %
```

```
229 % Inputs:
230 %   fun: función(ha de ser anonymous)
231 %   p0, p1: puntos que pertenecen al dominio de la función
232 %   tolerancia: tolerancia
233 %
234 % Outputs:
235 %   p: raíz de la función(fun)
236 %   f: valor de la función en p
237 %   nit: número de iteraciones
238 %
239 % Se asume:
240 %   - fun: fun es una función anonymous, i.e. es de la forma
241 %          fun=@(x) f(x)
242 %
243 % Fuente:
244 %   Numerical Methods Using MATLAB. 3ra Edición, Cap.2 - John H. Mathews,
245 %   Kurtis D. Fink
246 %
247 %-------------------------------------------------------------------------+
248
249   nitMax=500; %número máximo de iteraciones permitidas
250   delta=tolerancia;
251
252   f0=fun(p0);
253   f1=fun(p1);
254   nit=0;
255
256   if abs(f0)<tolerancia
257     p=p0;
258     f=f0;
259   elseif abs(f0)<tolerancia
260     p=p1;
261     f=f1;
262   else
263     f=-9999;
264     nit=0; %inicializamos variable
265
266     while abs(f)>tolerancia
267       p=p1-f1*(p1-p0)/(f1-f0);
268       f=fun(p);
269
270         p0=p1;
271         f0=f1;
272         p1=p;
273         f1=f;
```

```
274
275 %%%%Evitar errores de redondeo debido a p0 y p1 próximos(lo que es
276 %     lo mismo a f0 y f1 próximos)
277       if abs(p0-p1)<delta
278         p0=p0;
279         f0=f0;
280         p1=p;
281         f1=f;
282       end
283
284       nit=nit+1;
285       if nit>nitMax, error('Solución no converge.'); return, end
286     end
287   end
288 end
289 %
290 %=========================================================================
291
292
293 end %method(Static)
294 end %classdef metNum
```

### B.3.7. *problGeom*

```
1  classdef problGeom
2  methods(Static)
3
4  %*************************************************************************%
5  %                                                                         %
6  % problGeom(): módulo que recoge las funciones que calculan los          %
7  %              problemas geométricos                                      %
8  %   calculo_cambioSistCoord(): cambiar el sistema de referencias de       %
9  %     los vértices de un polígono simple o compuesto                      %
10 %   cambioSistCoord(): cambiar el sistema de referencias de los           %
11 %     vértices de un polígono simple                                      %
12 %   calculo_pc_poly_r(): calcular los puntos de corte entre una recta     %
13 %     y un polígono simple o compuesto                                    %
14 %   pc_poly_r(): calcular los puntos de corte entre una recta y un        %
15 %     polígono simple o compuesto                                         %
16 %   calculo_cortarArea(): determinar la porción de un polígono simple     %
17 %     o compuesto que queda a un lado de una recta                        %
18 %   clipArea(): determinar la porción de un polígono simple o compuesto   %
19 %     que queda a un lado de una recta                                    %
20 %   conjuntoSimpleArrays(): convertir un conjunto de cell en un           %
21 %     conjunto de arrays                                                  %
```

```
22 %                                                                      %
23 %**********************************************************************%
24
25
26 function ptos=calculo_cambioSistCoord(p,dx,dy,tita)
27 %-------------------------------------------------------------------------+
28 %
29 % ptos=calculo_cambioSistCoord(p,dx,dy,tita)
30 %
31 % Descripción función:
32 %   Función que cambia el sistema de referencia de un conjunto
33 %   de puntos(2D). (Contempla el caso de cambiar el sistema de
34 %   referencia de más de un conjunto de puntos, polígonos compuestos)
35 %
36 % Inputs:
37 %   p: conjunto de puntos que se desea trasladar y/o rotar
38 %   dx: traslación del eje x
39 %   dy: traslación del eje y
40 %   tita: ángulo de giro(radianes)
41 %
42 % Outputs:
43 %   ptos: puntos(p) referidos al nuevo sistema de coordenadas
44 %
45 % Dependencias:
46 %   conjuntoSimpleArrays, cambioSistCoord
47 %
48 %-------------------------------------------------------------------------+
49
50    if iscell(p)
51     p=problGeom.conjuntoSimpleArrays(p);
52     for i=1:length(p)
53       ptos{i}=problGeom.cambioSistCoord(p{i},dx,dy,tita);
54     end
55   else
56     ptos=problGeom.cambioSistCoord(p,dx,dy,tita);
57   end
58 end
59 %
60 %=========================================================================
61
62
63
64 function ptos=cambioSistCoord(p,dx,dy,tita)
65 %-------------------------------------------------------------------------+
66 %
```

