MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES
UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Desarrollo de una interfaz gráfica de usuario con GUIDE (MATLAB) para la comprobación de secciones de hormigón armado

Autor: D. Miguel MEDINA CHICO
Tutor: D. Mariano CACHO PÉREZ

Valladolid, junio, 2019

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES
UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Desarrollo de una interfaz gráfica de usuario con GUIDE (MATLAB) para la comprobación de secciones de hormigón armado

Autor: D. Miguel MEDINA CHICO
Tutor: D. Mariano CACHO PÉREZ

Valladolid, junio, 2019

RESUMEN

El presente Trabajo Fin de Máster consiste en el desarrollo y ejecución de una aplicación, a través del software MATLAB, por medio de la cual se conseguirá encontrar solución al problema de comprobación para una sección transversal de hormigón armado.

Dicho trabajo está dividido en dos partes. La primera basada en el desarrollo de un script en código MATLAB que resuelva el problema de comprobación para secciones de hormigón armado, atendiendo al método de los elementos finitos. La segunda parte consta de la creación de una aplicación ejecutable, que permita mediante la introducción de los datos de entrada, generar la solución de manera rápida y visual.

PALABRAS CLAVE

FLEXIÓN, HORMIGÓN ARMADO, SECCIÓN, COMPROBACIÓN, GUIDE

ABSTRACT

The present Final Master Project consists of the development and execution of an application, through the MATLAB software, by means of which it will be possible to find a solution to the testing problem for a reinforced concrete cross section.

This work is divided into two parts. The first one based on the development of a MATLAB code script that solves the problem of checking for reinforced concrete sections, according to the finite element method. The second part consists of the creation of an executable application, which allows by means of the introduction of the input data, to generate the solution in a fast and visual way.

KEYWORDS

BENDING, REINFORCED CONCRETE, SECTION, TESTING, GUIDE

AGRADECIMIENTOS

Son muchas las personas que han contribuido al proceso y conclusión de este TFM. En primer lugar, quiero agradecer a Mariano Cacho Pérez, tutor de este Trabajo Fin de Máster, por su apoyo y esfuerzo durante la realización del mismo.

También agradezco a la Universidad de Valladolid, lugar donde me he formado como ingeniero y donde he recibido enseñanzas y apoyo de todo tipo.

Por último, me gustaría agradecer a mi familia y amigos, ya que ellos son los que han visto la evolución de este trabajo día a día y me han dado fuerzas para poder finalizarlo.

INDICE GENERAL

INDICE FIGURAS

INDICE DE TABLAS

Capítulo 1

INTRODUCCIÓN

1. INTRODUCCIÓN

1.1.Introducción

Desde hace más de 100 años, el campo de las estructuras ha sido entregado casi al 100% a dos materiales ampliamente conocidos: el hormigón y el acero. Estos dos materiales, han sido objeto de estudios y teorías durante muchos años, por su gran utilización y aplicación en el campo de la ingeniería estructural.

Debido a las características mecánicas y físicas que presentan estos materiales, hasta hace unos años, han sido utilizados en distintos tipos de aplicación. Principalmente, el acero fue utilizado en estructuras más esbeltas y ligeras, mientras que el hormigón se consideraba más apto para proyectar construcciones de mayor envergadura. Aunque la situación ha cambiado mucho en los últimos años debido a la utilización de materiales nuevos, más resistentes y conocidos, y también, gracias al desarrollo de programas y herramientas de cálculo más precisas y potentes.

En la actualidad, debido a los costes de los recursos: materiales, mano de obra, etc. Resulta indispensable tener conocimientos en simulación de estructuras (de cualquier ámbito) para, entre otras cosas, poder reducir tiempos y costes antes siquiera, de ponerse a realizar cualquier obra u oferta para un proyecto. De esta manera, se tendrá una idea más acertada de si la solución estructural propuesta podrá resistir ciertas solicitaciones, la respuesta de la estructura en tensiones y desplazamientos, los factores de seguridad, si es viable, etc.

Para resolver dicho planteamiento, existe hoy en día multitud de métodos de cálculo y simulación (conocimientos). Pero resolver estos problemas manualmente, implicaría más tiempo que realizar maquetas y simulaciones. En este punto se encuadra este proyecto, ya que va a tratar sobre el desarrollo de una aplicación con el programa MATLAB, que integre la resolución mediante simulación, del problema de comprobación para vigas o columnas de sección poligonal cualquiera, de hormigón armado sometida a flexión biaxial. Este tipo de problemas, se ha resuelto hasta hace bien poco mediante el empleo de ábacos dimensionales u otros métodos numéricos (método de cargas recíprocas de Bresler, método del contorno de las cargas de la PCA, etc.) y generalmente las soluciones estaban definidas para secciones rectangulares.

1.2.Objetivos del proyecto

Los objetivos de este trabajo son varios:

- Resolver el problema de comprobación, para vigas o columnas de hormigón armado de sección rectangular de sección transversal poligonal con flexión biaxial o esviada.

- Desarrollo de una aplicación ejecutable mediante MATLAB (GUIDE), que implique la resolución del problema de comprobación, en el caso de una sección rectangular de una sección poligonal cualquiera, incluida la circular (n=∞ lados).

- Por último, se espera que este trabajo sirva como recurso de carácter pedagógico en un futuro.

1.3.Estructura del proyecto

Este trabajo está dividido en varios capítulos, en los cuales la información se aporta progresivamente para una mayor comprensión del mismo. En el texto se introducen gran cantidad de imágenes necesarias para el entendimiento del mismo, haciendo muy visual el trabajo.

El capítulo 2, está centrado en los estados límites, se comentarán los dos tipos de estados límite: estados límite últimos (ELU) y estados límites de servicio (ELS).

En el capítulo 3, se van a comentar los materiales que forman el hormigón armado, tanto el hormigón como el acero, se explicarán sus principales propiedades físicas y mecánicas. Posteriormente, se hablará del hormigón armado como material compuesto y se desarrollará el análisis de la sección a flexión. Se verán las hipótesis fundamentales a nivel de sección, los dominios de deformación en ELU de agotamiento y, por último, los tipos de flexión a los que se puede ver sometida una sección, como son flexión simple, flexión compuesta y flexión compuesta biaxial (sobre la que se cetra este trabajo). En último lugar, se explicará la manera de resolver el problema de comprobación y el método utilizado para ello.

El capítulo 4 sirve para desarrollar la manera en que se han ido resolviendo los problemas de comprobación para los dos tipos de secciones consideradas (rectangular y poligonal de n lados) en el software MATLAB para, por último explicar cómo se ha ido realizando la App "FLEXIÓN BOX" a través de MATLAB, con la ayuda de la herramienta que contiene el mismo programa, denominada GUI.

1.4.Contenido adicional

Además del presente escrito, se adjunta:

- El script con el que se ha generado la App.

Capítulo 2

MATERIALES

2. MATERIALES

En este punto, se comentarán primeramente las características mecánicas de los componentes del material hormigón armado, para después hablar del hormigón armado como material compuesto.

2.1 HORMIGÓN

Se denominan conglomerantes, hidráulicos aquellos productos que amasados con agua, fraguan y endurecen manteniéndose estables e inalterables en contacto posterior con dicho líquido. De los conglomerantes hidráulicos, el cemento es el más famoso por sus propiedades físicas, mecánicas y económicas. A modo de información, los diferentes tipos de cementos que hay, son los siguientes:

- Cementos Portland
- Cementos siderúrgicos
- Puzolánicos
- Sobresulfatados
- Aluminosos
- ...

El hormigón es una mezcla que está formada por cemento, agua, arena, grava y, al que también se le puede añadir ocasionalmente, algunos productos adicionales que confieren al hormigón unas características especiales como pueden ser: acelerar o retardar el fraguado, aumentar su fluidez, dar un color determinado, producir impermeabilidad, etc. Al conjunto de estos productos que se incorporan a la mezcla se les denomina aditivos, reservándose el término de adiciones para aquellos elementos que se añaden al cemento en la fase final de su elaboración (Jiménez Montoya, García Meseguer, & Morán Cabré, 1987).

2.1.1 Dosificación del hormigón

Habitualmente, el hormigón de una estructura se define en función a tres parámetros: resistencia, consistencia y tamaño máximo del árido.

- Tamaño máximo del árido: Cuando mayor sea el tamaño del árido utilizado, menor cantidad de agua habrá que utilizar para conseguir la consistencia requerida, debido a que la superficie a mojar de los áridos será más pequeña. Por este motivo, se puede reducir la cantidad de cemento (abaratar costes) obteniendo un hormigón con la misma resistencia y más económico.
  Por otro lado, hay que limitar el tamaño del árido porque con áridos demasiado gruesos, disminuye en exceso la superficie adherente, creando discontinuidades importantes dentro del hormigón armado.
- Consistencia: Se fija para garantizar que el hormigón podrá rellenar eficazmente el volumen nominal de la pieza, sin presentar un exceso de agua que daría como resultado una reducción de su resistencia y durabilidad. La consistencia se mide mediante el Cono de Abrams (ver figura) y se clasifica de la siguiente manera:

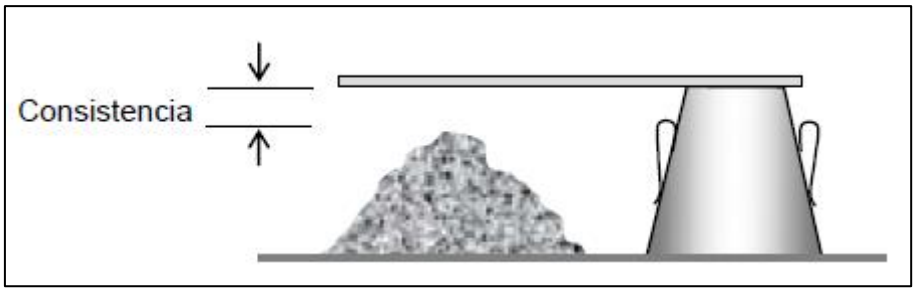

Figura 1. Cono de Abrams

Consistencia	Asiento (cm)	Tolerancia (cm)
Seca	0-2	0
Plástica	3-5	±1
Blanda	6-9	±1
Fluida	10-15	±2

Tabla 1. Valores de consistencia

- Resistencia a compresión: Es la característica fundamental del hormigón. El término "resistencia del hormigón" no tiene sentido, en sí mismo. La resistencia de un mismo hormigón depende del tipo y dimensiones de la probeta empleada, de su edad y condiciones de conservación. Debido a esto, se introduce el concepto de *resistencia característica* (f_{ck}) del hormigón, que es aquél valor que tiene un 95% de nivel de confianza (cualquier probeta ensayada tiene una probabilidad del 95% de superar el valor de la resistencia característica).

  Para conseguirse un nivel de confianza del 95%, el hormigón debe dosificarse de manera que en los ensayos llevados a cabos previos a la ejecución de una obra, la resistencia media (f_{cm}) sea superior a la resistencia característica (f_{ck}).

2.1.2 Propiedades del hormigón fresco

El hormigón fresco se considera un material heterogéneo en el que coexisten las fases: sólida (áridos y cemento), líquida (agua) y gas (aire). Las características más importantes del hormigón fresco, se consideran las siguientes (Jiménez Montoya, García Meseguer, & Morán Cabré, 1987):

- *Consistencia*. Se considera que es la mayor o menos facilidad que tiene el hormigón para ser deformado. Depende de numerosos factores como la cantidad de agua de amasado (más importante), tamaño de los áridos, ...
  Existen varios métodos para determinar la consistencia: cono de Abrams (comentado anteriormente), la mesa de sacudidas y el consistómetro Vebe.
  La consistencia líquida no es admisible para hormigón armado.
- *Docilidad.* Se considera como la aptitud que tiene un hormigón para ser puesto en obra con los medios de compactación de que se dispone.
- *Homogeneidad.* Es la característica que presenta un determinado hormigón, por la cual sus diferentes componentes aparecen repartidos de manera regular en toda su masa. La homogeneidad se consigue con un buen amasado y, para mantenerse, requiere un transporte cuidadoso y una colocación adecuada.
  La homogeneidad de un hormigón se puede perder por segregación (separación de los gruesos y los finos) o por decantación.

- *Peso específico.* Es un dato muy importante y da una gran idea del índice de uniformidad del hormigón a lo largo de una obra.

2.1.3 Propiedades del hormigón endurecido

Las principales propiedades del hormigón endurecido son las que se muestran, a continuación:

- *Peso específico.* Para el hormigón endurecido, el peso específico depende principalmente de los áridos empleados (naturaleza y tamaño de grano) y del método de compactación utilizado.
  El peso específico será directamente proporcional al peso de los áridos y a la cantidad de árido grueso que contenga el hormigón. Posibles valores de cálculo para hormigón en masa son 2.3t/m3 y para el hormigón armado 2.5 t/m3.
- *Compacidad.* Una buena compacidad proporciona al hormigón tres cosas principalmente: mayor resistencia mecánica, mayor resistencia física y química.
- *Permeabilidad.* El factor principal que influye en la permeabilidad es:
  - Relación agua/cemento. Si disminuye esta relación, disminuye la permeabilidad. Esta variable es la que tiene una mayor influencia sobre la resistencia a compresión del hormigón.

  Para determinar la permeabilidad hay diferentes métodos, unos para medir la permeabilidad bajo presión y otros para medir la permeabilidad por succión.

- *Resistencia al desgaste.* En determinados usos del hormigón, como es el caso de pavimentos, es necesario que se presente una elevada resistencia al desgaste. Para ello, la principal característica que debe presentar el hormigón es que sea lo suficientemente seco.
  Se puede también, aplicar tratamientos superficiales que endurezcan como silicatos o carbonatos.
  Se considera imprescindible, de la misma manera, utilizar arenas silíceas y no calizas.

2.1.4 Retracción del hormigón

Cuando se produce el fraguado y endurecimiento del hormigón, éste contrae su volumen si el proceso se lleva a cabo en el aire o se entumece si el proceso tiene lugar en el agua. Al primer fenómeno se le conoce como retracción (Martín, 2007).

Se considera la retracción como la pérdida de agua en el hormigón durante su endurecimiento.

Factores que influyen en la retracción:

- Tipo, clase y categoría del cemento. Cuanto más resistente y rápido aportan mayor retracción al hormigón, a igualdad en el resto de las variables.
- Mayor retracción cuanto más fino es el molido del cemento.
- La presencia de finos en el hormigón provoca un aumento considerable de la retracción.
- Cantidad de agua de amasado. Cuanta más cantidad de agua mayor es la retracción.
- El hormigón en masa tiene mayor retracción que el hormigón armado.

- Elemento en contacto con el medio ambiente. Cuando menor sea el espesor del elemento en contacto con el medio ambiente mayor será la retracción.

A pesar de los esfuerzos que se realizan en definir algún tipo de hormigón que no experimente retracción, de momento no ha sido posible fabricarlo.

La retracción es una deformación impuesta que provoca tensiones de tracción, que pueden desembocar en fisuras, cuando se encuentra impedido el libre acortamiento del hormigón. Debido a esto, la retracción tiene más influencia cuando más tensa y rígida es la estructura. A continuación, se presentan ciertas imágenes con fisuras debidas a retracción:

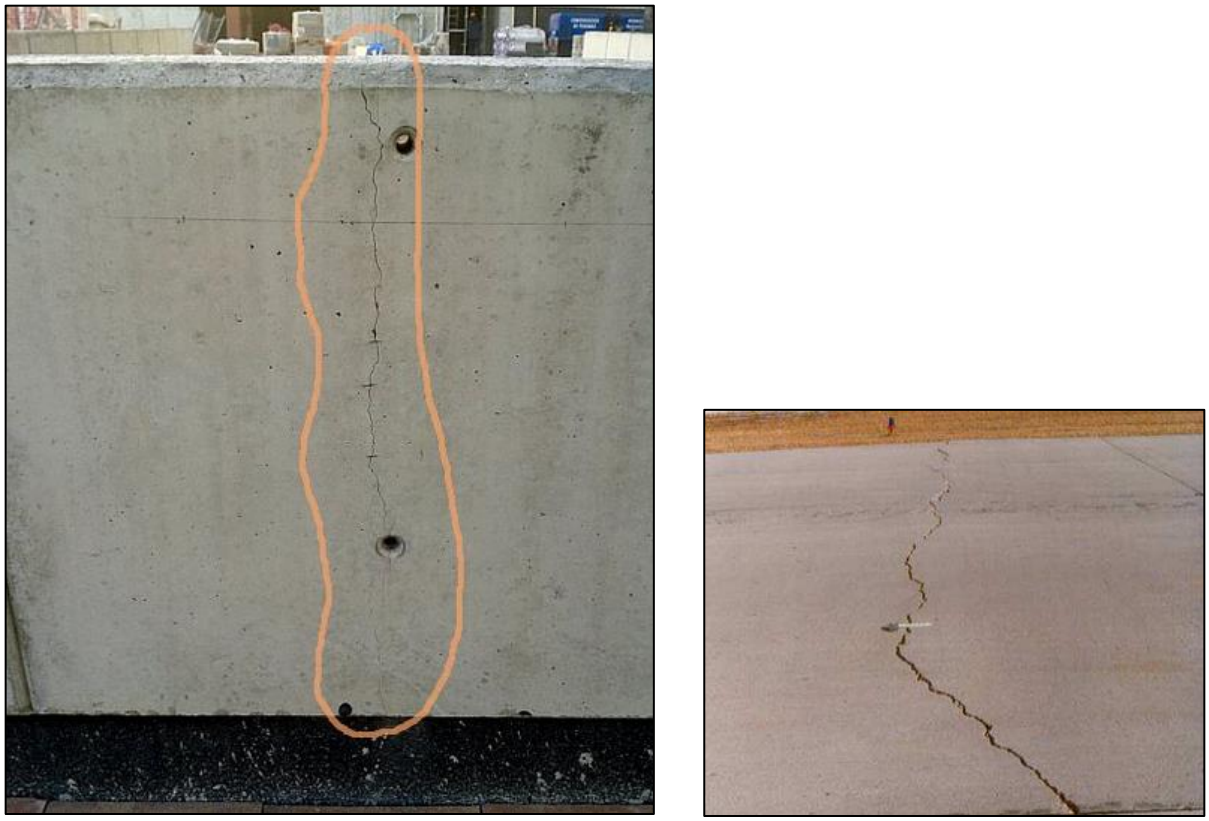

Figura 2. Ejemplos de fisuras por retracción del hormigón

2.1.5 El hormigón y la temperatura

El hormigón se comporta frente a las temperaturas experimentando una serie de fenómenos que se detallan en la siguiente tabla:

TEMPERATURA (^{0}C)	EFECTO SOBRE EL HORMIGÓN
<100^{0}C	Sin influencia
100^{0}C < T < 150^{0}C	El hormigón cede su agua capilar y de adsorción
150^{0}C durante tiempo	Ligera disminución de resistencia a compresión y gran caída de la resistencia a tracción
Hasta 250^{0}C en periodos cortos	Disminución de la resistencia a tracción
300^{0}C<T<500^{0}C	Pérdida de 20% de la resistencia a compresión. La resistencia de tracción puede haber desaparecido.
500^{0}C<T<900^{0}C	Se elimina el agua de cristalización, la cal hidratada se destruye
T = 900^{0}C ÷ 1000^{0}C	Destrucción completa del hormigón

Tabla 2. Acción de la temperatura sobre el hormigón

El coeficiente de dilatación térmica ($\propto$), mide el aumento de volumen que experimenta el hormigón cuando aumenta la temperatura. Como valor medio puede tomarse $\propto = 10^{-5}\ ºC$. Por tanto, la dilatación térmica se obtiene según la siguiente expresión:

$$\varepsilon_{\Delta T} = 10^{-5}\,\Delta T \qquad \text{(ec.1)}$$

2.1.6 Características mecánicas del hormigón

- Resistencia característica del hormigón

Es la resistencia a compresión simple. La característica más importante del hormigón. Esta característica, se determina por medio de ensayos normalizados con probetas cilíndricas. Eso sí, dependiendo de la obra y, el cuidado y rigor con que se realice el hormigón, estos valores de resistencia pueden diferir bastante de unas probetas a otras. Esto ha de tenerse en cuenta al definir la resistencia de un determinado hormigón.

Tradicionalmente, se ha adoptado el criterio de considerar la media aritmética de los valores de rotura de las probetas, más conocido como resistencia media. Aunque en la actualidad, este valor no refleja de manera muy fehaciente la dispersión que pueda presentar el hormigón y, por consiguiente, su calidad.

Buscando eliminar estos inconvenientes, se adopta el concepto de resistencia característica del hormigón, que es una medida estadística que tiene en cuenta, además

de la media aritmética (f_{cm}) , la dispersión. Así, se presenta la siguiente relación entre f_{cm} y f_{ck} (según el EC2):

$$\boldsymbol{f_{cm} = f_{ck} + 8(MPa)} \quad \text{(ec.2)}$$

- Resistencia a tracción del hormigón

Aunque el valor de la resistencia a tracción del hormigón, no suele tenerse en cuenta a efectos resistentes, sí que es importante para ciertos fenómenos como son la fisuración, esfuerzo cortante, la adherencia, etc. Es cierto, que en ciertos elementos de hormigón como pavimentos es importante conocer el valor de dicha resistencia porque dan un reflejo de su calidad y limpieza de los áridos.

De la misma manera que para la resistencia a compresión, el valor de la resistencia a tracción se obtiene por medio de ensayos y su valor depende del tipo de ensayo realizado. Hay tres maneras principales de obtener la resistencia a tracción: por flexotracción, por hendimiento y por ensayo de tracción axil.