```
67  %  ptos=cambioSistCoord(p,dx,dy,tita)
68  %
69  %  Descripción función:
70  %    Función que cambia el sistema de referencia de un conjunto
71  %    de puntos(2D) recogidos en un array.
72  %
73  %  Inputs:
74  %    p: conjunto de puntos que se desea trasladar y/o rotar
75  %    dx: traslación del eje x
76  %    dy: traslación del eje y
77  %    tita: ángulo de giro(radianes)
78  %
79  %  Outputs:
80  %    ptos: puntos(p) referidos al nuevo sistema de coordenadas
81  %
82  %------------------------------------------------------------------------+
83    ptos=[];%inicializamos variable
84    for i=1:length(p)
85      %Rotamos(tita) y trasladamos(dx,dy)
86      pto=[[cos(tita),sin(tita)]; ...
87      [-sin(tita),cos(tita)]]*p(i,:)'+[dx;dy];%(2x1)
88      pto=(pto)';%1x2
89      %incorporamos este punto(pto) en la lista de puntos(ptos)
90      ptos=[ptos;pto];
91    end
92  end
93  %
94  %========================================================================
95
96
97  function pc=calculo_pc_poly_r(p,r)
98  %------------------------------------------------------------------------+
99  %
100 %  p_corte=pc_poly_r(p,r)
101 %
102 %  Descripción función:
103 %    Función que calcula los puntos de corte entre una recta
104 %    y un polígono. (Contempla el caso de calcular puntos de
105 %    corte de una sección compuesta por diferentes porciones)
106 %
107 %  Inputs:
108 %    p: vértices que definen el polígono
109 %    r: recta de la forma c*y=a*x+b. Array 1x3.
110 %       Siendo:
111 %         [r(1),r(2),r(3)]=[a,b,c]
```

```
112 %
113 %  Outputs:
114 %    p_corte: puntos de corte entre el polígono(p) y la recta(r)
115 %
116 %  Dependencias:
117 %    conjuntoSimpleArrays, pc_poly_r
118 %
119 %-------------------------------------------------------------------------+
120
121   if iscell(p)
122     p=problGeom.conjuntoSimpleArrays(p);
123     pc=[];
124     for i=1:length(p)
125       pc_i=problGeom.pc_poly_r(p{i},r);
126       pc=[pc;pc_i];
127     end
128   else
129     pc=problGeom.pc_poly_r(p,r);
130   end
131 end
132 %
133 %=========================================================================
134
135
136 function p_corte=pc_poly_r(p,r)
137 %-------------------------------------------------------------------------+
138 %
139 %  p_corte=pc_poly_r(p,r)
140 %
141 %  Descripción función:
142 %    Función que calcula los puntos de corte entre una recta
143 %    y un polígono.
144 %
145 %  Inputs:
146 %    p: vértices que definen el polígono
147 %    r: recta de la forma c*y=a*x+b. Array 1x3.
148 %       Siendo:
149 %         [r(1),r(2),r(3)]=[a,b,c]
150 %
151 %  Outputs:
152 %    p_corte: puntos de corte entre el polígono(p) y la recta(r)
153 %
154 %-------------------------------------------------------------------------+
155
156   if size(p,1)<2
```

```
157     error('Para el cálculo de los puntos de corte al menos se requiere una
            arista(dos puntos)')
158     return
159   end
160
161   if size(p,1)>2 && p(1,1)==p(end,1) && p(1,2)==p(end,2)
162     p=p(1:end-1,:);
163   end
164
165   tolerancia=1E-6;
166
167   a=r(1);
168   b=r(2);
169   c=r(3);
170   nver=length(p);
171   p_corte=[];
172
173   for i=1:nver
174     if i==nver
175       j=1;
176     else
177       j=i+1;
178     end
179
180     corta_rAndli=((-a*p(i,1)+c*p(i,2))>b && (-a*p(j,1)+c*p(j,2))<b) || ...
181                  ((-a*p(i,1)+c*p(i,2))<b && (-a*p(j,1)+c*p(j,2))>b);
182
183     if (-a*p(i,1)+c*p(i,2))==b
184       %Caso en el que li y r son paralelas
185       p_corte=[p_corte;p(i,:)];
186
187     elseif corta_rAndli
188
189       if p(i,1)==p(j,1)
190       %Si x_i==x_i+1 --> se trata ec. recta x=x_i
191         ci=0;
192         ai=1;
193         bi=-p(i,1);
194       else
195         ci=1;
196         ai=(p(j,2)-p(i,2))/(p(j,1)-p(i,1));
197         bi=p(i,2)-ai*p(i,1);
198       end
199
200       if abs(-a*ci+c*ai)>tolerancia
```

```
201             %Resolvemos por Cramer
202             pc(1)=(b*ci-bi*c)/(-a*ci+c*ai);
203             pc(2)=(-a*bi+b*ai)/(-a*ci+c*ai);
204             p_corte=[p_corte;pc];
205           end
206         end
207
208       end
209     end
210     %
211     %=========================================================================
212
213
214     function A=calculo_cortarArea(p,r,condic)
215     %-------------------------------------------------------------------------+
216     %
217     % A=calculo_cortarArea(p,r,condic)
218     %
219     % Descripción función:
220     %   Función que determina el polígono resultante de
221     %   dividir un área por medio de una recta. El polígono resultante
222     %   es aquel que cae a un lado de la recta(especificado con condic)
223     %   (Contempla el caso de calcular los polígonos que cumplen
224     %   la condición  de una sección compuesta por diferentes porciones)
225     %
226     % Inputs:
227     %   p: vértices que definen el polígono(array) o conjunto de
228     %      poligonos(cell)
229     %   r: recta de la forma. c*y=a*x+b. Array 1x3.
230     %      Siendo: r=[a,b,c]
231     %   condic: condición que determina que lado de la recta escogemos.
232     %           Siendo:
233     %              condic==1, si c*y-a*x>=b
234     %              condic==2, si c*y-a*x<=b
235     %
236     % Outputs:
237     %   Ai: polígono o conjunto de polígonos que cumplen la condición
238     %      esto es, forma parte de p y cae a un lado de r(condic)
239     %
240     % Se asume:
241     %  Orden de la sucesión de vértices de p en s.a.r. si arista forma
242     % parte de arista exterior, s.c.a.r. si la arista forma parte de un
243     % agujero
244     %
245     % Dependencias:
```

```
246 %    conjuntoSimpleArrays, clipArea
247 %-------------------------------------------------------------------------+
248   if iscell(p)
249     p=problGeom.conjuntoSimpleArrays(p);
250     pc=problGeom.calculo_pc_poly_r(p,r);
251     if pc(1,1)~=pc(2,1)
252       %caso en el que r no es vertical
253       C1=pc(pc(:,1)==min(pc(:,1)),:); %pto de corte con la componente
254                                       %horizontal mínima
255       C2=pc(pc(:,1)==max(pc(:,1)),:); %pto de corte con la componente
256                                       %horizontal máxima
257     else
258       %En el caso de que r sea vertical
259       C1=pc(pc(:,2)==max(pc(:,2)),:); %pto de corte con la componente
260                                       %vertical máxima
261       C2=pc(pc(:,2)==min(pc(:,2)),:); %pto de corte con la componente
262                                       %vertical mínima
263     end
264
265     n=(C2-C1)/norm(C2-C1); %dirección de r
266
267     for i=1:length(p)
268       pc_i=problGeom.pc_poly_r(p{i},r);
269       if length(pc_i)==0
270         ni=(p{i}(1,:)-C1)/norm(p{i}(1,:)-C1);
271         %Si no hay punto de corte pero un punto cumple condición. Entonces
              todos cumplen(caso en el que no porción forme parte de una sección,
              cumple condición pero no corta la recta)
272         if (-1)^(condic-1)*det([n;ni])>0,A{i}=p{i}, end
273       else
274         A{i}=problGeom.clipArea(p{i},r,condic);
275       end
276     end
277     A=problGeom.conjuntoSimpleArrays(A);
278     A=A(~cellfun('isempty',A)); %como la salida de clipArea puede ser vacío
          para cierto p{i}, hay que quitar esos elementos
279   else
280     A=problGeom.clipArea(p,r,condic);
281   end
282 end
283 %
284 %=========================================================================
285
286
287
```