- Resistencia a flexotracción $(f_{ct,fl})$:

$$\boldsymbol{f_{ct,fl} = f_{ct}\ \frac{1 + 1.5\left(\frac{h}{100}\right)^{0.7}}{1.5\left(\frac{h}{100}\right)^{0.7}}} \quad \text{(ec.3)}$$

Siendo h el canto del elemento en mm.

Según la norma EHE y el EC2, la resistencia a tracción (f_{ct}) se puede considerar el 90% de la resistencia a tracción obtenida por hendimiento.

Los valores obtenidos en los ensayos de resistencia a tracción presentan bastante dispersión y su variación se establece referido al valor medio:

- Resistencia característica inferior a tracción → 0.7 f_{ctm}
- Resistencia característica superior → 1.3 f_{ctm}

Aunque la resistencia a tracción depende de muchas variables, el EC2, admite que la resistencia a tracción media, se encuentra relacionada con la resistencia característica del hormigón a compresión de la siguiente manera:

$$\boldsymbol{f_{ctm} = 0.3\sqrt[3]{f_{ck}{}^2}, para\ f_{ck} \leq 50MPa} \quad \text{(ec.4)}$$

$$\boldsymbol{f_{ctm} = 2.12 \ln\left(1 + \frac{f_{cm}}{10}\right), para\ f_{ck} \geq 50MPa} \quad \text{(ec.5)}$$

2.1.7 Características reológicas del hormigón

La reología es la rama de la mecánica que estudia la evolución de deformaciones de un material debido a causas tensionales a lo largo del tiempo.

Debido a las diferentes fases que presenta el hormigón (sólida, líquida y gas), presenta una compleja reología. El fenómeno de la retracción forma parte de la reología del hormigón. A continuación, se ven otros fenómenos que en conjunción gobiernan la evolución de las deformaciones del hormigón con el tiempo.

- Clasificación de las deformaciones del hormigón

Una probeta de hormigón que es sometida a procesos de carga y descarga, evoluciona, en cuanto a deformaciones, de la siguiente manera (ver figura 3):

1. Hay que distinguir dos partes en la deformación instantánea del hormigón: la deformación elástica y la deformación remanente o permanente.
   Estos dos tipos de deformación, se pueden advertir si se observa la gráfica. Si en el instante inicial, la probeta se carga con una tensión (en este caso σ_0), la probeta experimentará una deformación que llegará hasta la posición A (deformación elástica). Si se deja de aplicar dicha carga, la deformación no volverá a ser cero, tendrá una deformación remanente (no nula).

2. Por otro lado, si se carga la probeta de manera que se origine una tensión $\sigma_1 < \sigma_0$, aparecerá una deformación elástica (tramo BC). Si la carga se mantiene constante con el tiempo, la deformación irá aumentando (curva CD), por el comportamiento plástico que presenta el hormigón. Si llegado un instante la probeta se descarga, la deformación elástica (DE=BC) de manera instantánea. Si se deja más tiempo para que siga recuperando, seguirá recuperando algo de la deformación experimentada (tramo EF).

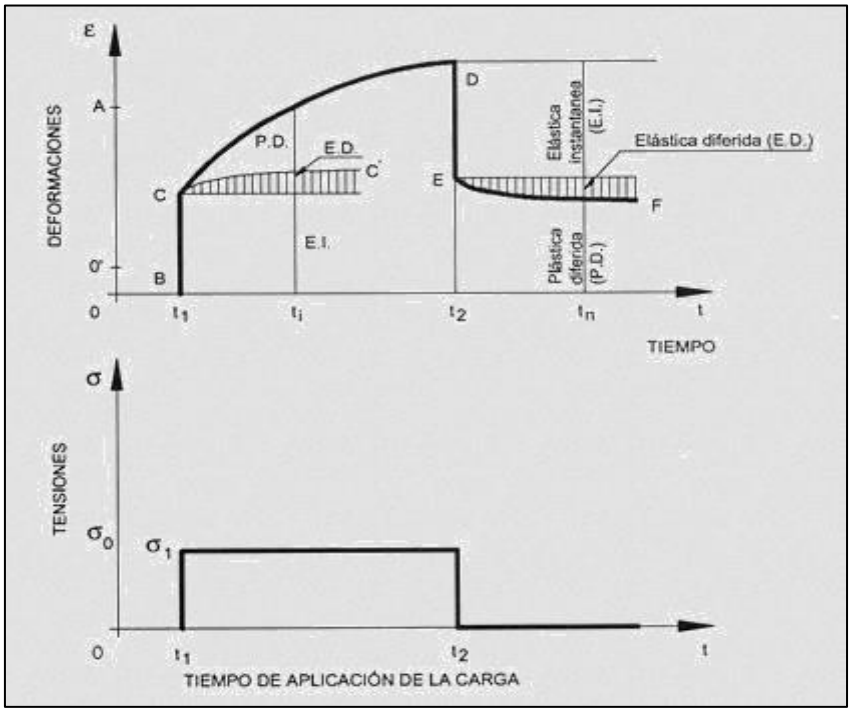

Figura 3. Deformaciones del hormigón

Por lo tanto, el hormigón presenta tres tipos de deformaciones fundamentales:

- Elástica instantánea → Reversible
- Elástica diferida → Reversible
- Plástica diferida → Remanente

A modo de explicación, se denomina fluencia al conjunto de deformaciones diferidas.

2.1.8 Cálculo en agotamiento. Estudio general

– Diagrama tensión-deformación del hormigón

Los resultados de los ensayos de rotura a compresión, en probetas de hormigón, dependen de la velocidad de aplicación de la carga. Se puede asumir, entonces, que el diagrama tensión-deformación depende del tiempo de aplicación de la carga (ver figura 4).

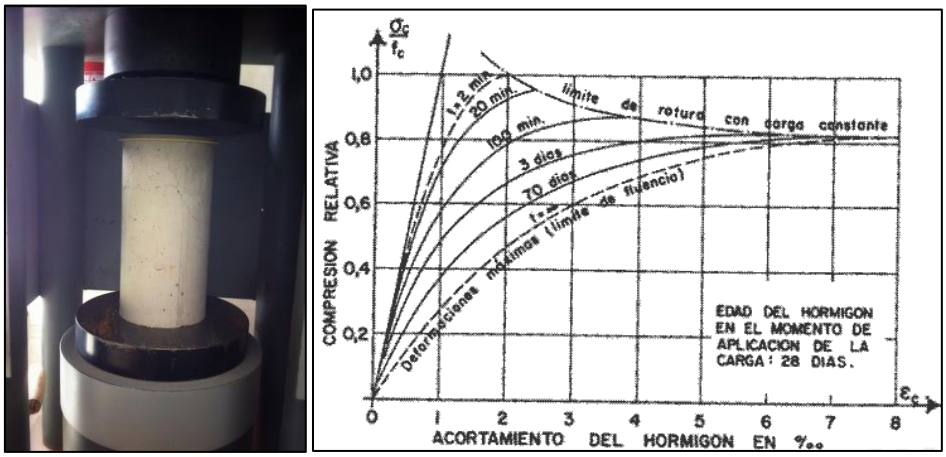

Figura 4. Diagrama tensión-deformación de una probeta sometida a un ensayo compresión

La diferencia entre diferentes tipos de diagramas tensión-deformación es menor cuanto mayor sea la edad del hormigón, es decir, para diferentes tiempos de aplicación de carga cuanto mayor es la edad del hormigón, menor es la diferencia entre sus diagramas de tensión-deformación. Diferentes variables también influyen en el aspecto del diagrama como son la sección, la humedad,...

Las normas establecidas del hormigón (EHE, EC2, etc.) proponen diferentes tipos de diagramas. Los diagramas se pueden dividir en dos grandes grupos: diagramas para el cálculo estructural y diagramas para el diseño de secciones en el ELU de agotamiento a tensiones normales.

Los diagramas empleados para el cálculo estructural deben proporcionar valores fiables en el rango de deformación que pretenda ser analizado. Por otro lado, cuando se quiere comprobar una sección en ELU, lo que interesa son los valores últimos de tensiones y deformaciones antes de la rotura.

Se hará una breve reseña de cada tipo de diagramas de tensión deformación.

- Diagramas para el cálculo estructural
  - *Modelo tensión-deformación lineal: deformación elástica*

  Este tipo de modelo se corresponde con una aproximación lineal del comportamiento unidimensional, para probetas cilíndricas que se ensayan a compresión durante poco tiempo (minutos).

La hipótesis más empleada en el cálculo de estructuras a base de hormigón es la de comportamiento lineal del material, con lo cual, se considera que el módulo de deformación longitudinal (E_{cm}) del hormigón a los 28 días es constante.

- *Modelo tensión-deformación no lineal*

En el caso de cálculos no lineales, tanto el EC2 como el EHE proporcionan un modelo tensión-deformación parabólico, como se muestra en la siguiente imagen:

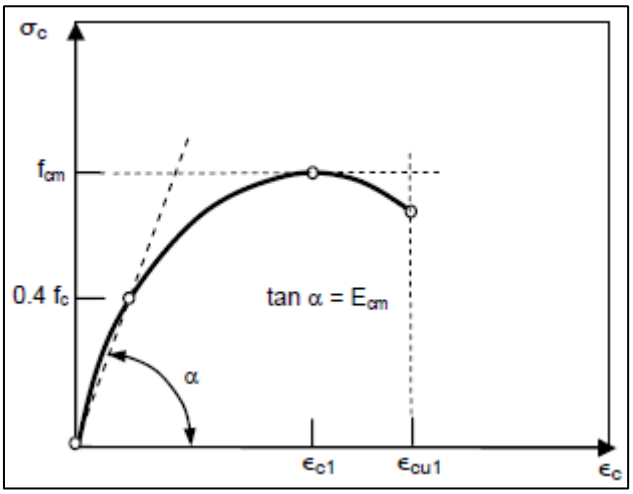

Figura 5. Diagrama parabólico tensión-deformación

Este tipo de diagrama, únicamente es utilizable con cargas de corta duración. Simplemente, se definirán los elementos que aparecen en el diagrama, sin entrar en formulación:

- σ_c → Tensión del hormigón para un valor de ε_c.
- ε_c → Deformación del hormigón
- ε_{c1} → Deformación del hormigón para tensión máxima
- ε_{cu1} → Deformación máxima del hormigón

Por último, decir de este diagrama que en la práctica es poco utilizado debido a la dificultad que entraña el hecho de que el módulo de deformación longitudinal dependa del nivel de tensión.

- <u>Diagramas para el diseño en rotura de secciones</u>

Esta parte va a ser de gran importancia para la resolución del respectivo proyecto, pues es donde se va a trabajar, y van a servir como base en nuestros posteriores cálculos.

Antes de nada, hay que introducir un término que de ahora en adelante será muy utilizado y es la resistencia de cálculo (o diseño) del hormigón (f_{cd}). Es necesario utilizar este término, debido a que tiene en cuenta coeficientes que aseguran que el hormigón no experimentará, en un futuro, fenómenos de cansancio. El EC2, lo define de la siguiente manera:

$$f_{cd} = \alpha_{cc} \frac{f_{ck}}{\gamma_c}$$

Donde:

γ_c → Coeficiente parcial de seguridad del hormigón. Los valores que puede tomar son:

Situación del proyecto	Hormigón
Persistente o transitoria	1.5
Accidental	1.3

Tabla 3. Coeficientes parciales de seguridad del hormigón

α_{cc}→ Coeficiente que tiene en cuenta los efectos negativos de la duración de la carga (cansancio) y de la forma de aplicar la carga. Tradicionalmente se toma el valor de 0.85. Aunque la norma española desarrolla el caso con valor 1.

La investigación a lo largo de los años ha hecho posible el desarrollo de modelos de tensión-deformación que permitan obtener el valor de la resistencia última del hormigón, con combinaciones de cargas de corta y larga duración. Los modelos más empleados son:

- Modelo rectangular

Este tipo de diagrama, es el más utilizado en rotura. Según el EC2, el diagrama rectangular corresponde a un valor constante de tensión ηf_{cd} en una profundidad efectiva comprimida de valor λx (siempre que $\lambda x \leq h, si\ \lambda x > h$ la profundidad efectiva vale h) donde x es la distancia de la fibra más comprimida a la fibra neutra y h es el canto de la sección. Los valores necesarios para definir el diagrama rectangular son:

$$\eta = \begin{cases} 1, & si\ f_{ck} \leq 50\ Mpa \\ 1 - \dfrac{f_{ck} - 50}{200}, & si\ 50 < f_{ck} \leq 90 Mpa \end{cases}$$

$$\lambda = \begin{cases} 0.8, & si\ f_{ck} \leq 50\ Mpa \\ 0.8 - \dfrac{f_{ck} - 50}{400}, & si\ 50 < f_{ck} \leq 90 Mpa \end{cases}$$

De la última ecuación, se puede concluir según el EC2 que la resistencia del hormigón (f_{ck}) depende de la profundidad efectiva (λx), debido a que el comportamiento del hormigón es más lineal cuanto mayor es su resistencia (ver figura 6):

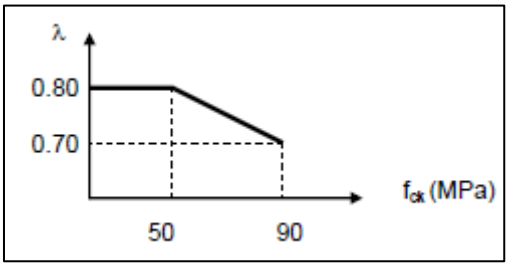

Figura 6. Valor de λ según el EC2

La deformación máxima que presenta el hormigón sometida a un estado de tensiones de flexo-compresión (ε_{cu3}) y la deformación máxima en compresión pura es (ε_{c3}). Sus valores se presentan en las siguientes ecuaciones:

$$\varepsilon_{c3} = \begin{cases} 1.75, & si\ f_{ck} \leq 50\ Mpa \\ 1.75 + 0.55\left(\dfrac{f_{ck} - 50}{40}\right), & si\ 50 < f_{ck} \leq 90 Mpa \end{cases}$$

$$\varepsilon_{cu3} = \begin{cases} 3.5, & si\ f_{ck} \leq 50\ Mpa \\ 2.6 + 35\left(\dfrac{90 - f_{ck}}{100}\right)^4, & si\ 50 < f_{ck} \leq 90 Mpa \end{cases}$$

A continuación, se presenta una representación de los diagramas de la distribución de tensiones tanto para el caso rectangular, como para el caso parabólico-rectangular que se explicará en el siguiente punto, de la zona comprimida:

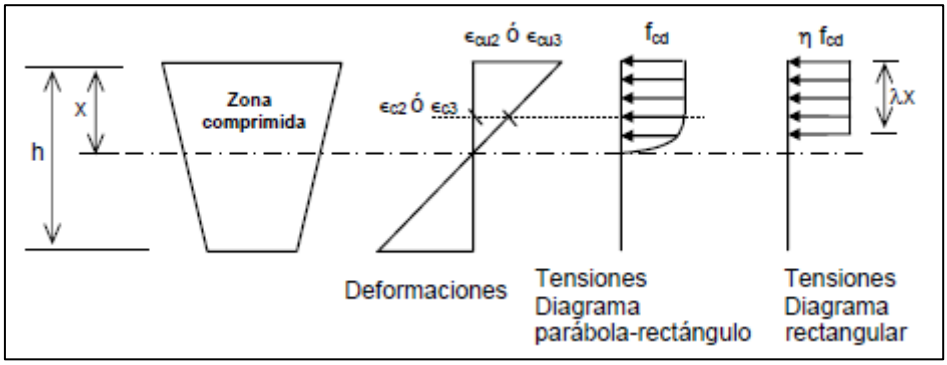

Figura 7. Distribución tensiones zona comprimida para los dos tipos de diagramas

- **Modelo parábola-rectángulo**

Una representación de este tipo de diagrama se presenta, a continuación:

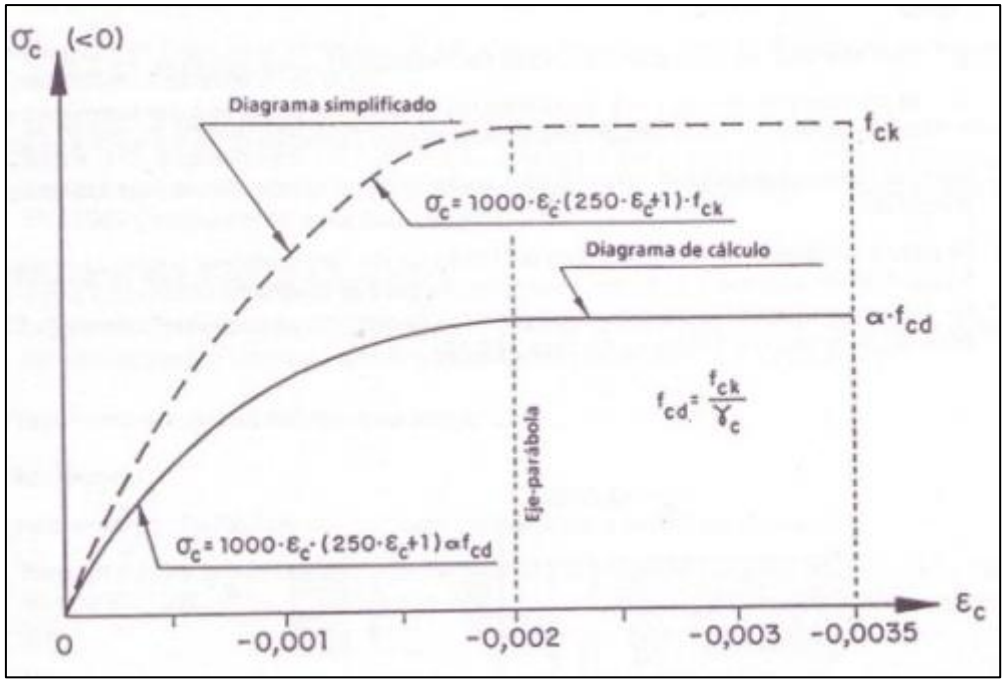

Figura 8. Diagrama parábola-rectángulo

Este tipo de diagrama consta de una parábola desde el origen y llega con pendiente horizontal al punto de deformación ε_{c2} con tensión f_{cd}. ε_{c2} es un valor constante (2 ‰), siempre que f_{ck} sea menos de 50 Mpa.

El diagrama parábola-rectángulo para valores de cálculo viene dado por la siguiente expresión:

$$\sigma_c = \begin{cases} f_{cd}\left[1-\left(1-\dfrac{\varepsilon_c}{\varepsilon_{c2}}\right)^n\right], para\ 0 \le\ \varepsilon_c\ \le\ \varepsilon_{c2} \\ f_{cd}, \qquad para\ \varepsilon_{c2}\ \le\ \varepsilon_c \le\ \varepsilon_{cu2} \end{cases}$$

Donde:

$$n = \begin{cases} 2 & si\ f_{ck} < 50Mpa \\ 1.4 + 23.4\left(\dfrac{90-f_{ck}}{100}\right)^4, & si\ \ f_{ck} \ge 50Mpa \end{cases}$$

$$\varepsilon_{c2}(‰) = \begin{cases} 2 & si\ f_{ck} < 50Mpa \\ 2 + 0.085(f_{ck}-50)^{0.53}, & si\ \ f_{ck} \ge 50Mpa \end{cases}$$

$$\varepsilon_{cu2}(‰) = \begin{cases} 3.5 & si\ f_{ck} < 50Mpa \\ 2.6 + 35\left(\dfrac{90-f_{ck}}{100}\right)^4, & si\ \ f_{ck} \ge 50Mpa \end{cases}$$

En el particular caso de una sección transversal sometida a compresión simple la deformación máxima del hormigón es la correspondiente a ε_{c2}.

Este tipo de diagrama, es el que se utilizará para la resolución de este proyecto.

2.2 ACERO

2.2.1 Armadura pasiva

Debido a la baja resistencia a tracción que presenta el hormigón, es necesario la introducción en la masa del hormigón de barras de acero, que confieren al material la resistencia a tracción que no aporta el propio hormigón. Si las barras que se introducen, se colocan sin tensión, se denominan armadura pasiva. Dicha armadura pasiva, también se puede introducir en zonas comprimidas, trabajando tanto la armadura como el hormigón a compresión.

Para garantizar el éxito de la armadura pasiva, las barras que trabajan a tracción deben de ir ancladas en zonas donde no se requiera su colaboración, que comúnmente suelen ser zonas comprimidas (Martín, 2007).

Para mejorar la adherencia entre el hormigón y el acero, la armadura pasiva generalmente es corrugada. El acero corrugado consiste en un conjunto de barras de

acero que tienen resaltos o “corrugas” (ver figura 9). Como ya se ha dicho anteriormente, éstas corrugas permiten una mejor adherencia aumentando la ductilidad del hormigón. Es un tipo de acero laminado, ideado principalmente para estructuras de hormigón armado. Con ello, disminuye la fragilidad del hormigón y se reducen riesgos de fractura ante grandes esfuerzos, por lo que ofrecen una gran seguridad a las construcciones.