```
288 function Ai=clipArea(vertexArray,r,condic)
289 %-------------------------------------------------------------------------+
290 %
291 % Ai=clipArea(vertexArray,r,condic)
292 %
293 % Descripción función:
294 %   Función que determina el poligono resultante de
295 %   dividir un área por medio de una recta. El poligono resultante
296 %   es aquel que cae a un lado de la recta(especificado con condic)
297 %
298 % Inputs:
299 %   vertexArray: vértices que definen el polígono
300 %   r: recta de la forma. c*y=a*x+b. Array 1x3.
301 %      Siendo: r=[a,b,c]
302 %   condic: condición que determina que lado de la recta escogemos.
303 %           Siendo:
304 %              condic==1, si c*y-a*x>=b
305 %              condic==2, si c*y-a*x<=b
306 %
307 % Outputs:
308 %   Ai: polígono o conjunto de polígonos que cumplen la condición
309 %      esto es, forma parte de vertexArray y cae a un lado de r(condic)
310 %
311 % Se asume:
312 %  Orden de la sucesión de vértices de p en s.a.r. si arista forma
313 % parte de arista exterior, s.c.a.r. si la arista forma parte de un
314 % agujero
315 %
316 % Referecias:
317 %   Algoritmo basado en la función clipArea() de:
318 %   Louis H. Turcotte and Howard B. Wilson. Computer Applications
319 % in Mechanics of Materials Using MATLAB. Prentice Hall, 1998.
320 %
321 % Válido para polígonos convexos, cóncavos y con huecos(convexos)
322 %
323 %-------------------------------------------------------------------------+
324
325   tol=eps*1E4; % Valor que consideramos como cero
326
327   % Garantizar que el primer vértice y el último no es el mismo
328   if abs(vertexArray(1,1)-vertexArray(end,1))<tol  && abs(vertexArray(1,2)-
        vertexArray(end,2))<tol
329     vertexArray=vertexArray(1:end-1,:);
330   end
331
```

```
332   a=r(1);
333   b=r(2);
334   c=r(3);
335
336   nvert=size(vertexArray,1);
337
338   %B: matriz booleana que indica qué puntos cumplen condición
339   %Ver qué vértices del polígono cumplen la condición
340   B=zeros(1,nvert+1);
341   vi_In_r=0;
342   for i=1:nvert
343     if abs(c*vertexArray(i,2)-a*vertexArray(i,1)-b)<tol
344       B(i)=0;
345       vi_In_r=vi_In_r+1;
346     else
347     if condic==1
348       if c*vertexArray(i,2)-a*vertexArray(i,1)>b
349         B(i)=1;
350       end
351     else %condic==2
352       if c*vertexArray(i,2)-a*vertexArray(i,1)<b
353         B(i)=1;
354       end
355     end
356     end
357   end%for
358
359   B(end)=B(1);
360
361   %Calcular número de puntos de corte (npc)
362   npc=0;
363   for i=1:nvert
364     existePc=(B(i)==0 && B(i+1)==1) || (B(i)==1 && B(i+1)==0);
365     if existePc, npc=npc+1; end
366   end
367
368   %Calcular número de puntos que cumplen la condición y no están
369   %  sobre recta r (nB1)
370   nB1=length(find(B(1:end-1)==1));
371
372   if all(B==0); Ai=[]; return, end %Caso en el que ningún vértice
373                                    %cumple la condición, Ai=[]
374   if all(B==1) && vi_In_r==0, Ai=[]; return; end %Si todos los puntos
375                              %cumplen condición, pero ninguno  se
376                              %encuentra sobre recta r, Ai=[]
```