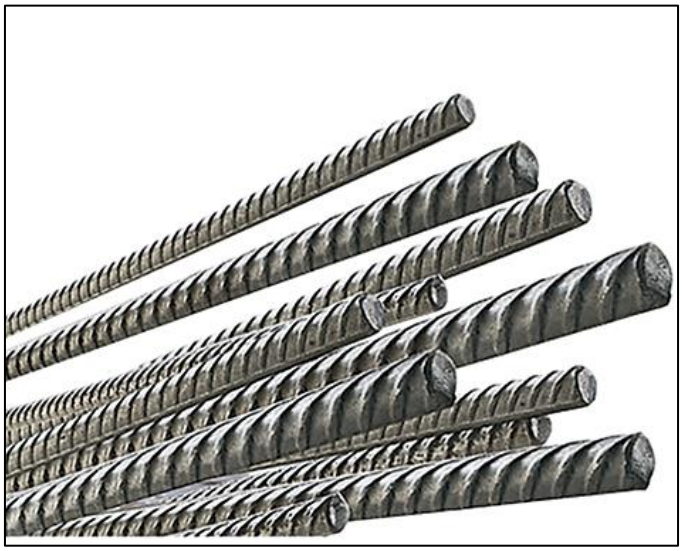

Figura 9. Barra de acero corrugado

La denominación de las barras corrugadas se realiza de la siguiente manera **B X00 S/SD**.

- **Letra B** → Siempre comienzan con la letra B
- **X00** → Corresponde al límite elástico f_y
- **Letra S** → Indica que es soldable
- **Letras SD** → Indica que tiene características especiales de ductilidad. También se utilizan en zonas sísmicas.

Los distintos tipos de aceros corrugados considerados por la EHE, se muestran en la siguiente imagen (Ver figura 10).

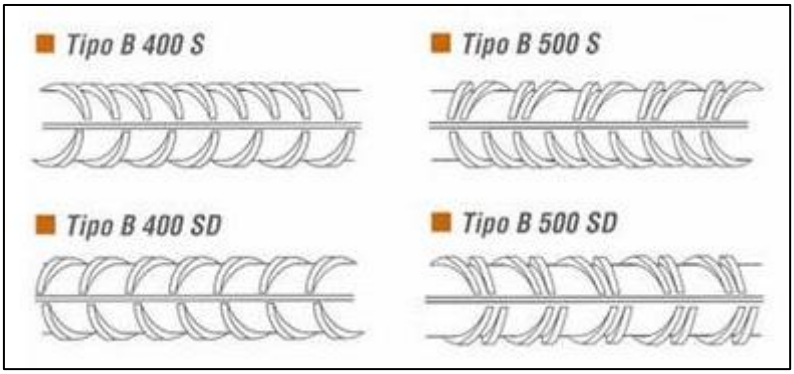

Figura 10. Tipos de aceros corrugados

Las características mecánicas de los tipos de aceros corrugados, se muestran en la tabla siguiente:

Características		Barras corrugadas			
Tipo de acero		B400S	B500S	B400SD	B500SD
Norma de producto		UNE 36068	UNE 36068	UNE 36065	UNE 36065
Límite elástico R_e (MPa)		400	500	400	500
Resistencia a la tracción R_m (MPa)		440	550	440	550
Relación R_m / R_e		1,05	1,05	≥1,20 ≤1,35	≥1,15 ≤1,35
Relación $R_{e\ real} / R_e$ nominal		-	-	≤1,20	≤1,20
Alargamiento de rotura A_5 (%)		14	12	20	16
Alargamiento total bajo carga máxima A_{gt} (%)	Recto	5,0	5,0	7,5	7,5
	Rollo*	7,5	7,5	10,0	10,0

Tabla 4. Tipos de aceros y algunas características mecánicas

La armadura pasiva se puede presentar de diversas formas: barras corrugadas, mallas electrosoldadas o armaduras clásicas electrosoldadas en celosía. (Ver figura 11).

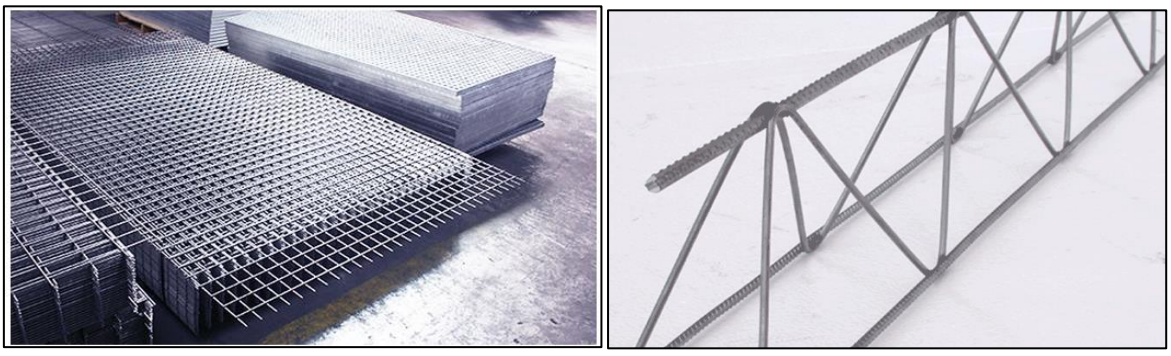

Figura 11. Armadura pasiva: malla (izquierda) y celosía (derecha)

Los diámetros nominales en los que se comercializan los diferentes tipos de armaduras pasivas son los mostrados, a continuación:

- Barras corrugadas: 6-8-10-12-14-16-20-25-32-40 mm.
- Mallas electrosoldadas: 6-6.5-7-7.5-8-8.5-9-9.5-10-10.5-11-11.5-12-14 mm.
- Armaduras electrosoldadas en celosía: 5-6-7-8-9-10-12 mm.

Posteriormente se verá el diagrama tensión-deformación más idóneo para los cálculos.

Diagramas tensión deformación de la armadura pasiva

Operando con hormigón, se puede echar mano de dos tipos de diagramas para la armadura pasiva. Uno, obtenido a partir de datos experimentales, siempre y cuando

los resultados de ensayos estén de acuerdo a la normativa vigente en el año que se lleven a cabo. Otro, puede ser el que se presenta en la figura 12 (propio del EC2), que sirve tanto para tracción, como para compresión (Jiménez Montoya, García Meseguer, & Morán Cabré, 1987).

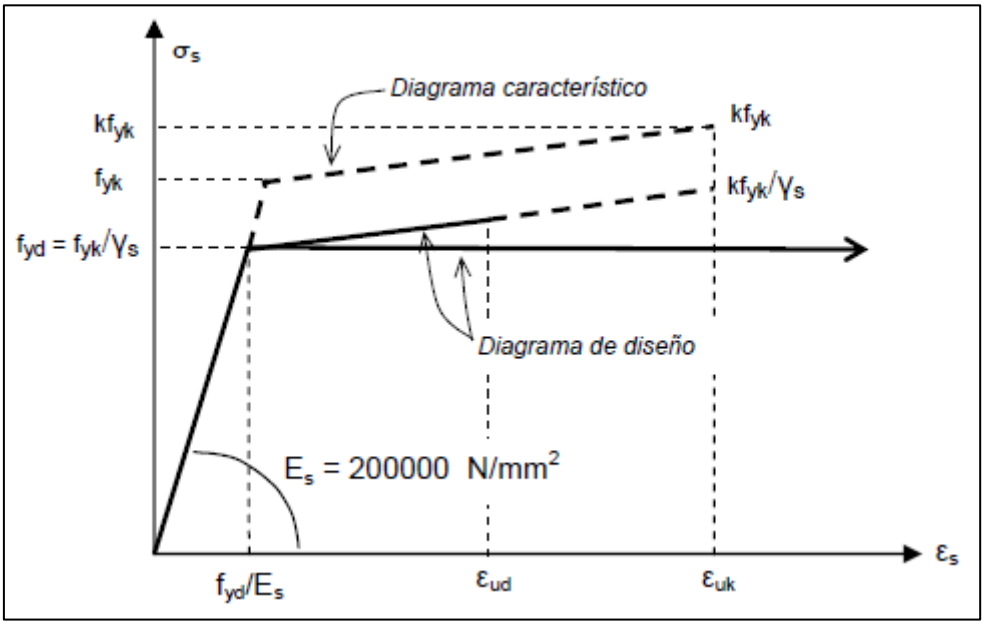

Figura 12. Diagrama de armadura pasiva

El valor f_{yk} es el límite elástico característico y f_{yd} es la resistencia de cálculo del acero. El módulo de elasticidad de la armadura pasiva es (ver figura 11) $E_s = 200000\ Mpa$.

2.2.2 Armadura activa

La otra manera de apaliar la escasa resistencia a tracción que presenta el hormigón, es mediante las armaduras activas. Se conocen como armaduras activas aquéllas que, fuerzan al hormigón a trabajar a compresión debido a las fuerzas de pretensado que introducen dichas armaduras (ver figura 13).

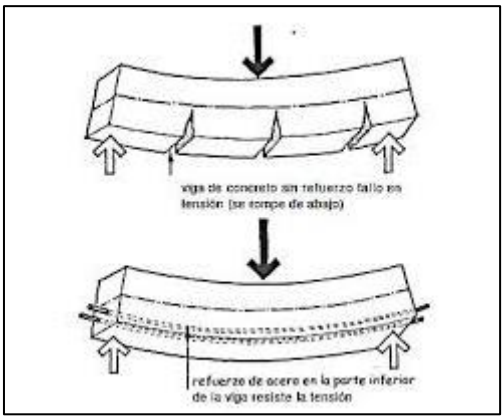

Figura 13. Ejemplo de viga sin refuerzo y viga con acero pretensado

Este tipo de armaduras se puede presentar por medio de tres formas diferentes (Martín, 2007):

- ***Alambres***: Se suministran por medio de bobinas. Producto de sección maciza de alto contenido en carbono (0.7 a 0.85%). Se confecciona a partir de un alambrón que se obtiene por laminado. La fuerza de pretensado se transmite por medio de la superficie de contacto. La EHE designa los alambres en relación a su carga unitaria máxima (f_{max}), como se observa en la siguiente tabla (tabla 4):

Designación	Diámetros nominales (mm)	Carga unitaria máxima (MPa)
Y 1570 C	9.4 – 10	1570
Y 1670 C	7 – 7.5 – 8	1670
Y 1770 C	3 – 4 – 5 – 6	1770
Y 1860 C	4 – 5	1860

Tabla 5. Tipos de alambres

- ***Barras***: Productos de sección maciza. Se comercializa mediante elementos rectilíneos. Se fabrican mediante estirado en frío. Los diámetros de las barras varían entre 20 y 40 mm.
- ***Cordón***: Se forman mediante varios alambres arrollados de manera helicoidal. El cordón de uso más común es el de 7 alambres. Se diferencia entre cordones de alta y baja relajación, en función del proceso de obtención. Los cordones, al igual que los alambres, se designan por su carga unitaria máxima (f_{max}).

También hay que incluir el tendón, que se considera al conjunto de varias armaduras de pretensadas paralelas, alojadas dentro de un mismo conducto.

Diagramas tensión deformación de la armadura pasiva

Para el caso de armadura activa, el diagrama característico tensión-deformación, tiene también dos variantes. La primera, puede ser suministrada directamente por el fabricante (con la correspondiente garantía). La segunda opción, tiene dos variantes en función de la norma de la que provenga:

1. EHE. Como se observa en la figura 13, tiene un primer tramo recto con pendiente E_p, y un segundo tramo a partir del punto 0.7f_{pk} de tensión.

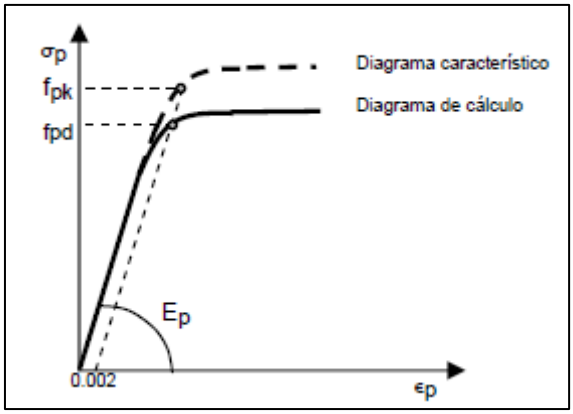

Figura 14. Diagrama tensión-deformación armadura activa, según EHE

En este caso, los módulos de deformación longitudinal pueden ser E_p=200000 MPa para alambres y barras o E_p=190000 MPa para cordones.

El valor de f_{pk} corresponde a la resistencia a tracción. Según se observa en la figura 13, ε_{uk} corresponde con la deformación para la carga máxima.

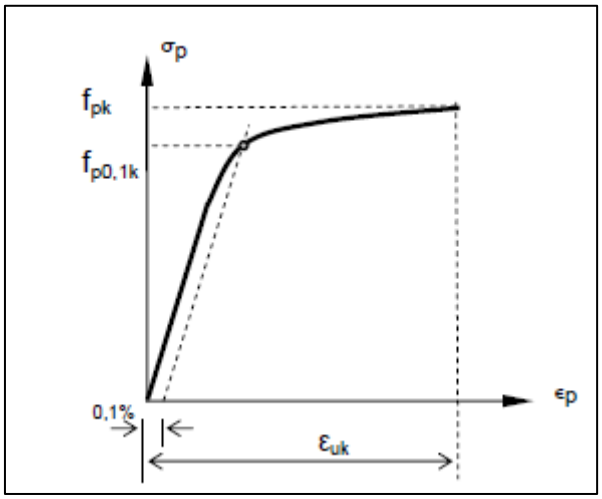

Figura 15. Diagrama tensión-deformación (valores de tracción)

2. <u>EC2.</u> El EC2 propone el siguiente diagrama mostrado a continuación (ver figura 15):

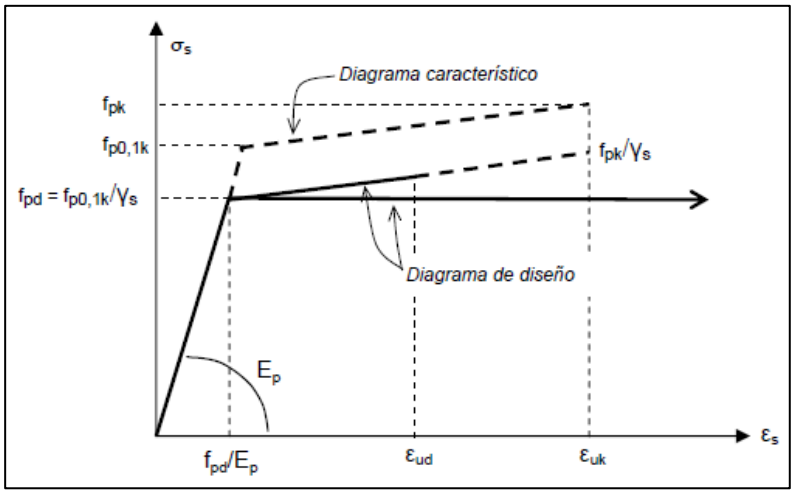

Figura 16. Diagrama tensión-deformación según el EC2 par armadura activa

Este diagrama recomienda como valor de deformación unitaria límite $\varepsilon_{ud} = 0.8\ \varepsilon_{uk}$.

Capítulo 3

HORMIGÓN ARMADO. ANÁLISIS DE LA SECCIÓN A FLEXIÓN

3 HORMIGÓN ARMADO. ANÁLISIS DE LA SECCIÓN A FLEXIÓN

En este capítulo, inicialmente se comentarán los estados límite. Se verá el comportamiento que presenta el conjunto hormigón-acero a nivel de sección transversal. Centrándose más ampliamente en el estudio de las secciones sometidas a tensiones normales, es decir, aplicados unos esfuerzos normales se realizará la comprobación de cómo se comporta una sección transversal: deformación unitaria, dimensionamiento (es lo que centrará nuestro proyecto y resistencia.

Primeramente, se hará un repaso de las hipótesis a nivel de sección que hay que tener en cuenta. Se verán los diagramas tensión-deformación que presentan cada uno de los elementos que forman el hormigón armado.

Posteriormente se verán los diferentes tipos de flexión, para posteriormente centrarse en el caso que atañe a este proyecto, que es el caso de flexión esviada.

3.1 Estados límite

A modo de introducción, se hablará brevemente de los estados límite.

Toda estructura debe reunir condiciones adecuadas de seguridad, funcionalidad y durabilidad, con objeto de que pueda rendir el servicio para el que fue confeccionado y proyectado.

De ahora en adelante, se definirá como estado límite todo estado, ya sea de tensión como de deformación, que de ser rebasado, hace que la estructura quede fuera de servicio (Páez, 1986). Los estados límite se clasifican en dos categorías radicalmente distintas, debido a la muy diferente magnitud de las pérdidas que potencialmente genera la situación extrema:

- ***Estados límite últimos (ELU).*** Corresponden a la máxima capacidad resistente de la estructura. Quedan en este grupo comprendidos aquellos estados que, al ser alcanzados, pueden provocar la ruina inmediata de la estructura con el peligro físico que ello conlleva para los utilizadores de dicha estructura.

  Estos estados límite pueden a su vez clasificarse en dos subgrupos:
  - ELU causados por una situación desfavorable que pueda producir una solicitación que agote, con su acción, la capacidad resistente de una sección crítica.
  - ELU que, de modo progresivo, deterioran finalmente la capacidad mecánica de la pieza.

  Los estados límites últimos se relacionan con la seguridad de la estructura y son independientes de la función que ésta cumpla. Los más importantes son:
  - Equilibrio. Caracterizado por pérdida de estabilidad estática.
  - Agotamiento. Caracterizado por el agotamiento resistente ya sea por rotura o por deformación plástica excesiva.
  - Pandeo. El pandeo es un fenómeno de inestabilidad en elementos esbeltos que puede ser provocado por compresión, flexión o flexocompresión en el

que, para cierto valor crítico de carga, flecta súbitamente y aparecen grandes desplazamientos transversales para incrementos muy pequeños de la carga.

- Fatiga. Se caracteriza por la rotura por efecto de fatiga ante acción de cargas repetidas.

- Adherencia. Se caracteriza por la rotura de la adherencia entre las armaduras de acero y el hormigón. Se estudia a nivel de sección.

- Anclaje. Evalúa la posibilidad de que un anclaje ceda. Se estudia de manera local en zonas de anclaje.

- ***Estados límite de servicio (ELS).*** Corresponden a la máxima capacidad de servicio de la estructura. Se relacionan con la funcionalidad, estética y durabilidad de la estructura y dependen de la función que ésta deba cumplir. Se corresponde con aquellos estados que, de ser alcanzados, dejan fuera de servicio a la estructura sin provocar su ruina. Debido a que estos estados no atentan la seguridad de los usuarios, se admiten probabilidades de ocurrencia mucho mayores que para los ELU. Los más importantes, para estructuras de hormigón armado son:
  - Deformación excesiva. Se caracteriza por alcanzarse un determinado movimiento (flecha, giro) en un elemento de la estructura.

  - Fisuración controlada. Se caracteriza por que la abertura máxima de las fisuras de una pieza alcance un determinado valor límite, que es función de las condiciones ambientales que rodeen a dicha pieza.

  - Vibraciones excesivas. Se caracteriza por la producción en la estructura de vibraciones de una determinada amplitud o frecuencia.

El presente trabajo, se va centrar en el ELU de agotamiento flexión biaxial o esviada en secciones de hormigón armado.

El método de los estados límite centra su estudio sobre las acciones y las características de los materiales. Se supone que los materiales de la estructura permanecerán en perfecto estado durante toda su vida útil. Para poder garantizar dicha suposición, hay que tomar medidas paliativas en función del tipo de ambiente en el que se encuentre dicha estructura y se asegurará la calidad durante el total del proceso constructivo (Martín, 2007).

Las hipótesis de cálculo establecidas para el análisis estructural son apropiadas para predecir el comportamiento estructural y los estados límites considerados. El método de los estados límites se basa en el principio de la superposición y, por tanto, todo análisis de tipo estructural que no cumpla dicho principio no es compatible con el método de los estados límite último (Martín, 2007).

Las acciones sobre una estructura se pueden clasificar en función de varios criterios:

- Por su forma de aplicación: directas (cargas) o indirectas (deformaciones impuestas, acciones térmicas, giros impuestos, etc.).

- Según su variación con el tiempo:
  - G, permanentes. El peso propio

- Q, variables. Sobrecarga de uso, viento o nieve.
- A, accidentales. Como pueden ser explosiones o impactos ocasionales.

- Por su variación espacial: fijas o libres

- Por su naturaleza: estáticas o dinámicas

3.2 Hipótesis fundamentales a nivel de sección de hormigón armado

Las principales hipótesis que conforman el estudio del hormigón armado, a nivel de sección para este proyecto en particular son (Jiménez Montoya, García Meseguer, & Morán Cabré, 1987) (Torrano & Martí):

1. *Caracterización del ELU*. Desde la tracción simple a la compresión pura, hay una serie de situaciones de agotamiento correspondientes a las distintas solicitaciones normales que cubren. Esto se verá en el punto (3.2.3).

2. *Compatibilidad de deformaciones*. Las deformaciones que sufren las armaduras y el hormigón que las envuelve es la misma, bajo las acciones que actúan. Se admite aquí, la ley de Bernouilli, de que las deformaciones normales a una sección transversal siguen una ley plana.

   Debido a estas dos primeras hipótesis, quedan determinadas las deformaciones en todas las fibras.

3. *Diagrama tensión-deformación del hormigón*. Se verá en el siguiente punto (3.2.2). únicamente establecer que el diagrama que se utilizará en este proyecto para la consecución de los objetivos, es el diagrama parábola-rectángulo.

4. *Diagrama tensión-deformación de los aceros*. Se desarrollará en el siguiente punto también (3.2.2).

5. *Condiciones de equilibrio*. A nivel de sección, tiene que existir equilibrio entre las resultantes de las tensiones de hormigón y acero y los esfuerzos exteriores. El estado de tensiones presente en una sección debe desarrollar los esfuerzos que la solicitan, ya sea en ELU o en ELS. Para el caso de flexión compuesta (axil y momento):

$$\boldsymbol{N = \Sigma N = \int_{A_C} \sigma_c \, dA_c + \int_{A'_s} \sigma'_s \, dA'_s - \int_{A_s} \sigma_s \, dA_s - \int_{A_p} \sigma_p \, dA_p} \quad \text{(ec.6)}$$

$$\boldsymbol{M = \Sigma M = \int_{A_C} \sigma_c \, y \, dA_c + \int_{A'_s} \sigma'_s \, y \, dA'_s - \int_{A_s} \sigma_s \, y \, dA_s - \int_{A_p} \sigma_p \, y \, dA_p} \quad \text{(ec.7)}$$

Siendo:

- σ_c → Tensión en el hormigón
- σ_s → Tensión en la armadura pasiva
- σ_p → Tensión en la armadura activa

Considerando positivo el esfuerzo axil de compresión y el momento que produce un giro en sentido anti horario.

3.3 Dominios de deformación de las secciones, en ELU de agotamiento de sección

Se va a analizar la sección transversal de la viga, que se observa en la figura 17. Dicha sección, puede agotar por solicitaciones normales siguiendo distintos planos (que se verán a continuación) que pasan desde la tracción pura a la compresión pura, como ya se ha dicho anteriormente. Estos planos de deformación, también se denominan planos de rotura.

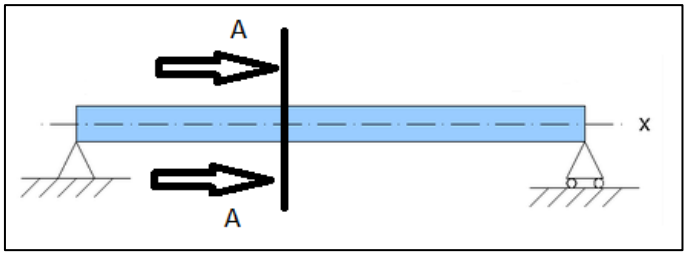

Figura 17. Sección A-A de viga

La nomenclatura que se utilizará durante el desarrollo del estudio de la sección transversal A-A es la siguiente: d es el canto útil o distancia que hay desde la fibra superior hasta el centro de gravedad de la armadura traccionada, d' es la distancia entre la fibra superior y el centro de gravedad de la armadura comprimida, b es la anchura de la sección y h es la altura de la sección.

En la figura 18, se muestran todos los planos de rotura posibles. A continuación se verán las características de cada uno de los planos (Martín, 2007):

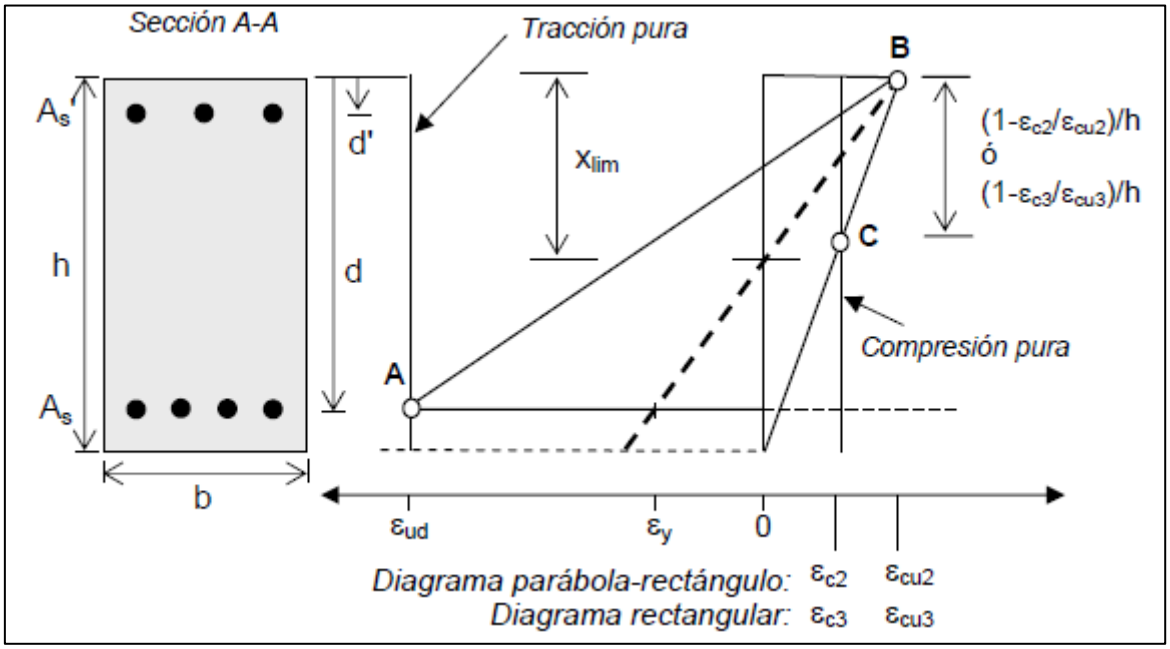

Figura 18. Criterios de agotamiento del hormigón armado

En primer lugar, los puntos:

- A → Deformación última del acero en tracción
- B → Deformación última del hormigón en compresión con flexión

- C → Deformación última del hormigón en compresión pura

A continuación, se van a ir viendo cada uno de los planos de deformación o rotura:

- **Plano 1** → Se corresponde con la línea vertical que pasa por el punto A (tracción pura). Toda la sección está a tracción y con una deformación ε_{ud} (10 ‰) agotando el acero a tracción.

- **Plano 2** → Este plano no aparece dibujado en la figura 16. Es el plano en el cual el acero inferior se encuentra a tracción con una deformación ε_{ud} y la zona superior de la viga se encuentra sin deformar.

- **Plano 3** → Es aquél formado por los puntos A y B. Es decir, el acero inferior se encuentra traccionado con una deformación ε_{ud} y la zona superior de la viga se encuentra a compresión con la deformación máxima ε_{cu2} o ε_{cu3} (suponen un 3.5‰ en hormigones con resistencia inferior a 50 MPa).

- **Plano 4** → En este plano, el acero inferior se encuentra traccionado con una deformación $\varepsilon_y = f_{yd}/E_S$, mientras que la zona superior de la viga se encuentra comprimida con una deformación ε_{cu2} o ε_{cu3} (3.5‰). La profundidad de la fibra neutra se conoce como profundidad límite (x_{lim}), ver figura 16. Ver dicho plano representado mediante línea a trazos.

- **Plano 5** → El acero inferior presenta una deformación nula, mientras que la zona superior de la viga se encuentra con una deformación ε_{cu2} o ε_{cu3} (3.5‰). Este plano no aparece en la figura 16.

- **Plano 6** → La zona inferior de la viga (o fibra inferior) tiene tensión nula y la zona superior de la viga (o fibra superior) se encuentra a compresión con una deformación ε_{cu2} o ε_{cu3} (3.5‰). Ver en la figura 16, el plano que une los puntos B y C.

- **Plano 7** → La viga se encuentra a compresión pura con una deformación máxima de ε_{c2} o ε_{c3} (2‰, en hormigones con una resistencia inferior a 50 MPa).

Cuando aparece el subíndice 2, se refiere al caso en el que se emplee el diagrama parábola rectángulo, mientras que si aparece el subíndice 3 está referido al caso del diagrama rectangular.

A su vez, se definen dominios comprendidos por distintos planos:

- **Dominio 1** → Compone el rango de planos del 1 al 2. Es tracción simple o compuesta con toda la sección a tracción. Las rectas de deformación giran en torno al punto A correspondiente a una deformación del acero de 10 ‰.

- **Dominio 2** → Compone el rango del plano 2 al plano 3. Flexión simple o compuesta donde el hormigón no alcanza la rotura. Las rectas de deformación siguen girando en torno al punto A.

- **Dominio 3** → Comprendido entre los planos 3 y 4. Flexión simple o compuesta donde las rectas de deformación giran en torno al punto B, que corresponde con la máxima deformación (acortamiento) del hormigón (3.5‰). El alargamiento de

la armadura más traccionada se encuentra en 10 y ε_y, siendo ε_y la deformación correspondiente al límite elástico del acero.

- **Dominio 4** → Comprendido entre los planos 4 y 5. Flexión simple o compuesta donde las rectas de deformación continúan girando en torno al punto B. El alargamiento de la armadura más traccionada se encuentra entre ε_y y cero.

- **Dominio 4a** → Comprendido entre los plano 5 y 6. Flexión compuesta donde las armaduras (superior e inferior) se encuentran comprimidas y existe una pequeña zona de hormigón a tracción. Las rectas de deformación siguen girando en torno al punto B.

- **Dominio 5** → Comprendido entre los planos 6 y 7. Compresión simple o compuesta y donde, tanto acero como hormigón trabajan a compresión. Las rectas de deformación giran en torno al punto C.

En la figura 17, se representan los dominios de rotura o deformación para flexión positiva, es decir, que la zona comprimida se encuentra en la parte superior y la zona traccionada se encuentra en la zona inferior de la sección transversal considerada. Para el caso de flexión negativa, sería al contrario y también tiene sus correspondientes dominios de deformación.

Cada uno de estos planos de deformación, son planos de agotamiento por rotura.

A la hora de definir un plano de deformación, son necesarias dos variables, que normalmente suelen ser la deformación del centro de gravedad y la curvatura. Es interesante apuntar, que un plano de rotura queda definido únicamente por una variable, que suele ser la profundidad de la fibra neutra x. Por lo tanto, cada valor de profundidad de fibra neutra corresponde con un único valor de plano de rotura.

En este punto, hay que introducir el término de *diagrama de interacción*. Se denomina diagrama de interacción a un gráfico M-N donde se representan los pares de puntos (M, N) que producen los planos de rotura. Este diagrama es un gráfico cerrado donde los puntos que caen dentro del mismo pueden ser resistidos por la sección, mientras que los puntos que caen fuera del mismo serían los puntos que no resistiría dicha sección. Ver figura 19.

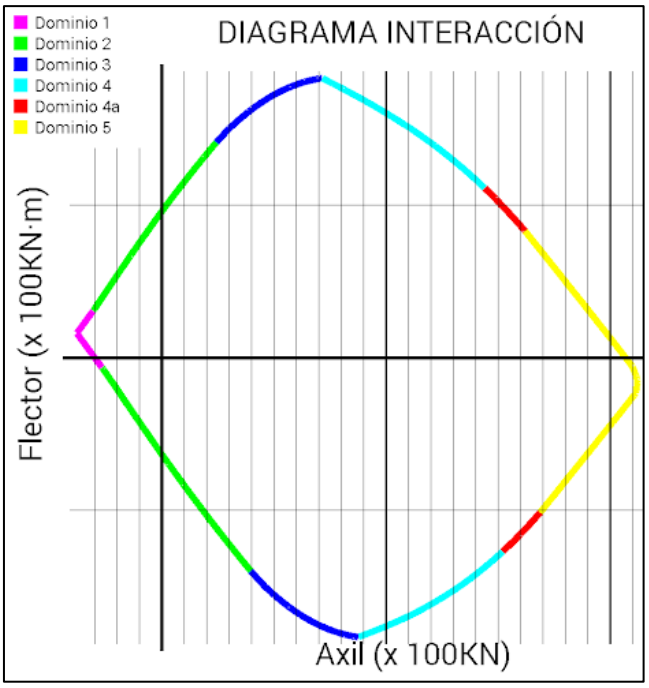

Figura 19. Ejemplo de diagrama de interacción

Los diagramas de interacción son muy útiles a la hora de comprobar si una sección transversal resiste o no resiste una solicitación del tipo (N-M).

3.4 Flexión simple y compuesta uniaxial

Antes de comenzar a hablar de la flexión compuesta biaxial o esviada, se va a introducir la flexión simple o compuesta uniaxial para ir del caso más simple al caso más complejo. Estos dos casos, son más sencillos de abordar y ayudan a entender más fácilmente, cómo funciona el caso que se verá en el siguiente punto, que presenta mayor dificultad.

En primer lugar hay que diferenciar entre:

- ***Flexión pura***. Exclusivamente, se aplica un momento flector en un eje principal de inercia

- ***Flexión simple***. Sobra la sección actúa un momento flector en un eje principal y un cortante.

- ***Flexión compuesta o compuesta uniaxial***. Además de lo anteriormente dicho, actúa un esfuerzo axil sobre la sección.

- ***Flexión esviada o biaxial***. Cuando el momento flector no actúa directamente sobre un eje principal de inercia o más de un momento flector actúa sobre la sección. Se verá más adelante en un punto aparte.

El estudio de una sección sometida a flexión simple o compuesta uniaxial se puede plantear de dos maneras diferentes: el problema de comprobación y el problema de dimensionamiento. Un poco más adelantes se verán en profundidad.

A parte, habrá que aplicar las ecuaciones de equilibrio (ec6 y ec7), ya vistas anteriormente:

$$N_d = \Sigma N = \int_{A_C} \sigma_c \, dA_c + \int_{A'_s} \sigma'_s \, dA'_s - \int_{A_s} \sigma_s \, dA_s - \int_{A_p} \sigma_p \, dA_p \quad \text{(ec.8)}$$

$$M_d = \Sigma M = \int_{A_C} \sigma_c \, y \, dA_c + \int_{A'_s} \sigma'_s \, y \, dA'_s - \int_{A_s} \sigma_s \, y \, dA_s - \int_{A_p} \sigma_p \, y \, dA_p \quad \text{(ec.9)}$$

Si existen dos niveles de armadura, situados a distancias d y d' de la fibra superior, como se vio en la figura 16, las ecuaciones de equilibrio resultarían de la siguiente manera:

$$N_d = \Sigma N = -\sigma_s A_s + \sigma'_s A'_s + N_c \quad \text{(ec.10)}$$

$$M_d = \Sigma M = \sigma_s A_s \left(d - \frac{h}{2}\right) + \sigma'_s A'_s \left(\frac{h}{2} - d'\right) + N_c \, y_{Nc} \quad \text{(ec.11)}$$

Siendo:

- N_c, el sumatorio de los axiles en el hormigón que depende de x.
- y_{Nc}, distancia al centro de gravedad de la sección.

3.4.1 Problema de comprobación

Este tipo de problema se basa en analizar, si una determinada sección resiste un determinado par de esfuerzos (N, M). Sobre la resolución de este problema, está basado este proyecto.

Para la resolución de esta clase de problema se cuenta con los siguientes datos:

- Sección. La forma de la sección transversal, que puede ser de cualquier tipo: rectangular, circular, doble T,...

- Área de las armaduras. Tanto el área de la armadura superior (A'_s) como el área de la armadura inferior (A_s).

- Acciones que actúan sobre la sección. Es el par de esfuerzos N_d y M_d.

Las incógnitas que se tendrán en este problema son:

- X → Fibra neutra

- M_{max} o N_{max} → Momento máximo o axil máximo que resistirá la sección.

Una manera de explicar de manera sencilla cómo funciona este problema, es la siguiente. Teniendo un diagrama de interacción, si el par de esfuerzos (N_d, M_d), se encuentra dentro de dicho diagrama, la sección no agotará. Por el contrario, si dicho par de esfuerzos se encuentra fuera del diagrama de interacción, la sección agotará y no resistirá. Esto se muestra en la siguiente imagen (ver figura 20):

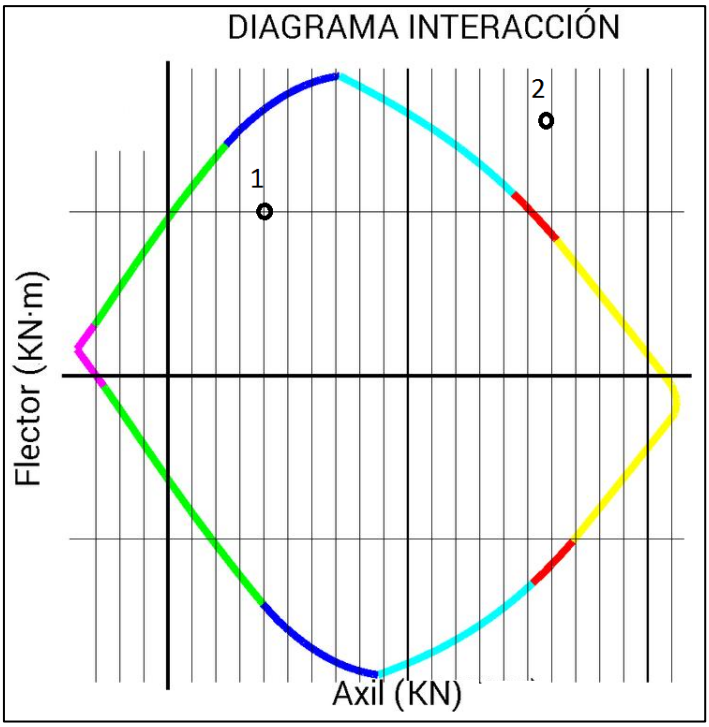

Figura 20. Ejemplo diagrama interacción

En la figura 20, el punto 1 se corresponde con un par de esfuerzos que resiste la sección, mientras que el punto 2 es un par de esfuerzos que no resiste la sección y agota.

La manera más rápida y menos costosa de hacer la comprobación anterior, debido a que resultaría demasiado tiempo y coste realizar el diagrama de interacción completo, es la siguiente:

- Se considera un punto cualquiera, en este caso se considerará el punto P, el cual tiene un par de esfuerzos ($N_{d,P}$, $M_{d,P}$). En la ecuación (ec.21), se impone que $N_d = N_{d,P}$ y se despeja x.

- Utilizando el valor de x obtenido anteriormente en la ec.22, se obtiene un momento que se denomina M_{Pmax}.

- Se comparan M_{Pmax} con $M_{d,P}$. Entonces puede haber dos casos:
  - $M_{d,P} > M_{Pmax}$ → La sección no resiste
  - $M_{d,P} < M_{Pmax}$ → La sección puede soportar dicho par de esfuerzos.

Este mismo planteamiento se puede hacer si, en vez de imponer la igualdad de axiles se impone la igualdad de momentos y el problema se resolvería de la misma manera.

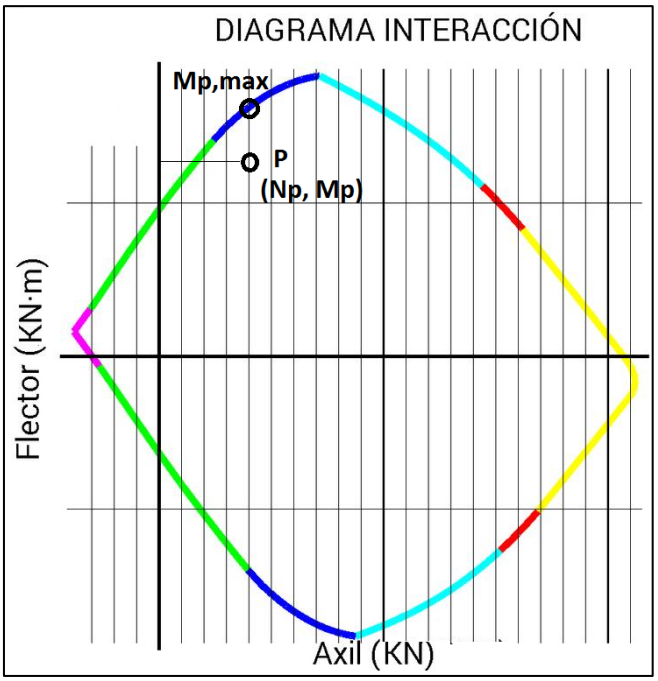

Figura 21. La sección resiste

3.4.2 Problema de dimensionamiento

Este caso, consiste en dimensionar una determinada sección transversal. Para ello, hay que calcular la armadura necesaria para que dicha sección resista un determinado par de esfuerzos (N_d, M_d). Para la resolución de este problema, se cuenta con dos ecuaciones de equilibrio (ec.8 y ec.9) y con 3 incógnitas que son: las áreas de las armaduras (A_s y A'_s) y la fibra neutra (x). Por lo tanto, el problema de dimensionamiento con armadura superior e inferior tiene infinitas soluciones.

No se entrará en exceso en la resolución de este tipo de problema, sólo se apuntará que para su resolución se suele utilizar alguna hipótesis que reduzca el número de incógnitas, como son:

- $A_s = A'_s$ → Típico caso de pilares
- A'_s=0 → Casos en los que no es necesario disponer armadura de compresión
- Imponer $x=x_{lim}$

El problema de dimensionamiento con dos niveles distintos de armadura tiene infinitas soluciones. Las infinitas soluciones se pueden representar gráficamente, mediante diagramas de armado a flexión o RSD, que representan las infinitas soluciones en función de la profundidad de la fibra neutra.

3.5 FLEXIÓN COMPUESTA BIAXIAL O ESVIADA

Una vez introducidos los conceptos de flexión uniaxial y compuesta, analizadas las ecuaciones de equilibrio y planteados los dos tipos de problemas que puede haber (comprobación y dimensionamiento), se va a pasar a desarrollar el tema de la flexión compuesta biaxial o esviada.