```
377   if nB1+vi_In_r==nvert, Ai=vertexArray; return, end %Caso en el que
378                 %todos los puntos cumplen condición y r tg a vertexArray
379
380   %Calcular array R
381
382   %inicializamos variables
383   nR=nB1+npc;
384   R=nan(nR,2);
385   pcArray=nan(npc,2);
386   indexR=1;
387   i_pc=1;
388   nOutIn=str2num(sprintf('%.0f',npc/2))+1;
389   index_outInArray=nan(nOutIn,1);
390   index_p0outInArray=nan(nOutIn,1);
391   index_v0outInArray=nan(nOutIn,1);
392   index_outIn=1;
393
394   for i=1:nvert
395     if i==nvert, j=1; else, j=i+1; end
396     if B(i)==0
397       if B(j)==1
398         if abs(c*vertexArray(j,2)-a*vertexArray(j,1)-b)<tol
399           pc=vertexArray(j,:);
400         else
401
402           if vertexArray(i,1)==vertexArray(j,1)
403       %Si x_i==x_i+1 --> se trata ec. recta x=x_i
404             ci=0;
405             ai=1;
406             bi=-vertexArray(i,1);
407           else
408             ci=1;
409             ai=(vertexArray(j,2)-vertexArray(i,2))/(vertexArray(j,1)-vertexArray
                    (i,1));
410             bi=vertexArray(i,2)-ai*vertexArray(i,1);
411           end
412           %Resolvemos por Cramer
413           pc(1)=(b*ci-bi*c)/(-a*ci+c*ai);
414           pc(2)=(-a*bi+b*ai)/(-a*ci+c*ai);
415
416         end
417         R(indexR,1)=pc(1,1);
418         R(indexR,2)=pc(1,2);
419         index_outInArray(index_outIn)=indexR;
420         index_v0outInArray(index_outIn)=i;
```

```
421         index_outIn=index_outIn+1;
422         indexR=indexR+1;
423 
424         pcArray(i_pc,1)=pc(1,1);
425         pcArray(i_pc,2)=pc(1,2);
426         i_pc=i_pc+1;
427       end
428 
429     else %if B(i)==1
430       R(indexR,1)=vertexArray(i,1);
431       R(indexR,2)=vertexArray(i,2);
432       indexR=indexR+1;
433       if B(j)==0
434         if abs(c*vertexArray(i,2)-a*vertexArray(i,1)-b)<tol
435           pc=vertexArray(i,:);
436         else
437 
438           if vertexArray(i,1)==vertexArray(j,1)
439       %Si x_i==x_i+1 --> se trata ec. recta x=x_i
440             ci=0;
441             ai=1;
442             bi=-vertexArray(i,1);
443           else
444             ci=1;
445             ai=(vertexArray(j,2)-vertexArray(i,2))/(vertexArray(j,1)-vertexArray
                    (i,1));
446             bi=vertexArray(i,2)-ai*vertexArray(i,1);
447           end
448           %Resolvemos por Cramer
449           pc(1)=(b*ci-bi*c)/(-a*ci+c*ai);
450           pc(2)=(-a*bi+b*ai)/(-a*ci+c*ai);
451         end %caso en que B(i)==0 B(i+1)==1
452         R(indexR,1)=pc(1,1);
453         R(indexR,2)=pc(1,2);
454         indexR=indexR+1;
455 
456         pcArray(i_pc,1)=pc(1,1);
457         pcArray(i_pc,2)=pc(1,2);
458         i_pc=i_pc+1;
459       end
460     end
461   end
462 
463   if isnan(index_outInArray(end,1)), index_outInArray=index_outInArray(1:end
        -1,:); end
```