Se dice que una sección se encuentra sometida a un estado de flexión compuesta biaxial, cuando no se conoce la dirección de la fibra neutra. Este estado puede presentarse en los casos siguientes:

- Secciones que no presentan un plano de simetría, debido a la forma que presentan. Pueden nombrarse en este caso las secciones en L.

- Secciones que, aun siendo simétricas en forma, están armadas de manera asimétrica respecto a su plano de simetría.

- Secciones que, aun siendo simétricas en forma y armaduras, están sometidas a solicitaciones que no están contenidas en el plano de simetría.

Este último caso es el más frecuente. Dentro de él, se encuentran:

- Vigas que puedan estar sometidas a cargas laterales (viento, empujes de agua o tierras, etc.).

- Pilares que forman parte de pórticos planos pues el viento o el mismo pilar puede provocar flexiones secundarias que, por otro lado, suelen despreciarse.

Hasta hace pocos años, este tipo de problema ha sido bastante rehuido, debido a su complejidad y a que, hasta hace "poco", no había métodos prácticos o capacidad de cálculo con los ordenadores para resolverlo.

Las principales diferencias que hay entre la flexión uniaxial y la flexión biaxial son:

1. En el caso de flexión uniaxial se tenían dos ecuaciones de equilibrio, una para axiles (ec.21) y otra para momentos (ec.22). Para el caso de la flexión biaxial, se añade otra nueva ecuación de equilibrio de momentos, teniendo el siguiente conjunto de ecuaciones:

$$\boldsymbol{N = \Sigma N = \int_{A_C} \sigma_c \, dA_c + \int_{A_s} \sigma_s \, dA_s + \int_{A_p} \sigma_p \, dA_p}$$

$$\boldsymbol{M_x = \Sigma M = -\int_{A_C} \sigma_c \, y \, dA_c - \int_{A_s} \sigma_s \, y \, dA_s \; - \int_{A_p} \sigma_p \, y \, dA_p}$$

$$\boldsymbol{M_y = \Sigma M = \int_{A_C} \sigma_c \, x \, dA_c + \int_{A_s} \sigma_s \, x \, dA_s + \int_{A_p} \sigma_p \, x \, dA_p}$$

   Siendo A_s la armadura pasiva y A_p la armadura activa.

2. En el caso de flexión uniaxial, se vio que la posición de la fibra neutra se podía definir únicamente con una variable que era x. Para el caso de flexión biaxial, son necesarias dos variables para definir tal posición. Ver imagen 22, para el caso de sección rectangular.

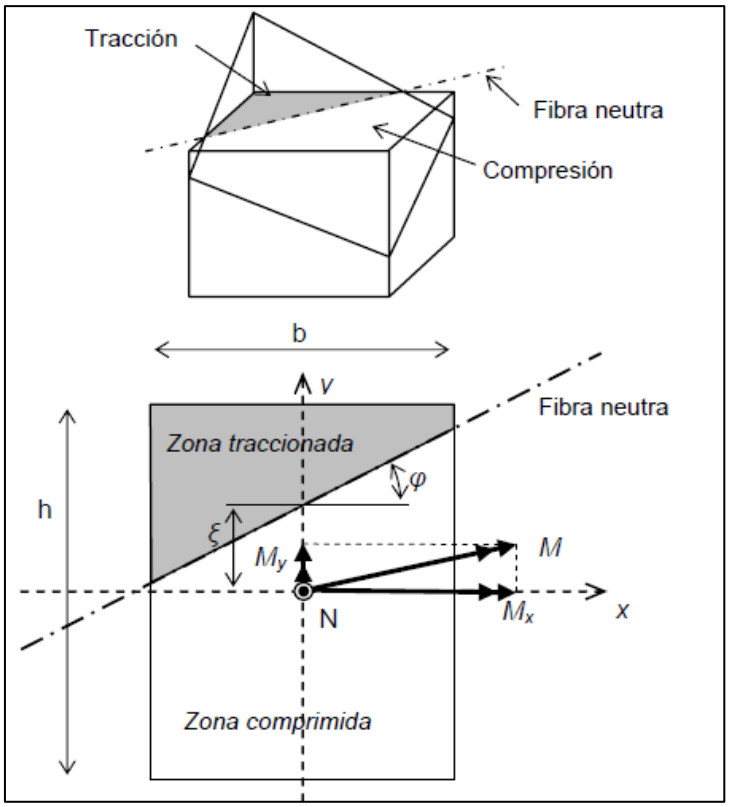

Figura 22. Caso de flexión biaxial en una sección rectangular

Las dos variables que definen la posición de la fibra neutra son: $\xi \; y \; \varphi$

A la hora de resolver nuestro problema, se ha definido el plano de deformación que, proyectándolo da lugar a una recta sobre la sección que es la línea neutra (L.N). Ver figura 22 parte de arriba. Dicho plano queda definido por tres parámetros que se tendrán que calcular para posteriormente obtener los esfuerzos. Tal plano, es el siguiente:

- **Se sabe que la ecuación de un plano general es:**

$$\boldsymbol{P\,(x,y) = a + b \cdot x + c \cdot y}$$

- **En nuestro caso, se define de la siguiente manera:**

$$\boldsymbol{\xi(x,y) = e_0 + ay \cdot x + ax \cdot y}$$

  Siendo:
  - e_0 → **deformación axial**
  - ay → **giro en torno al eje y**
  - ax → **giro en torno al eje x**
- **De tal manera que, si se consigue:**

$$\boldsymbol{\xi(x,y) = 0}$$

  Se obtiene la línea neutra en un plano de 1D.

En la imagen 23, se observa lo explicado para aclararlo, en caso de que hubiese alguna duda.

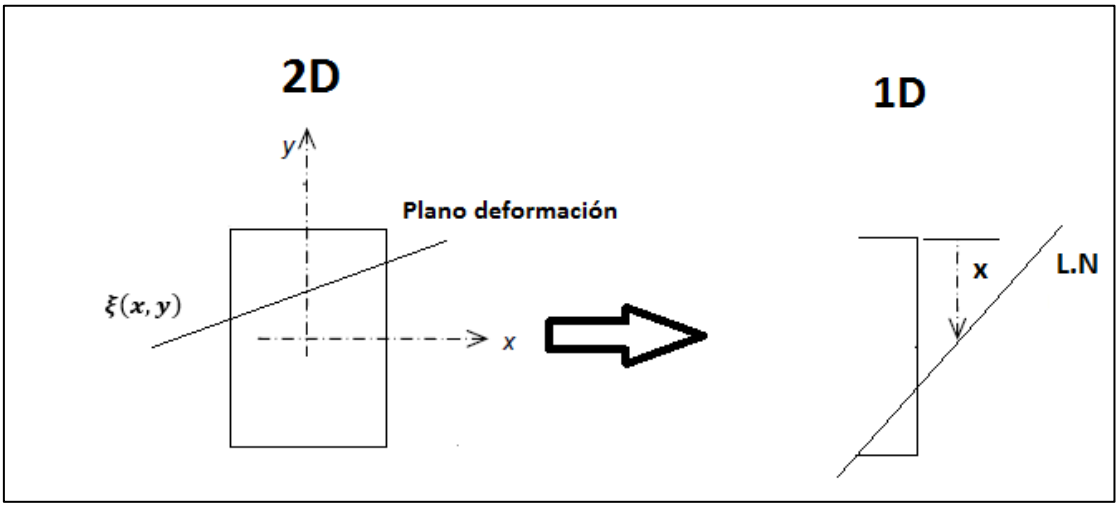

Figura 23. Representación obtención línea neutral en nuestro problema

En nuestro problema se trabajará con el plano de deformación y las incógnitas e_0,ay, ax para la obtención de los esfuerzos de comprobación. Lo que se realizará para su obtención es una continua iteración, por medio de MATLAB y la instrucción *fsolve*, para la obtención de las tres incógnitas. Más adelante se explicará con mayor detalle.

3. Para el caso de flexión biaxial, suele ser necesario disponer de armaduras en cada una de las caras de la sección.

3.5.1 Problema de comprobación

En este caso, el problema de comprobación biaxial está basado en determinar si una sección transversal totalmente definida resiste una terna de esfuerzos (N_d, M_{dx}, M_{dy}).

Para este tipo de problema estarán definidos los siguientes parámetros:

- La sección, es decir, su forma geométrica.
- El área de la armadura (A_s) y la posición que presentan las armaduras en la sección.
- Las acciones que actúan sobre la sección (N_d, M_{dx}, M_{dy}).

Los valores de las tensiones en el acero y en el hormigón se obtienen a partir de la posición de la fibra neutra, que ya se ha explicado anteriormente cómo se obtiene.

A la hora de abordar la resolución de este tipo de problema, hay que decir que no admite solución analítica exacta, debiendo recurrirse a métodos aproximados. Por lo tanto, lo que se suele hacer es tomar dos valores de la terna de esfuerzos (en nuestro caso, tomaremos N_d, M_{yd}) e introducirlos en las ecuaciones de equilibrio correspondientes, de tal manera que se obtenga la posición del plano de deformación o fibra neutra (Mansilla, 2018). Teniendo estos valores se introducirán en una tercera ecuación, de la que se obtendrá un momento máximo, que para nuestro caso será (M_{xmax}). Este momento obtenido habrá que compararlo con el momento de la acción de entrada correspondiente, pudiendo darse dos opciones:

1. $M_{xmax} \geq M_{xd}$ → La sección resistirá
2. $M_{xmax} \leq M_{xd}$ → La sección no aguantará.

Aunque posteriormente se verá, la tercera ecuación que se introduce para la resolución del problema en nuestro caso particular, es la siguiente:

$$\varepsilon_{max} = \varepsilon_{cu2} = 0.0035$$

De la misma manera que en el caso uniaxial, hay diagramas de interacción formados por las ternas de esfuerzos N, M_x, M_y, que en este caso por ser tridimensional serán superficies, de manera que bastará con comprobar que la terna N_d, M_{dx}, M_{dy} cae dentro del diagrama de interacción. La obtención de estos diagramas es, si cabe, más costosa que la obtención del diagrama de interacción para el caso de flexión uniaxial, ya que en este caso depende de 3 variables (ver figura 24):

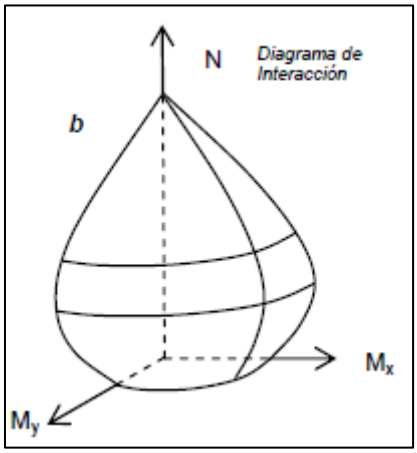

Figura 24. Diagrama de interacción caso biaxial

Este es el método utilizado en nuestro problema, explicado de modo general. Ahora, se va a entrar más en detalle para explicar determinadas formas de análisis que se utilizan en este problema en particular.

3.5.2 Ecuaciones de equilibrio

Se tiene una sección transversal de hormigón de una forma aleatoria cualquiera y con una distribución de armaduras, sometida a una solicitación de flexión compuesta esviada (Torrano & Martí).

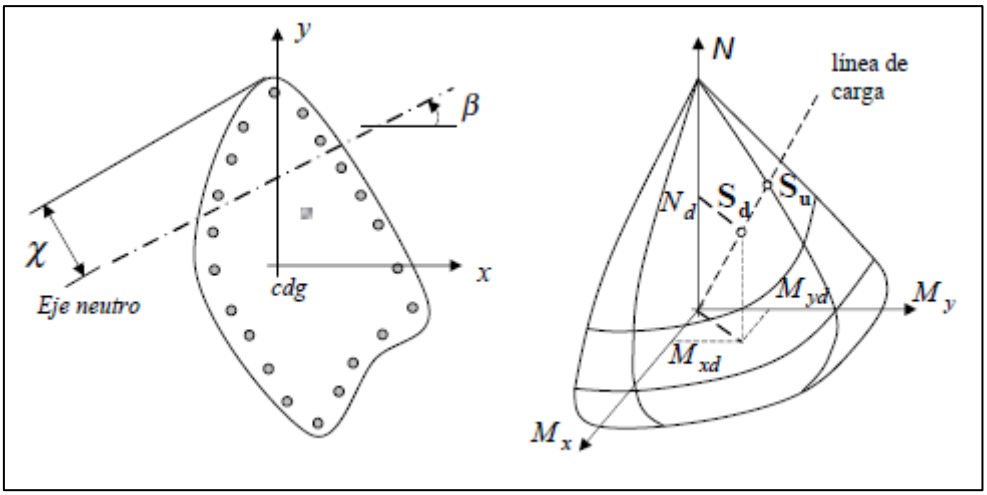

Figura 25. Sección de forma cualquiera sometida a flexión biaxial y diagrama de interacción

Las ecuaciones de equilibrio que se tienen son:

$$\boldsymbol{N = \iint_S \sigma_c(\varepsilon_c) ds + \sum_{j=1}^{n} \sigma_{sj}(\varepsilon_{sj}) A_j - \sum_{j=1}^{n} \sigma_{cj}(\varepsilon_{cj}) A_j}$$

$$\boldsymbol{M_x = \iint_S \sigma_c(\varepsilon_c) y ds + \sum_{j=1}^{n} \sigma_{sj}(\varepsilon_{sj}) y_j A_j - \sum_{j=1}^{n} \sigma_{cj}(\varepsilon_{cj}) y_j A_j} \qquad \text{(ec25)}$$

$$\boldsymbol{M_y = \iint_S \sigma_c(\varepsilon_c) x ds + \sum_{j=1}^{n} \sigma_{sj}(\varepsilon_{sj}) x_j A_j - \sum_{j=1}^{n} \sigma_{cj}(\varepsilon_{cj}) x_j A_j}$$

Siendo:

- N, Mx, My → Terna de esfuerzos, dato de entrada
- σ_c y σ_{cj} → Tensión en el hormigón y tensión del hormigón en la posición del redondo j, respectivamente.
- σ_{sj}→ Tensión en el redondo j
- ε_c y ε_{cj} → Deformación del hormigón y deformación del hormigón en la posición del redondo j, respectivamente.
- ε_{sj}→ Deformación del redondo j.
- x, y → Coordenadas de un punto de la sección
- x_j, y_j → Coordenadas del redondo j.
- A_j → Área del redondo j.
-
- n → Número de redondos de la sección

Se consideran positivos aquellas solicitaciones que producen compresión.

3.5.3 Método de los elementos finitos (MEF)

El método de los elementos finitos (MEF), es un método numérico utilizado para la resolución de problemas de ingeniería y matemática física. Fue desarrollado, debido a que en determinados problemas complejos, no es posible la obtención de soluciones analíticas exactas. Sin embargo, no deja de ser un método aproximado de cálculo. Consiste en simplificar la geometría del objeto continuo mediante la subdivisión del mismo en elementos discretos. Estos elementos, se denominan elementos finitos y están conectados por medio de nodos. La solución al problema es la combinación de las soluciones de cada uno de los nodos. De esta manera, las ecuaciones diferenciales que describen el sistema, tienen un número finito de variables. Dichas ecuaciones deben satisfacer las condiciones de equilibrio y compatibilidad (Jiménez Montoya, García Meseguer, & Morán Cabré, 1987).

En la figura 26, se observa la discretización de un sistema continuo. Cada triángulo en que se ha subdividido el sistema continuo es un elemento finito que forma la malla. Los vértices de cada triángulo son los nodos.

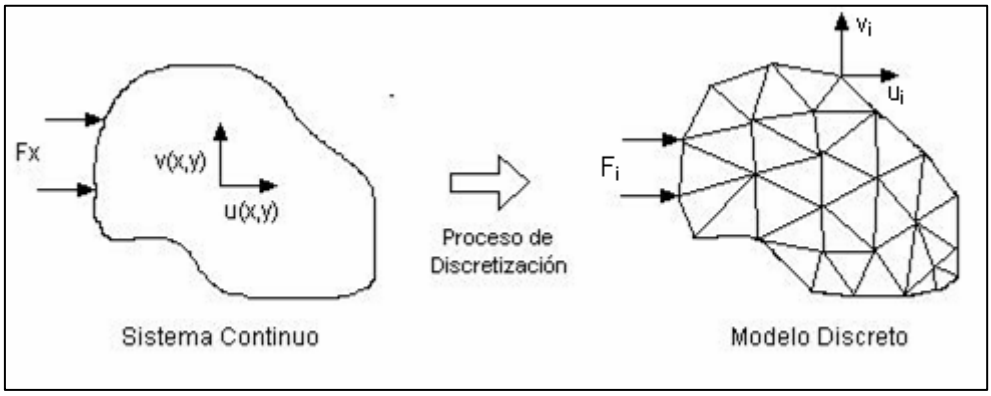

Figura 26. Discretización triangular de sistema continuo en sistema discreto

De suma importancia es la calidad de la malla generada, debido a que si no fuese lo suficientemente buena, los cálculos podrían no ser muy exactos por no llegar a converger. Cuanto menos se subdivida el sistema, más exacta será la solución. De forma general, el mallado suele ser rectangular o triangular para elementos 2-D, como es nuestro caso.

Esta técnica es muy utilizada en diferentes campos: cálculo estructural, transmisión de calor, fluido mecánica, etc. La resolución por MEF suele ser muy sistemática, por lo que este método se ayuda de softwares y computadoras diseñados para ello.

Es aquí donde entra en juego nuestro problema, ya que la integración de las tensiones en el hormigón, se realiza sobre la sección discretizada. Para realizar la discretización de la sección, hay diversas opciones. Por ello, en primer lugar se va a hablar de las características de las mallas, después se comentarán las propiedades de los elementos y por último se nombrarán los algoritmos de generación de mallas, explicándose el método escogido.

1. <u>Mallado</u>

La generación de la malla es un punto clave, ya que para geometrías complejas requiere un tiempo importante y no se trata de una operación trivial (Castillo, 2007).

Por otro lado, la malla debe estar correctamente diseñada, ya que la calidad de los resultados depende de la calidad de aquella.

A continuación, se van a ver los tipos de mallas:

- Malla conforme/no conforme. En una malla conforme los elementos adyacentes comparten nodos o caras. Ver figura 27.

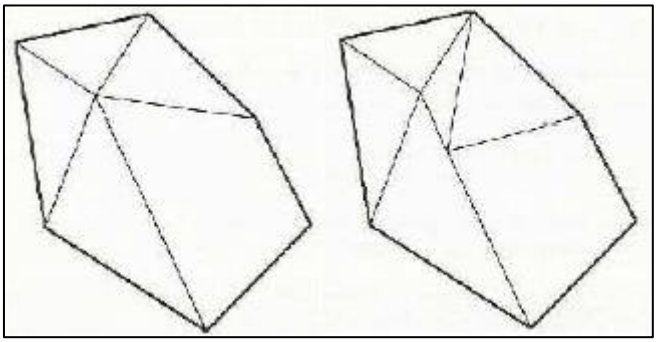

Figura 27. Malla conforme (izquierda). Malla no conforme (derecha)

- Malla estructurada/no estructurada. En una malla estructurada, cada nodo del interior es compartido por el mismo número de elementos. Ver figura 28.

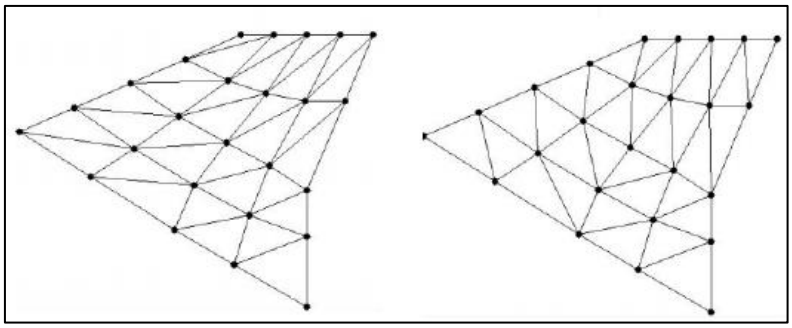

Figura 28. Malla estructurada (izquierda). Malla no estructurada (derecha)

2. Propiedades de los elementos finitos

Se va a diferenciar entre dos tipos distintos de propiedades: propiedades geométricas y propiedades físicas.

- Propiedades geométricas:
  - La variación del tamaño entre elementos adyacentes debe ser progresiva.
  - La densidad de elementos en algunas regiones de la malla debe ser más alta.
  - En las mallas de elementos triangulares se deben evitar ángulos obtusos.
  - En general, los elementos deben ser suficientemente regulares y satisfacer ciertas propiedades relativas a su forma.
- Propiedades físicas:
  - Puede haber ciertos aspectos físicos del problema que condicionen la geometría de los elementos.

3. Algoritmos de generación de mallas

Los diferentes métodos que hay para generar las mallas son:

- Manuales.
- Métodos basados en la transformación de un dominio con geometría simple.

- Métodos basados en la solución de un sistema de ecuaciones en derivadas parciales
- Métodos basados en la deformación y modificación local de una malla sencilla.
- Métodos basados en la composición de mallados de subconjuntos del dominio a mallar.
- Métodos automáticos que obtienen la malla final, elemento por elemento, por medio de la definición del contorno:

    - Métodos de avance frontal

    - Algoritmos basados en la construcción de Voronoi-Delaunay. Este caso es el seleccionado para realizar la discretización de las secciones en nuestro proyecto. Una triangulación de Delauny, verifica que las circunferencias circunscritas a cada triángulo no contienen vértices de otros elementos (Morcillo, 2000).

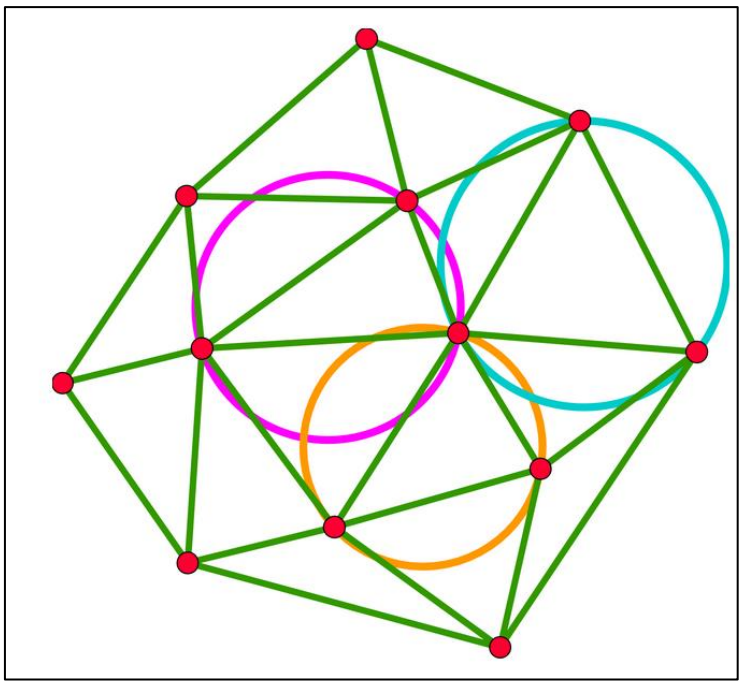

Figura 29. Triangulación Delaunay

Otra propiedad que presenta este tipo de triangulación es, que uniendo los centros de las circunferencias circunscritas a todos los triángulos que comparten un vértice, se obtienen polígonos de Voronoi.

Las triangulaciones de Delaunay son ampliamente utilizadas en la informática científica. Lo que hace tan útil este tipo de triangulación son las propiedades geométricas tan favorables que presenta.

Dicha triangulación se verá posteriormente que es generada por medio de una orden en Matlab.

3.5.4 Integración numérica de elementos triangulares o Integración de Hammer

Las expresiones correspondientes a las características de los elementos finitos (rigidez, fuerzas, etc.) son del tipo integral, extendiéndose normalmente al volumen del elemento. Debido al cambio de coordenadas en la formulación iso paramétrica, estas integrales se extienden a dominios simples, normalmente entre -1 y +1. En todo caso la complejidad de las integrales a evaluar hace que su cálculo sólo se pueda realizar de forma numérica (Rathod, Nagaraja, & Venkatesudu).

Hay que hablar aquí, por lo tanto, de la cuadratura de Gauss. En este método se aproxima la función mediante un polinomio, pero no se especifica a priori la posición de los puntos empleados para definirlo. La posición de estos puntos se determina con la condición de alcanzar la mayor precisión posible de la integral. Considerando que la integral se aproxima mediante una expresión como la siguiente:

$$I = \int_{-1}^{+1} f(\xi) d\xi = \sum_{i=1\ldots n} H_i \, f(\xi_i)$$

Resulta que se dispone de $2n$ parámetros a definir (Hi, ξi), por lo que puede definirse un polinomio de grado $2n-1$ y efectuar su integral de forma exacta.

En consecuencia, una regla de este tipo con n puntos integra de forma exacta un polinomio de grado $2n-1$ y el error es del orden $O(h^{2n})$.

Este método particular suele ser conocido como cuadratura de Gauss-Legendre. En el análisis por el método de los elementos finitos, los cálculos más complicados son los relativos a la evaluación del integrando, ya que éste involucra el cálculo de la matriz, la jacobiana de la transformación de coordenadas, su determinante… Por lo tanto, el procedimiento de Gauss es idealmente el más favorable, puesto que requiere un menor número de evaluaciones de la función, y eso hace que sea el más utilizado.

Para nuestro caso particular es necesario aplicar la integración en regiones triangulares, que expresa las integrales en función de las coordenadas de área de la siguiente forma (Zienkiewicz):

$$I = \int_{0}^{1} \int_{0}^{1-L} f(L_1, L_2, L_3) \; dL_1 dL_2$$

Resolviéndose por medio de las fórmulas desarrolladas por Hammer y obteniéndose lo siguiente:

$$I = \int_{0}^{1} \int_{0}^{1-L} f(L_1, L_2, L_3) \; dL_1 dL_2 = \sum_{i=1\ldots n} W_i \, f(L_{1i}, L_{2i}, L_{3i})$$

A continuación, se muestra una tabla con los puntos a usar y los factores de peso correspondientes:

Nº puntos (n)	Orden	Coordenadas	Pesos (Wi)
1	1	[1/3, 1/3, 1/3]	0.5
3	2	[1/2, 1/2, 1/2 1/2, 1/2, 1/2 1/2, 1/2, 1/2]	1/6 1/6 1/6
4	3	[1/3, 1/3, 1/3 0.6, 0.2, 0.2 0.2, 0.6, 0.2 0.2, 0.2, 0.6]	-27/96 25/96 25/96 25/96

Tabla 6. Puntos y factores de peso de la integración de Hammer

El método que se ha optado por utilizar para la resolución de este problema, ha sido la integración de las tensiones mediante integrales de superficie. De tal manera que, las integrales de superficie expresadas en las ecuaciones de equilibrio (ec.25), en coordenadas locales (mediante transformación geométrica) quedan de la siguiente manera:

$$\iint_S \boldsymbol{\sigma_c}(\boldsymbol{\varepsilon_c})\boldsymbol{ds} = \sum_{k=1}^{ne}\left(\sum_{j=1}^{n}\sum_{i=1}^{n} \boldsymbol{w_i w_j}\, \boldsymbol{F}(\boldsymbol{\xi},\boldsymbol{\eta_j})|\boldsymbol{J}\,(\boldsymbol{\xi},\boldsymbol{\eta_j})|\right)$$

$$\iint_S \boldsymbol{\sigma_c}(\boldsymbol{\varepsilon_c})\, \boldsymbol{x}\, \boldsymbol{ds} = \sum_{k=1}^{ne}\left(\sum_{j=1}^{n}\sum_{i=1}^{n} \boldsymbol{w_i w_j}\, \boldsymbol{G}(\boldsymbol{\xi},\boldsymbol{\eta_j})|\boldsymbol{J}\,(\boldsymbol{\xi},\boldsymbol{\eta_j})|\right)$$

$$\iint_S \boldsymbol{\sigma_c}(\boldsymbol{\varepsilon_c})\, \boldsymbol{y}\, \boldsymbol{ds} = \sum_{k=1}^{ne}\left(\sum_{j=1}^{n}\sum_{i=1}^{n} \boldsymbol{w_i w_j}\, \boldsymbol{H}(\boldsymbol{\xi},\boldsymbol{\eta_j})|\boldsymbol{J}\,(\boldsymbol{\xi},\boldsymbol{\eta_j})|\right)$$

Siendo:

ne→ Número de elementos en que se ha discretizado la sección

$F(\xi,\eta)$ → Integrando de la función $\iint_S \sigma_c(\varepsilon_c)ds$, expresada en coordenadas naturales

$G(\xi,\eta)$ → Integrando de la función $\iint_S \sigma_c(\varepsilon_c)\, y\, ds$, expresada en coordenadas naturales

$H(\xi,\eta)$ → Integrando de la función $\iint_S \sigma_c(\varepsilon_c)\, x\, ds$, expresada en coordenadas naturales

$w_i\ y\ w_j$→ Los pesos de Gauss correspondientes a las posiciones i y j, en las direcciones ξ,η.

n → Es el número de puntos de Gauss en cada sección

|J| → Determinante del jacobiano de la transformación que tiene lugar

A continuación, se comenta la transformación que ha tenido lugar para poder realizar la integral de la sección, ver figura 30:

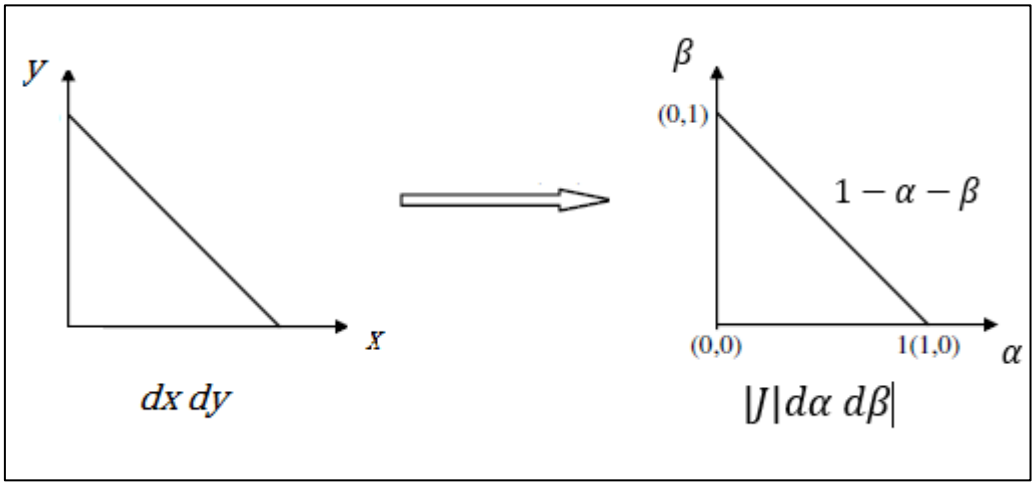

Figura 30. Transformación de un triángulo cualquiera de la sección discretizada en el equivalente según integración Hammer

Las direcciones α y β son las direcciones equivalentes a las consideradas en la formulación (ξ, η).

Utilizando esta teoría y formulación, se pasarán a realizar los cálculos por medio del software de cálculo MATLAB. En Matlab, se implementará un código en el que se introducirán estas formulaciones para poder llevar a cabo la realización del problema de comprobación para una sección transversal de hormigón armado de forma cualquiera.

Capítulo 4

RESULTADOS. MATLAB APLICADO AL CÁLCULO ESTRUCTURAL

4 RESULTADOS. MATLAB APLICADO AL CÁLCULO ESTRUCTURAL

En primer lugar se hablará del software MATLAB para conocer sus aplicaciones y posibilidades. Posteriormente, se verá la manera en que se ha desarrollado el problema de comprobación a través de MATLAB, los resultados obtenidos y, por último, el desarrollo mediante GUIDE de una aplicación que permita la resolución del problema mediante la inserción de los datos necesarios de entrada, así como los resultados que se obtienen.

4.1 Introducción

MATLAB es un entorno de computación técnica que posibilita la ejecución del cálculo numérico y simbólico de una manera rápida y sencilla, acompañado de características gráficas y de visualización avanzadas adecuadas al trabajo científico e ingenieril. Este programa, es un entorno interactivo que sirve para el análisis y modelado que implementa un gran número de funciones para el trabajo en distintos campos de la ciencia.

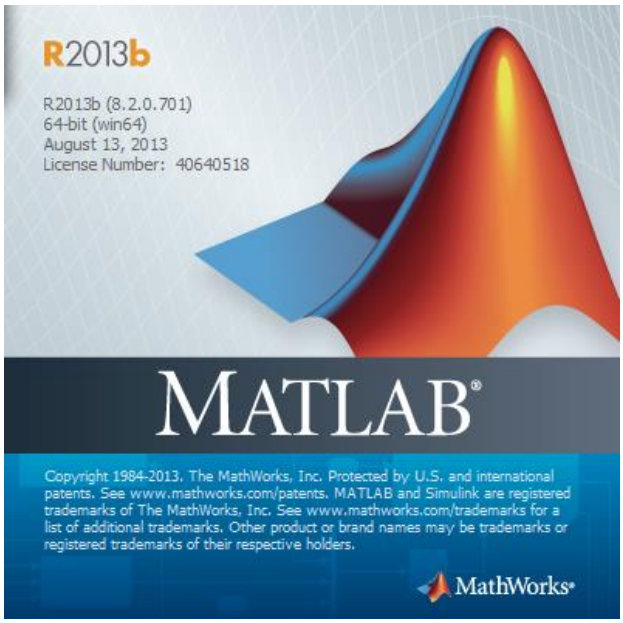

Figura 31. Interfaz inicial MATLAB 2013

MATLAB integra: análisis numérico, cálculo matricial, procesamiento de señales, gráficos; en un entorno de fácil utilización, donde los problemas y las soluciones son expresados tal cual se escriben matemáticamente, sin tener que utilizar la programación tradicional. El nombre MATLAB proviene de "MATrix LABoratory" y fue escrito originalmente para proporcionar acceso sencillo al software matricial desarrollado por los proyectos LINSPACK y EISPACK, que representan lo más avanzado en cuanto a programas de cálculo matricial.

El elemento básico de datos de MATLAB es una matriz que no precisa dimensionamiento. Esto conlleva la resolución de diversos problemas numéricos, con un ahorro notable de tiempo en comparación con diferentes leguajes de programación como C, BASIC o FORTRAM.

La evolución de dicho programa en los últimos años ha sido de gran envergadura, asentándose en los ámbitos universitarios como una de las principales herramientas en la introducción del álgebra lineal aplicada, así como en cursos más avanzados para desarrollar todo tipo de cálculos. Dentro del campo industrial, se utiliza para la

investigación, simulación y resolución de problemas prácticas, teniendo gran importancia en aplicaciones de control y procesamiento de señales. Dentro de MATLAB existen las TOOLBOXES, que son una serie de soluciones específicas incluidas dentro del propio programa y se utilizan para resolver diferentes tipos de problemas como: procesamiento de señales, diseño de sistemas de control, simulación de sistemas dinámicos, identificación de sistemas, redes neuronales, etc.

Una de las características más importantes y, por la cual es cada vez más utilizado, es la capacidad de crecimiento y adaptación al tiempo actual. Dicha característica permite a los usuarios convertirse en autores contribuyentes, creando aplicaciones. En resumen, las prestaciones más importantes que presenta MATLAB son:

- Escritura del programa en lenguaje matemático.
- Implementación de matrices como elemento básico de lenguaje, con lo que se consigue una reducción importante del código.
- Implementación de aritmética compleja.
- Un gran contenido de órdenes específicas, que se han comentado anteriormente, agrupadas en TOOLBOX.
- Posibilidad de ampliar y adaptar el lenguaje, mediante ficheros de scripts y funciones.
- Herramientas para crear aplicaciones con interfaces de usuario personalizadas.

4.2 Problema de comprobación. Resolución

En este punto se desarrollará la manera en que se ha llevado a cabo la resolución del problema de comprobación para una sección transversal de hormigón armado poligonal de forma cualquiera.

La resolución de este problema se basa en determinar la posición del eje neutro, para lo cual hacen falta 3 ecuaciones para obtener 3 incógnitas:

(1) N_d
(2) M_{yd}
(3) $\varepsilon_{max}=\varepsilon_{cu2}$

Una vez sean resuelto ese sistema de ecuaciones se tendrá la posición del eje neutro por medio de 3 variables (e0, ax, ay), que se utilizarán para obtener los valores de los esfuerzos, a partir del cual se comprobará el $M_{xd} = M_{xmax}$.

Tras esta primera rápida explicación se pasará a ver paso por paso como se ha llegado a la solución del problema. A su vez, se verán varios ejemplos de secciones trabajadas y los momentos de cálculo obtenidos.

En primer lugar, se va a comenzar haciendo una división, ya que este proyecto se ha realizado en dos partes. Por un lado se ha desarrollado la solución del problema de comprobación para una sección rectangular, mientras que por el otro lado se ha desarrollado la resolución del problema de comprobación para una sección de forma poligonal cualquiera (pentágono, hexágono, octógono, hasta el círculo).

4.2.1 Sección transversal rectangular

El modo de operar, para llevar a cabo la resolución del problema es el siguiente:

1. Datos de entrada. Se introducen los datos de entrada necesarios. En este caso:

- Dimensiones de sección: *b* (ancho) y *h* (canto).
- Características mecánicas del hormigón y el acero: *fck*, *fyk* y E_s.
- Características geométricas de las armaduras: número de armaduras (ni), diámetro de los armados (di) y recubrimiento mecánico (ri)

2. Definición de la armadura. En esta sección se definen las armaduras de las cuatro caras que forman la sección: número de armaduras (ni), diámetros (di), recubrimiento mecánico (ri), distancia entre armados y posición de cada armadura.

3. Graficar sección con armadura. Salida por pantalla de la sección transversal con sus respectivos armados en las posiciones definidas anteriormente.

4. Discretización de la sección por medio de la discretización de Delaunay.

5. Inserción de las acciones que actúan sobre la sección.

6. Cálculos y obtención de resultados. Integración Hammer, obtención de la posición del eje neutro y posterior obtención de los esfuerzos de comprobación.

A continuación, se irá explicando más en detalle cada uno de los 6 puntos expuestos arriba:

1. Datos de entrada. Para comenzar con la resolución del problema, es necesario disponer de unos datos de entrada.
   Por ello, para el problema de comprobación los datos de entrada necesarios son:

   - *Definición de la sección*. Se definen las dimensiones de la sección rectangular, tanto el ancho como el canto, irán expresados en mm.

   - *Definición de las armaduras*. Se definirán todas las características geométricas necesarias de la armadura y los armados. Las características que se piden como dato de entrada son:

     - Diámetro de los armados (di). La sección de los armados se puede obtener, pues al ser círculos, el área de cada armado es:

$$A_i = \pi * (di/2)^2$$

     - Recubrimiento mecánico (ri). El recubrimiento mecánico es la distancia que hay desde el centro de gravedad de la armadura hasta superficie de hormigón. Ver imagen 32.

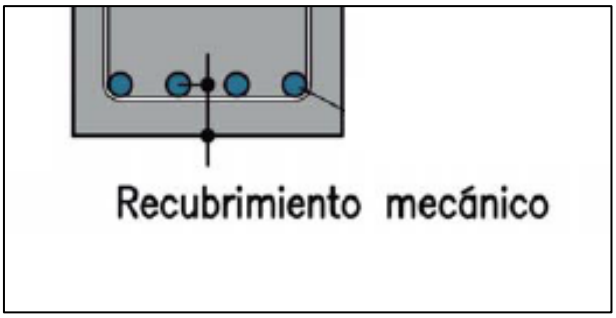

Figura 32. Ejemplo de recubrimiento metálico

- Número de armados por cara (ni). En nuestro caso se podrán poner armaduras en las 4 caras de la sección.

- Definición características mecánicas del hormigón y del acero de la armadura. Se definen 3 características:
  - Resistencia característica del hormigón: fck (MPa)
  - Resistencia características del acero: fyk (MPa)
  - Módulo de elasticidad: Es (MPa)

2. <u>Definición de la armadura.</u> Anteriormente se han definido los parámetros geométricos de la armadura como son: diámetro, recubrimiento mecánico y número de armaduras en cada cara.
   En este punto se definirán las posiciones que tendrá cada armado dentro de la sección y la distancia entre cada armado de cada cara.
   Se irán recorriendo las caras de la sección, definiendo las distancias entre armados de cada cara y se irá rellenando una matriz en la que se irán definiendo las posiciones de cada armado en particular, junto con su diámetro y recubrimiento mecánico.
   La distancia entre cada armado vendrá expresado con una relación de la siguiente forma:
   - Para la armadura superior e inferior:

$$\boldsymbol{da = \frac{(b - (2 * ri + ni * di))}{(ni - 1)}}$$

   - Para la armadura lateral derecha y lateral izquierda:

$$\boldsymbol{da = \frac{(h - (r1 + d1 + r2 + d2 + ni * di))}{(ni + 1)}}$$

   Siendo:
   - b,h → Ancho y canto de sección
   - ri, di, ni → recubrimiento mecánico, diámetro y número de armaduras de la cara
   - r1, d1 → recubrimiento mecánico y diámetro de armado de la armadura de la cara superior.

- r2, d2 → recubrimiento mecánico y diámetro de armado de la armadura de la cara inferior.

3. <u>Graficar sección con armadura.</u> Llegado a este punto en el que están definidos tanto la sección transversal, como las armaduras, así como las propiedades mecánicas de hormigón y acero, el siguiente paso consta simplemente de graficar la sección con la armadura para ver si hemos colocado de manera correcta y en su sitio, tanto la sección como la armadura.

   En primer lugar, se dibujará el rectángulo de manera muy simple. Sabiendo las dimensiones se dibujan cuatro líneas (dos horizontales y dos verticales) que definen la sección rectangular.

   Posteriormente, para representar los armados, se van representando de uno en uno, comenzando por la armadura superior, siguiendo con la armadura inferior, armadura izquierda para acabar con la armadura derecha. Esta forma de representar se consigue con un simple bucle for. La manera de representar los armados es la siguiente:

   - Teniendo la posición del centro de gravedad del armado
   - Dibuja un círculo de diámetro di alrededor de ese punto definido.