```
464   if isnan(R(end,1)), R=R(1:end-1,:); end
465   nR=size(R,1);
466
467   %Inicializamos
468   i_Ai=1;
469   i_0=1;
470
471   %Generar los polígonos
472   for i=1:length(index_outInArray)
473     if index_outInArray(i)~=1 && index_outInArray(i)<=nR
474     pcArray_aux=setdiff(pcArray,[R(index_outInArray(i)-1,:);R(index_outInArray(i
          ),:)],'rows');
475     if c==0
476       %Si r es una recta vertical
477       ymax=max(R(index_outInArray(i)-1,2),R(index_outInArray(i),2));
478       ymin=min(R(index_outInArray(i)-1,2),R(index_outInArray(i),2));
479
480       if any((ymax-pcArray_aux(:,2))>tol & (pcArray_aux(:,2)-ymin)>tol)
481         isAny_pcInLi=1;
482       else
483         isAny_pcInLi=0;
484       end
485
486     else
487      xmax=max(R(index_outInArray(i)-1,1),R(index_outInArray(i),1));
488      xmin=min(R(index_outInArray(i)-1,1),R(index_outInArray(i),1));
489      if any((xmax-pcArray_aux(:,1))>tol & (pcArray_aux(:,1)-xmin)>tol)
490        isAny_pcInLi=1;
491      else
492        isAny_pcInLi=0;
493      end
494     end
495     p1=R(index_outInArray(i)-1,:);
496     p2=vertexArray(index_v0outInArray(i),:);
497     p3=R(index_outInArray(i),:);
498     a1=p2(1)*p3(2)-p3(1)*p2(2);
499     a2=p3(1)*p1(2)-p1(1)*p3(2);
500     a3=p1(1)*p2(2)-p2(1)*p1(2);
501
502     if (a1+a2+a3)<tol*10
503       if (abs(p1(1)-p2(1))<tol && abs(p1(2)-p2(2))<tol && abs(p1(1)-p3(1))<tol
            && abs(p1(2)-p3(2))<tol) && index_outInArray(i)>1 && index_outInArray(i
            )<nR
504         p1=R(index_outInArray(i)-2,:);
505         p3=R(index_outInArray(i)+1,:);
```

```
506           a1=p2(1)*p3(2)-p3(1)*p2(2);
507           a2=p3(1)*p1(2)-p1(1)*p3(2);
508           a3=p1(1)*p2(2)-p2(1)*p1(2);
509
510         elseif abs(p2(1)-p3(1))<tol && abs(p2(2)-p3(2))<tol && index_outInArray(i)
              <nR
511           p3=R(index_outInArray(i)+1,:);
512           a1=p2(1)*p3(2)-p3(1)*p2(2);
513           a2=p3(1)*p1(2)-p1(1)*p3(2);
514           a3=p1(1)*p2(2)-p2(1)*p1(2);
515
516         end
517       end
518
519       if (isAny_pcInLi || a1+a2+a3<=0) && ~( abs(p1(1)-vertexArray(1,1))<tol &&
            abs(p1(2)-vertexArray(1,2))<tol && abs(p1(1)-p2(1))<tol && abs(p1(2)-p2
            (2))<tol && abs(p1(1)-p3(1))<tol && abs(p1(2)-p3(2))<tol)
520         n=index_outInArray(i)-1-i_0;
521         Ai{i_Ai}=nan(n,2);
522         m=1;
523         for j=i_0:index_outInArray(i)-1
524           Ai{i_Ai}(m,1)=R(j,1);
525           Ai{i_Ai}(m,2)=R(j,2);
526           m=m+1;
527         end
528
529         i_0=index_outInArray(i);
530         i_Ai=i_Ai+1;
531
532       end
533
534   end
535   end
536
537   Ai{i_Ai}=R(i_0:end,:);
538   %Problema polígonos con huecos (puntos repetidos)
539   if size(Ai{end},1)==1 && length(Ai)>3 && abs(Ai{end}(1,1)-Ai{end-1}(end,1))<
        tol && abs(Ai{end}(1,2)-Ai{end-1}(end,2))<tol, Ai=Ai(1,1:end-1); end
540
541   if ismember(i_0,index_outInArray) && i_0~=1
542     pcArray_aux=setdiff(pcArray,[Ai{end}(1,:);Ai{1}(end,:)],'rows');
543     if c==0
544       %Si r es una recta vertical
545       ymax=max(Ai{end}(1,2),Ai{1}(end,2));
546       ymin=min(Ai{end}(1,2),Ai{1}(end,2));
```