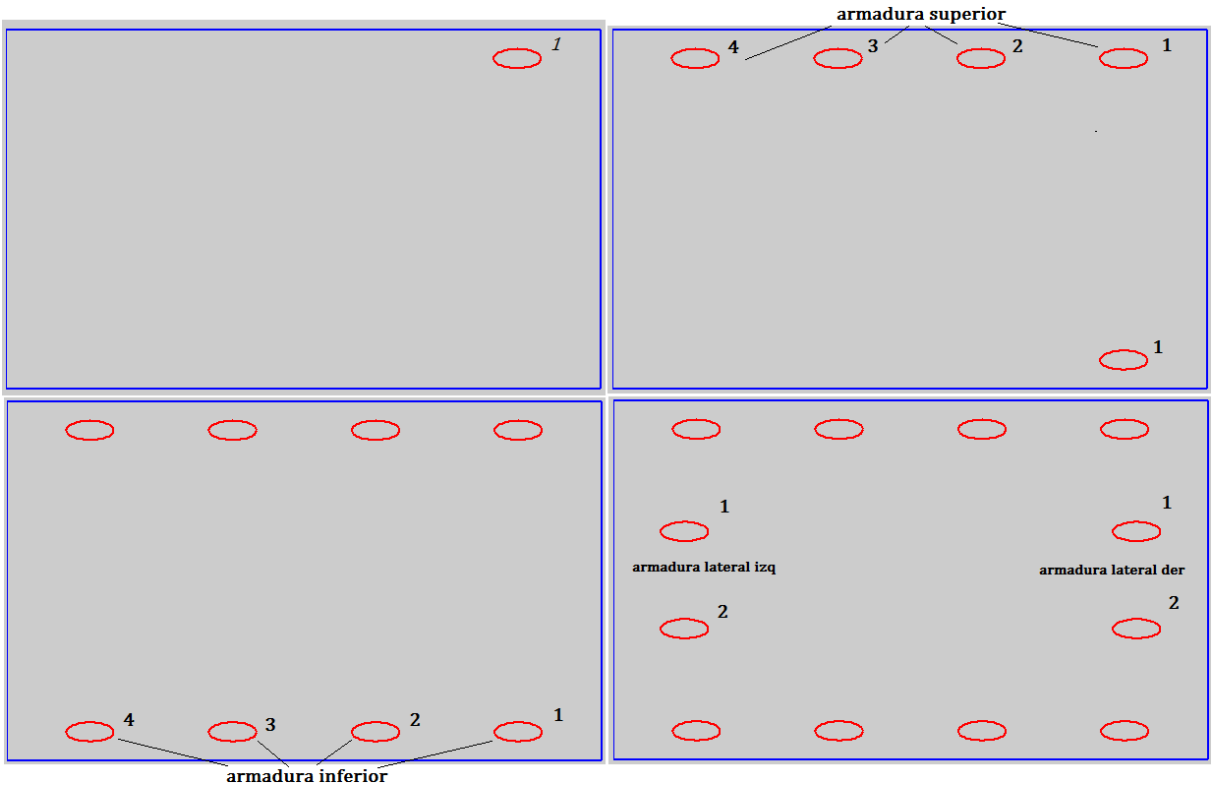

Figura 33. Desarrollo de cómo se va graficando la sección con la armadura. La secuencia sería izquierda arriba, derecha arriba, izquierda abajo y, por último, derecha abajo.

A continuación, se muestran unos ejemplos de la representación gráfica:

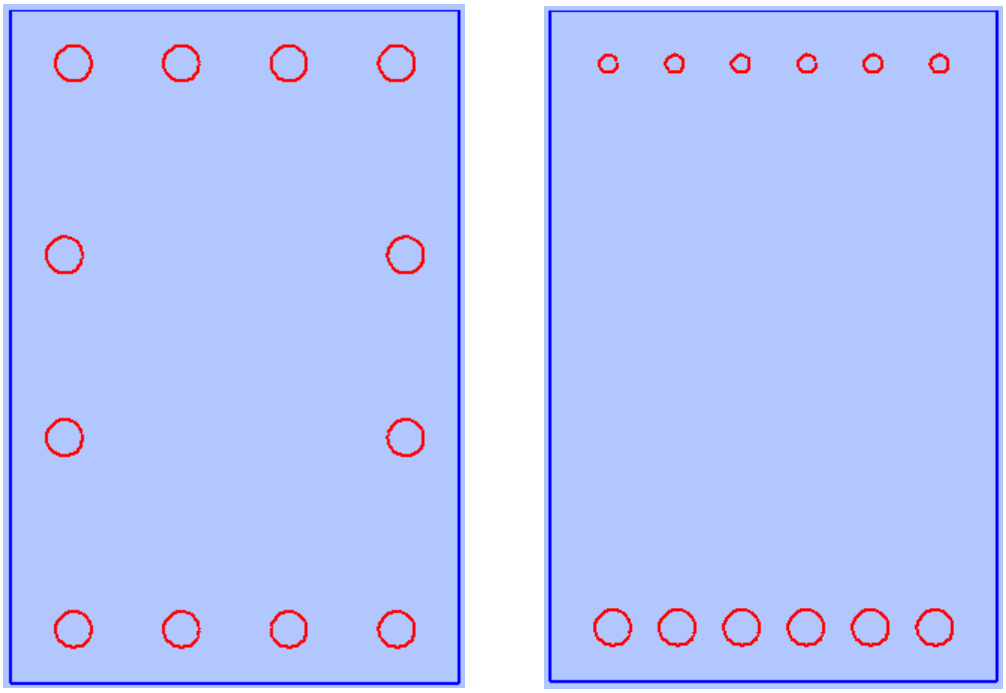

Figura 34. Sección rectangular, a la izquierda con armado en las 4 caras y a la derecha con armado en dos caras y con diámetros de armado diferentes

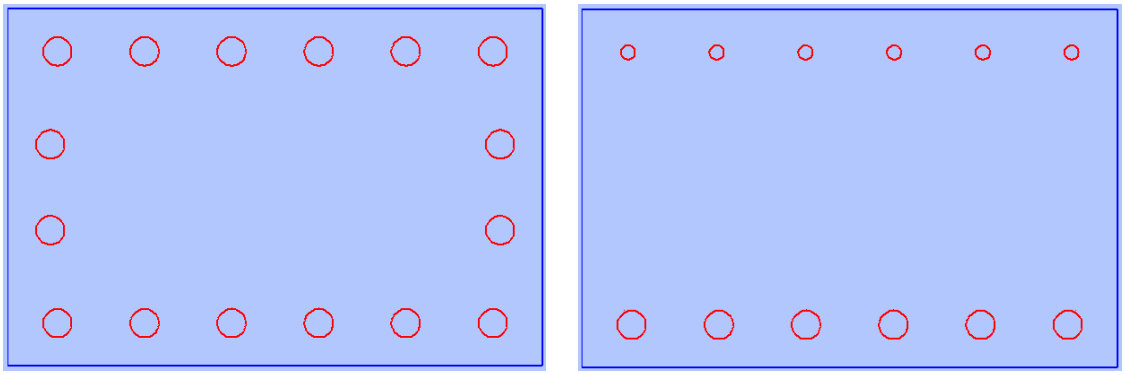

Figura 35. Mismo caso que el anterior pero cambiando las dimensiones del rectángulo

4. <u>Discretización de la sección de Delaunay.</u> Realizada la sección con el armado y teniéndolas ambas ya representadas, el siguiente paso que se lleva a cabo es la discretización de la sección para poder posteriormente realizar la integración de Hammer de la sección. Como ya se dijo anteriormente, se va a realizar la discretización de la sección por el método de Delaunay. La discretización de la sección, se realiza por medio de MATLAB, que presenta dos posibilidades para realizar la triangulación (The MathWorks, 2019):
    - Función Delaunay y delaunayn. Esta forma no es la que se ha utilizado para la discretización de la sección. Debido a que se utiliza en casos más simples donde únicamente se requieren datos básicos de triangulación y donde los datos son suficientemente completos para su aplicación.
    - La clase *delaunayTriangulation.* Esta clase es la utilizada en este proyecto para llevar a cabo la discretización y triangulación de la sección de estudio. Este tipo de función es útil cuando se va a realizar alguna de las siguientes operaciones:

- Búsqueda en la triangulación para triángulos o tetraedros que encierren un punto de consulta.
- Utilice la triangulación para realizar una búsqueda de puntos vecinal más cercana.
- Consulte la adyacencia topológica o las propiedades geométricas de la triangulación.
- Modifique la triangulación para insertar o quitar puntos.
- Restringir los bordes de la triangulación, esto se denomina triangulación de Delaunay restringida.
- Triangular un polígono y, opcionalmente, eliminar los triángulos que están fuera del dominio.
- Utilice la triangulación de Delaunay para calcular el casco convexo o el diagrama Voronoi.

En nuestro caso, algunas de estas operaciones como la búsqueda de puntos, elementos, posibles modificaciones de la triangulación,...

Por lo tanto, una vez explicado el comando para la obtención de la triangulación de la sección "*delaunayTriangulation",* se pasa a ver cómo se desarrolla la operación. En nuestro caso, se determinan las líneas de cuadrículas verticales y horizontales que aparecerán en el gráfico por medio de dos vectores. A continuación, con el comando *meshgrid,* se generan las coordenadas de la cuadrícula 2-D que están basadas en los dos vectores definidos con anterioridad. Por último, se hace la llamada al comando "*delaunayTriangulation",* se hace un *"triplot"* a través de MATLAB y posteriormente se definen los nodos y los elementos generados por la triangulación. Al final, se obtiene lo siguiente:

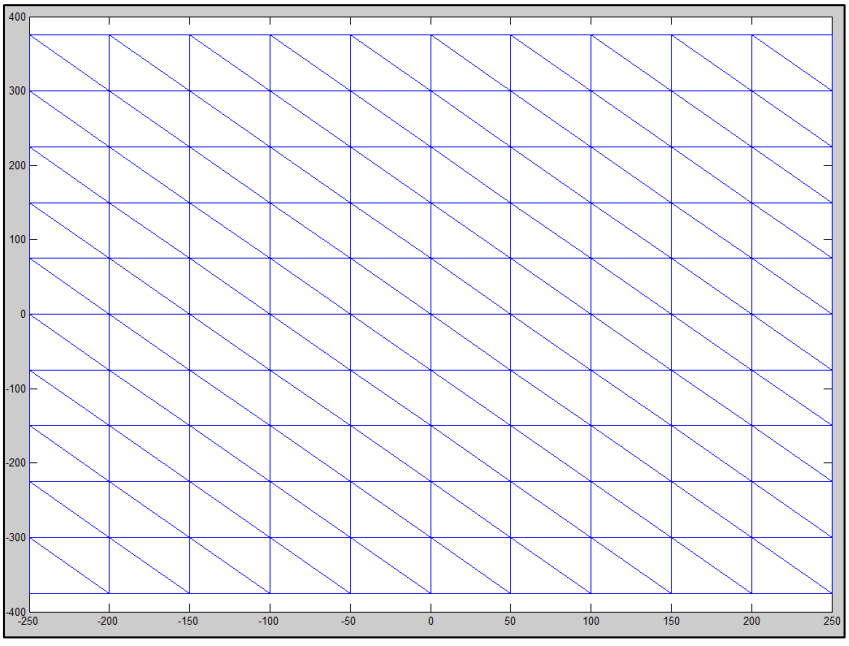

Figura 36. Discretización sección rectangular de 500 mm x 750 mm

5. <u>Inserción de las acciones que actúan sobre la sección</u>. En este punto, se añadirán las acciones que actúan sobre la sección que, a su vez, son los valores a partir de los que se llevará a cabo la posición del eje neutro y de los esfuerzos máximos. Se introducirán los valores de: N_d (KN), M_{yd} (KN*m) y M_{xd} (KN*m).
6. <u>Cálculos y obtención de resultados</u>. Esta es la parte del script en la que se realizaran los cálculos. Se desarrolla en dos partes:

   En primer lugar, por medio del comando "***fsolve***", se obtendrá la posición del eje neutro. A través del comando "fsolve", se resuelve un sistema de ecuaciones no lineales. La función en la que se desarrolla este cálculo, es la nombrada como "***calcula (s)***" y lo que se realiza es un proceso iterativo, con el cual se obtendrán los valores de la posición del eje neutro, definidos como e_0 (deformación axial), a_x (giro en torno al eje x) y a_y (giro en torno al eje y).

   El modo de operar es el siguiente:

   - Se toma como iteración inicial para la terna (e_0, a_x, a_y) = (0, 0, 0).
   - Se recorre cada elemento triangular, en que se ha discretizado la sección.
   - Se realizan los cálculos referentes al hormigón, considerando el modelo tensión deformación de parábola-rectángulo, con las siguientes expresiones:

$$\sigma_c = \begin{cases} f_{cd}\left[1-\left(1-\dfrac{\varepsilon_c}{\varepsilon_{c2}}\right)^n\right], para\ 0 \le\ \varepsilon_c\ \le\ \varepsilon_{c2} \\ f_{cd}, \qquad para\ \varepsilon_{c2}\ \le\ \varepsilon_c \le\ \varepsilon_{cu2} \end{cases}$$

   Donde:

$$n = \begin{cases} 2 & si\ f_{ck} < 50Mpa \\ 1.4 + 23.4\left(\dfrac{90-f_{ck}}{100}\right)^4, & si\ \ f_{ck} \ge 50Mpa \end{cases}$$

$$\varepsilon_{c2}(‰) = \begin{cases} 2 & si\ f_{ck} < 50Mpa \\ 2 + 0.085(f_{ck}-50)^{0.53}, & si\ \ f_{ck} \ge 50Mpa \end{cases}$$

$$\varepsilon_{cu2}(‰) = \begin{cases} 3.5 & si\ f_{ck} < 50Mpa \\ 2.6 + 35\left(\dfrac{90-f_{ck}}{100}\right)^4, & si\ \ f_{ck} \ge 50Mpa \end{cases}$$

   En este punto, también se aplica la integración de Hammer vista anteriormente. Los parámetros de la integración de Hammer definidos son los propios de un sistema de orden 2.

Pesos	Coordenadas
[1/2, 1/2, 1/2	1/6
1/2, 1/2, 1/2	1/6
1/2, 1/2, 1/2]	1/6

Tabla 7. Parámetros para integración de Hammer utilizados en el script

- Se obtiene, también, la aportación del acero de la armadura. Para el acero, se considera el modelo bilineal sin endurecimiento. Ver figura 37.

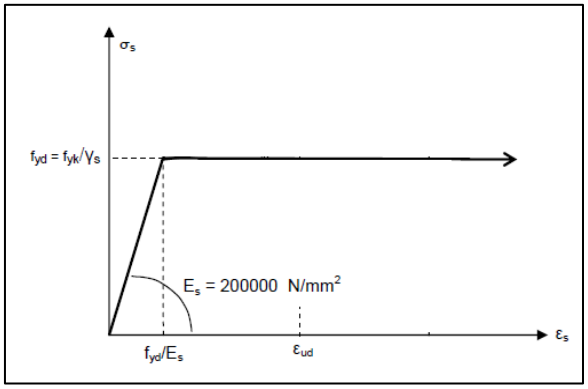

Figura 37. Diagrama empleado para el acero

De tal manera que:

- Para el caso positivo:
  - Si $\varepsilon_s < \varepsilon_y \rightarrow \sigma_s = \varepsilon_s * E_s$
  - Si $\varepsilon_s > \varepsilon_y \rightarrow \sigma_s = f_{yd}$
- Para el caso negativo:
  - Si $\varepsilon_s > -\varepsilon_y \rightarrow \sigma_s = \varepsilon_s * E_s$
  - Si $\varepsilon_s < \varepsilon_y \rightarrow \sigma_s = -f_{yd}$

Como parte final de esta función "calcula", se realiza la comprobación, para los valores obtenidos, por medio de la terna (N_d, M_{yd}, ε_{max}). Cuando los valores obtenidos para (e_0, a_x, a_y), proporcionen unos valores de (N_d, M_{yd}, ε_{max}), idénticos a los introducidos al principio, se habrá obtenido la solución final para la posición del eje neutro. Se van a ver, seguidamente, los resultados para la posición de eje neutro, en función de los valores probados para la terna (N_d, M_{yd}, ε_{max}), teniendo en cuenta, la sección con su correspondiente armadura, para un par de ejemplos:

Caso 1. Sección rectangular de 750x500mm², con las siguientes características:

- Características mecánicas del hormigón: fck=25 MPa
- Características mecánicas del acero: fyk=500 MPa y $E_s=2*10^5$ MPa
- Características de los armados:

- Armado superior: d_i= 20 mm , r_i= 60 mm, número de armados= 4
- Armado inferior: d_i= 20 mm, r_i= 60 mm, número de armados= 4
- Armado lateral izquierdo: d_i= 20 mm, r_i= 60 mm, número de armados= 2
- Armado lateral derecho: d_i= 20 mm, r_i= 60 mm, número de armados= 2

- Valores de la terna: N_d=2000 KN, M_{yd}=200 KN*m y ε_{max}= 0.0035.

Resultados. Se obtiene como resultados para la terna:

$$(e_0, a_x, a_y) = (-2.613*10^{-4}, 1.162*10^{-5}, 2.28*10^{-6})$$

Considerando la deformación axial positiva hacia arriba y giros positivos en sentido anti horario.

Caso 2. Sección rectangular de 1000x500mm^2, con las siguientes características:

- Características mecánicas del hormigón: fck=30 MPa
- Características mecánicas del acero: fyk=500 MPa y E_s=2*10^5 MPa
- Características de los armados:
  - Armado superior: d_i= 15 mm , r_i= 60 mm , número de armados= 4
  - Armado inferior: d_i= 20 mm, r_i= 60 mm, número de armados= 4
  - Armado lateral izquierdo: d_i= 15 mm, r_i= 60 mm, número de armados= 2
  - Armado lateral derecho: d_i= 15 mm, r_i= 60 mm, número de armados= 2
- Valores de la terna: N_d=1500 KN, M_{yd}=350 KN*m y ε_{max}= 0.0035

Resultados. Se obtiene como resultados para la terna:

$$(e_0, a_x, a_y) = (-0.0016, 1.5408*10^{-5}, 2.5384*10^{-6})$$

Considerando la deformación axial positiva hacia arriba y giros positivos en sentido anti horario.
Se requiere de alrededor de 20 iteraciones para llegar al resultado final. Esto sirve para hacerse una idea de lo costoso que supondría hacer estos cálculos a mano.

En segundo lugar, se procede a la obtención de los esfuerzos de comprobación. Habiendo obtenido la posición del eje neutro, se van a obtener por medio de otra función, denominada en el script "esfuerzos", los propios esfuerzos que van a servir de comprobación para poder analizar si la función resiste o no, las acciones introducidas al comienzo del problema. La función "esfuerzos", es como la función "calcula", pero en este caso ya no hay que realizar iteración alguna pues se tienen los valores de las acciones aplicadas sobre la sección y la posición del eje neutro.

Por lo tanto, utilizando los valores de la terna (e_0, a_x, a_y), se obtiene, primeramente, la aportación del hormigón recorriendo cada uno de los elementos triangulares de la sección y aplicando integración de Hammer. Después, se obtiene la aportación del acero. Juntándolo, se obtendrán los valores de los esfuerzos de comprobación, siendo de mayor utilidad para la comprobación el valor M_{xd} de la terna de esfuerzos obtenidos. A través del valor M_{xd}, se tiene el criterio de si la sección resistirá o no dichas acciones aplicadas.

Se van a ver los resultados obtenidos, para los dos ejemplos realizados anteriormente:

Caso 1. Introducidos los valores de las acciones aplicadas:

$$(N_d, M_{xd}, M_{yd}) = (2000, 500, 200)$$

Habiendo obtenido los valores para la posición del eje neutro:

$$(e_0, a_x, a_y) = (-2.613*10^{-4}, 1.162*10^{-5}, 2.28*10^{-6})$$

Los valores que resultan para los esfuerzos son:

$$(N_d, \mathbf{M_{xmax}}, M_{yd}) = (2000, \mathbf{526.17}, 200)$$

Por tanto, la conclusión que se puede extraer de este caso es que, la sección sí resistirá las acciones aplicada debido a que:

$$\mathbf{M_{xmax} > M_{xd}}$$

Caso2. Introducidos los valores de las acciones aplicadas:

$$(N_d, M_{xd}, M_{yd}) = (1500, 700, 350)$$

Habiendo obtenido los valores para la posición del eje neutro:

$$(e_0, a_x, a_y) = (-0.0016, 1.5408*10^{-5}, 2.5384*10^{-6})$$

Los valores que resultan para los esfuerzos son:

$$(N_d, \mathbf{M_{xmax}}, M_{yd}) = (1500, \mathbf{517.5}, 350)$$

Por tanto, la conclusión que se puede extraer de este caso es que, la sección no resistirá las acciones aplicada debido a que:

$$\mathbf{M_{xmax} < M_{xd}}$$

Con esto, ya se han podido ver dos ejemplos, uno en el cual la sección resiste y otro para el cual, la sección no resiste.

4.2.2 Sección transversal poligonal cualquiera

Como en el apartado anterior, la resolución de este problema consta de una serie de pasos, que se van a detallar, a continuación:

1. Datos de entrada. Se introducen los datos de entrada necesarios. En este caso, va a ser una sección poligonal que se va a definir, en función del diámetro del círculo circunscrito a esa sección, por lo tanto, se necesita:
    - Diámetro del círculo circunscrito a la sección: *r* (mm).
    - Número de lados de la sección: *na.*

- Características mecánicas del hormigón y el acero: *fck*, *fyk* y E_s.
- Características geométricas de las armaduras: diámetro de los armados (di) y recubrimiento mecánico (ri).

2. Definición de la armadura.

3. Graficar sección con armadura. Salida por pantalla de la sección transversal con sus respectivos armados en las posiciones definidas.