```
547
548         if any((ymax-pcArray_aux(:,2))>tol & (pcArray_aux(:,2)-ymin)>tol)
549           isAny_pcInLi=1;
550         else
551           isAny_pcInLi=0;
552         end
553
554       else
555        xmax=max(Ai{end}(1,1),Ai{1}(end,1));
556        xmin=min(Ai{end}(1,1),Ai{1}(end,1));
557
558        if any((xmax-pcArray_aux(:,1))>tol & (pcArray_aux(:,1)-xmin)>tol)
559          isAny_pcInLi=1;
560        else
561          isAny_pcInLi=0;
562        end
563       end
564
565       p1=Ai{1}(end-1,:);
566       p2=Ai{1}(end,:);
567       p3=Ai{end}(1,:);
568       a1=p2(1)*p3(2)-p3(1)*p2(2);
569       a2=p3(1)*p1(2)-p1(1)*p3(2);
570       a3=p1(1)*p2(2)-p2(1)*p1(2);
571
572    if (~isAny_pcInLi && a1+a2+a3>0) || (size(Ai{1},1)==2 && abs(a1+a2+a3)<tol*10
       && size(Ai{end},1)==1)
573         Ai{1}=[Ai{1};Ai{end}];
574         Ai=Ai(1,1:end-1);
575
576         if abs(Ai{1}(end,1)-Ai{1}(1,1))<tol && abs(Ai{1}(end,2)-Ai{1}(1,2))<tol,
              Ai{1}=Ai{1}(1:end-1,:);end
577       end
578     end
579
580     %Si Ai es un polígono simple, cell --> array
581     if length(Ai)==1, Ai=Ai{1}; end
582  end
583  %
584  %=========================================================================
585
586
587  function pSimple=conjuntoSimpleArrays(p)
588  %-------------------------------------------------------------------------+
589  %
```

```
590 %  pSimple=conjuntoSimpleArrays(p)
591 %
592 %  Descripción función:
593 %    Función que convierte un conjunto de conjuntos(un cell con elementos
594 %    cell) en un conjunto de arrays(un cell con todos sus
595 %    componentes arrays)
596 %
597 %  Inputs:
598 %    p: conjunto de conjuntos(estructura cell)
599 %
600 %  Outputs:
601 %    pSimple: conjunto de arrays
602 %
603 %-------------------------------------------------------------------------+
604     for i=1:length(p)
605       b=0;
606       if iscell(p{i}), b=1; break, end
607     end
608     if b==1
609       p=[p{:}];
610       %Función recursiva
611       pSimple=problGeom.conjuntoSimpleArrays(p); %recursivo
612     else
613       pSimple=p;
614     end
615 end
616 %
617 %=========================================================================
618
619 end
620 end
```

# Bibliografía


[1] *Software Architecture and Design.* Tutorials Point.

[2] John Burkardt. Brent's method, matlab version. `http://people.sc.fsu.edu/~jburkardt/m_src/brent/zero.m`.

[3] Fernández Díaz-Munío, Rafael . *Plasticidad Abreviada 2001.* E.T.S.INGENIEROS CAMINOS C.P.MADRID, 1996.

[4] Inkscape Project. Inkscape. `https://inkscape.org`.

[5] John H. Mathews and Kurtis D. Fink. *Numerical Methods Using MATLAB.* Prentice Hall, 1999.

[6] John W. Eaton, David Bateman, Soren Hauberg, and Rik Wehbring. *GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations.* 2015.

[7] Johnson, Richard. *MATLAB Style Guidelines 2.0.* 2014.

[8] Yong-Ming Li. Centroid, area, and moments of inertia. 1997.

[9] Louis H. Turcotte and Howard B. Wilson. *Computer Applications in Mechanics of Materials Using MATLAB.* Prentice Hall, 1998.

[10] Martha A. Brandstad and John C. Cherniavsky. Validation, verification, and testing for the individual programmer. 1980.

[11] Miguel Cervera Ruiz and Elena Blanco Díaz. *Resistencia de Materiales.* CIMNE, 2015.

[12] Park, J.S. *MATLAB GUI (Graphical User Interface) Tutorial for Beginners.* University of Incheon.

[13] Perelli Botello, Jorge . *Resistencia de Materiales, Elasticidad y Plasticidad. Parte 2 Plasticidad.* RETINEO, 2013.

[14] Pat Scott. *Practical Numerical Methods in Physics and Astronomy.*

[15] W. Richards Adrion, Martha A. Brandstad, and John C. Cherniavsky. Validation, verification, and testing of computer software. 1982.

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, 2002.