4. Discretización de la sección por medio de la discretización de Delaunay.

5. Inserción de las acciones que actúan sobre la sección.

6. Cálculos y obtención de resultados. Integración Hammer, obtención de la posición del eje neutro y posterior obtención de los esfuerzos de comprobación.

Se van a explicar más en detalle, los puntos 2, 3 y 4, ya que son los que presentan más cambios con respecto a la resolución del problema de la sección rectangular. El resto de puntos, tanto la discretización, como la inserción de las acciones y, por último, los cálculos y obtención de los resultados, se van a realizar de la misma forma que en el caso anterior. Lo único que cambia es la sección en la que se realiza la integración de Hammer.

Por lo tanto, se procede a explicar los puntos dichos:

2. <u>Definición de la armadura.</u> Para este caso de una sección de un número de lados definida, no es fácil, situar en los puntos correctos y a las distancias adecuadas los armados que conforman la armadura de la sección de estudio de hormigón armado.
   En el caso de la sección rectangular, era más sencillo ubicar los armados porque únicamente, hay que situar una línea paralela (vertical y horizontal) a la superficie exterior de la sección para ubicar los armados y definir los puntos donde se va a ubicar. Realizar eso con MATLAB, resulta relativamente sencillo.
   Pero en el caso actual, de una sección de un número de lados cualesquiera, al comienzo no se encontraba una manera fácil de situar los armados en las posiciones adecuadas. Para un radio de círculo constante, no éramos capaces de colocar en su debida posición y a la distancia especificada, los armados en todos los tipos de polígonos. Como demostración, se adjuntan imágenes de lo que sucedía, ver figura 37:

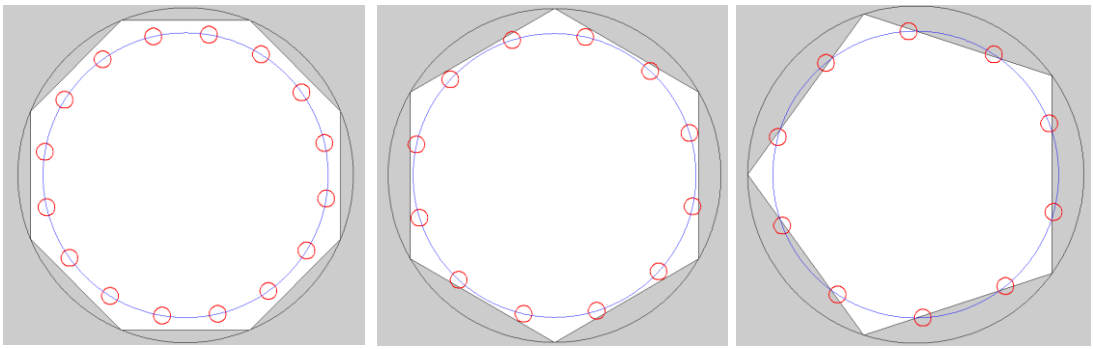

Figura 38. Representación de la armadura inicialmente

El problema que había era que, como la posición de los círculos pequeños, que representan los armados, se obtenía por medio de la distancia entre dos círculos concéntricos, cuanto menor era el número de lados de la sección, más se distorsionaba esa medida respecto la medida requerida. Si se aumentaba el número de lados de la sección, ese defecto cada vez se hacía menos visible.

Es por eso, que hubo que buscar otra manera para definir la armadura con sus posiciones. La forma que se encontró para resolver este escollo, fue la de hacer un polígono inscrito a otro polígono de manera que, los lados paralelos de los polígonos estén separados una distancia *ri* (recubrimiento mecánico). Ver imagen explicativa, a continuación:

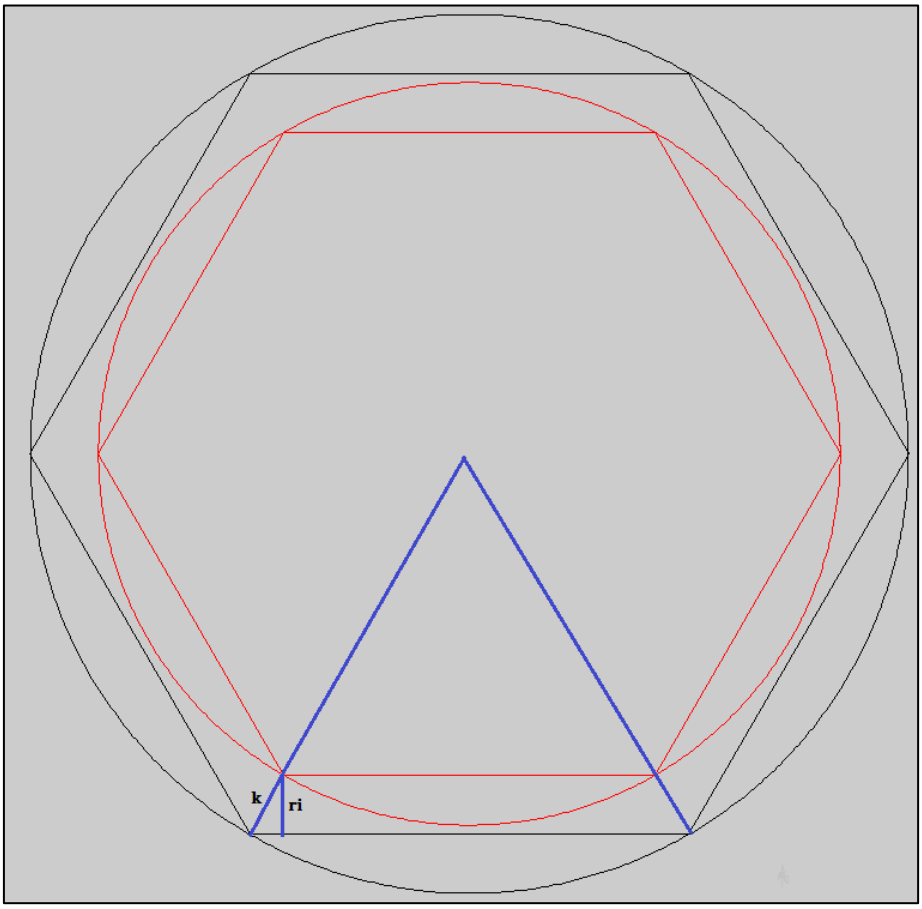

Figura 39. Obtención solución para situar armadura a distancia correcta

Si sobre el radio se disminuye una distancia k, que se puede obtener, se obtendrán los puntos en los que irán los vértices del polígono inscrito que, a su vez, darán como resultado que los dos polígonos estén separados una distancia ri (recubrimiento mecánico). La expresión resultante para k, es la siguiente:

$$k = \frac{ri}{\cos\left(\frac{\pi}{2 * nº\ lados}\right)}$$

Entonces, obteniendo esa relación, basta con introducirla en el script y se obtendrá el polígono inscrito, sobre el que irán los armados. Definido el polígono sobre el que irán situados los armados, falta definir las posiciones en las que se situarán.

Se ha considerado que habrá dos armados por lado para polígonos hasta un número de lados igual a veinte, mientras que para polígonos con un mayor número de lados, se considera la armadura correspondiente a un polígono de veinte lados, sea cual sea el número de lados.

Para situar los armados, sobre el polígono inscrito obtenido anteriormente se va a situar un polígono del doble de lados y sobre sus vértices se colocarán los respectivos armados. Esto, se puede ver en la figura 40, mostrada a continuación:

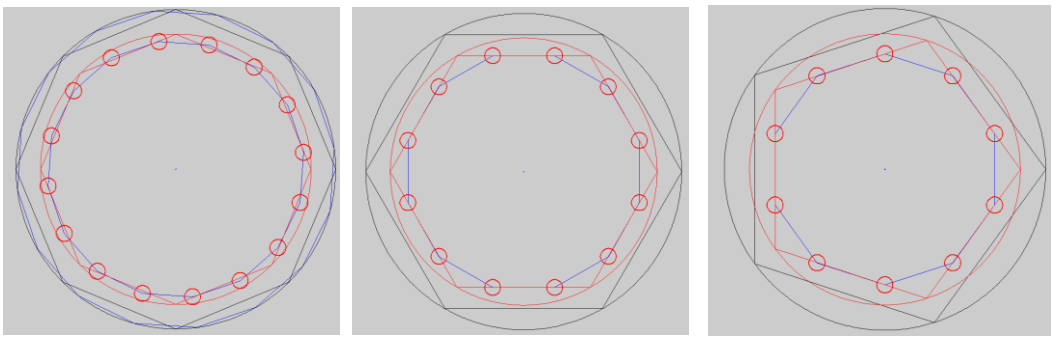

Figura 40. Solución final situación armados

Habiendo visto esto, ya se tienen definidas las posiciones de las armaduras, sea cual sea la sección poligonal que se pida.

3. <u>Graficar sección con armadura.</u> Llegado a este punto en el que están definidos tanto la sección transversal, como las armaduras, así como las propiedades mecánicas de hormigón y acero, el siguiente paso consta simplemente de graficar la sección con la armadura para ver si hemos colocado de manera correcta y en su sitio, tanto la sección como la armadura.
   La manera en sale por pantalla la sección junto con los armados, es la siguiente:

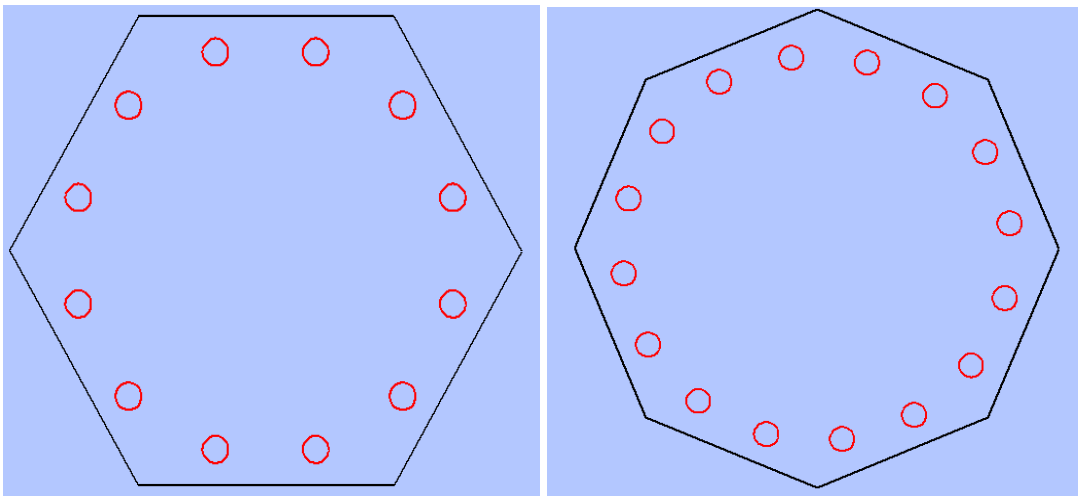

Figura 41. Sección hexagonal (izquierda) y sección octogonal (derecha)

A partir de aquí, el resto de pasos hasta los cálculos y obtención de resultados se desarrollan de la misma manera.

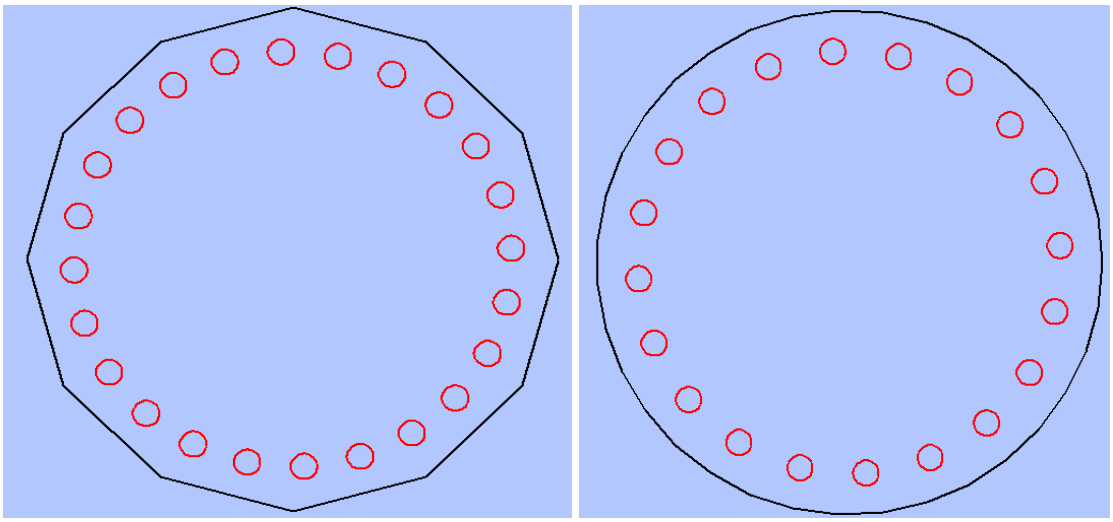

Figura 42. Sección de doce lados (izquierda) y sección "circular" (derecha)

4. Discretización de la sección por medio de la discretización de Delaunay. Habiendo representado gráficamente la sección, el siguiente paso que presenta algún cambio respecto al caso de la sección rectangular, sucede para la discretización de la sección.
   La diferencia que va a presentar este apartado es la división de la sección. Anteriormente, para el caso rectangular, se dividía la sección por medio de líneas horizontales y verticales separadas una cierta distancia. En este caso, no se puede realizar así la división de la sección. El procedimiento que se lleva a cabo es crear una variable para dividir la sección de n lados en n porciones, igualmente distribuidas. A partir de ahí, se realizan un número determinado de bucles (en nuestro caso se ha decidido que cuatro bucles son suficientes) para ir dividiendo la sección en triángulos, según las coordenadas de los vectores x e y a las que se les va añadiendo (en forma de concatenación, no de suma) un valor siguiente:

$$\frac{r}{i} * \cos(t)$$

   Se seguirá utilizando el comando "*delaunayTriangulation*", para realizar la triangulación de la sección.
   Se presentan, a continuación, algunos casos de triangulaciones de secciones:

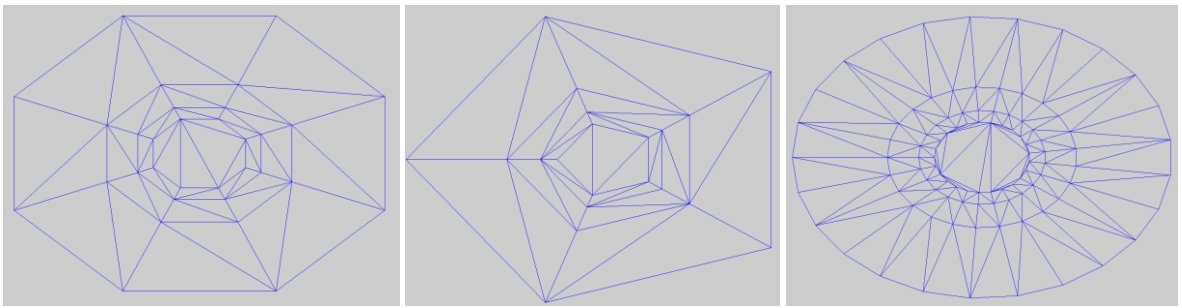

Figura 43. Ejemplos de secciones descretizadas de n=8,5 y 25 lados, respectivamente

4.3 Desarrollo de App de MATLAB mediante GUIDE

En esta sección, se va a explicar cómo se ha ido desarrollando la app mediante el editor de diseño GUIDE, contenido en el software MATLAB. Se irá desarrollando en varias fases, hasta finalmente obtener una interfaz que englobe la resolución de los dos problemas

vistos anteriormente, pudiendo cualquier usuario, nada más abrir la App, elegir qué tipo de problema desea resolver, si el caso particular de sección rectangular o el caso de cualquier sección poligonal de n lados.

4.3.1 Introducción a GUI

GUIDE permite crear aplicaciones con interfaces gráficas en MATLAB. Las GUI, también conocidas como interfaces gráficas de usuario o interfaces de usuario, permiten un control sencillo, por medio del ratón, de las aplicaciones de software, lo que elimina la necesidad de aprender lenguaje específico de MATLAB y escribir comandos para ejecutar cualquier tipo de aplicación (The MathWorks, Inc., 2019).

Las apps de MATLAB son programas autónomos con una interfaz gráfica de usuario "GUI" que automatizan tareas o realizan determinados cálculos. Una GUI debe ser consistente y fácilmente entendible por el usuario. Por lo general, una GUI incluye controles, que se encuentran en la paleta de componentes. Hay varios tipos de controles aunque se van a comentar, únicamente, los utilizados en nuestra app son (The MathWorks, MATLAB. The Language of Technical Computing, 2001):

- Push button. Generan una acción cuando son pulsados.

  Propiedades para establecer:

  - Cadena o "string": En este campo, se introduce lo que se quiere ver por pantalla.
  - Etiqueta o "tag": GUIDE utiliza la propiedad etiqueta para nombrar la subfunción de devolución de llamada en el archivo M de la aplicación. Es importante dar un nombre descriptivo a la etiqueta (por ejemplo, en nuestro caso, calcul_fig) antes de activar la GUI, debido a que, de lo contrario cuando se cree el código no será fácil situar cada botón, en caso de que haya varios.

- Edit text. Los controles de edición de texto son campos que permiten a los usuarios ingresar o modificar cadenas de texto. Se utilizan cuando se quieren meter datos de entrada.

- Static text. Son líneas de texto que no son posibles modificar. Sirven para indicar qué valor introducir, definir direcciones o para etiquetar otros controles.

- Popup Menus. Los popup menú o los menús emergentes, se abren para mostrar una lista de opciones cuando los usuarios presionan la flecha.
  La propiedad "string" contiene la lista de opciones que se muestran en el desplegable.
  Los menús emergentes son útiles cuando se desea proporcionar a los usuarios una serie de opciones diferentes, pero no desea ocupar la cantidad de espacio que requiere una serie de botones de opción.
  Se puede programar la devolución de llamada del menú desplegable para que funcione si se verifica el índice del elemento seleccionado o se puede obtener el valor introducido en el elemento seleccionado.
  Lo que hace la devolución de llamada del popup menú, es comprobar el índice del elemento seleccionado en el menú y se puede utilizar un switch para tomar medidas en función del valor.

- Habilitador o deshabilitador de controles. Configurando la propiedad “Enable” se puede controlar si un componente responde a un click del ratón. Un control puede presentar tres estados:
    - On. El control es operacional.
    - Off. El control está deshabilitado y su etiqueta aparece en color gris, sin poder pulsarla con el ratón. S i se encuentra en este estado, es necesario realizar una acción precedente para que se habilite.
    - Desactivado.

- Axes. Los ejes permiten mostrar gráficos (por ejemplo, gráficos o imágenes). Como todos los objetos gráficos, los ejes tienen propiedades que se pueden configurar para controlar muchos aspectos de su comportamiento y apariencia.

- Figure. Las figuras son las ventanas que contienen la GUI que se diseña con el editor de diseño “Layout Editor”. Es decir es el archivo que contiene todo lo definido en la interfaz.

Todos los controles se programan y definen en su “Property Inspector”, es ahí donde se definen sus propiedades del control, las acciones que realizarán dentro del programa, etc. Para seleccionar el “property inspector”, basta con pulsar el botón derecho del ratón sobre el componente seleccionado. Una vez dentro se pueden definir todas sus propiedades tanto de apariencia, como de llamada al archivo .m generado, etc.

Es importante hablar también de los “callbacks”, que son las funciones que realizan las acciones requeridas cuando un determinado componente se activa. Este código lo genera automáticamente GUIDE. Cuando se añade un componente al layout del GUIDE, se le asigna un nombre en la propiedad “Tag”, que es la usada para generar el nombre de la llamada o “callback”. La forma en que se presenta una función callback, creada en el guide, es la siguiente:

```
function Object_Callback(hObject, eventdata, handles)
```

Los argumentos que presenta dicha función significan:

- h → Identificador del objeto que se está ejecutando
- eventdata → Vacío, para un uso futuro
- handles → Una estructura que contiene los identificadores de todos los componentes en la GUI, cuyos nombres de campo están definidos por la propiedad Tag del componente.

Estos callbacks son susceptibles de añadirles propiedades. Todos los objetos gráficos tienen tres propiedades que permiten definir la rutina de llamada:

- ButtonDownFcn –MATLAB ejecuta esta llamada cuando el usuario pulsa el botón izquierdo del ratón y el cursor está sobre el objeto o cercano a él.
- CreateFcn – MATLAB ejecuta esta llamada cuando se crea el objeto
- DeleteFcn – MATLAB ejecuta esta llamada solo antes de eliminar el objeto.

Una de las grandes ventajas de GUI es que permite crear tus propias aplicaciones personalizadas. Con GUIDE (entorno de desarrollo de GUI), es posible diseñar gráficamente la interfaz de usuario. GUIDE genera entonces de manera automática el código de MATLAB para construir la interfaz, el cual se puede modificar para programar el comportamiento de la app.

Cuando se crea una app con guide, se crean dos ficheros. Uno de ellos es un archivo .m que incluye el código y, el otro fichero es un .fig que es la interfaz a partir de la cual introduciendo los datos de entrada te va a devolver por pantalla los resultados sin necesidad ninguna de manipular el código.

Finalmente, si se quiere crear un ejecutable, hay que dirigirse a la pantalla principal de MATLAB, a la pestaña APPS y Application Compiler, ver figura

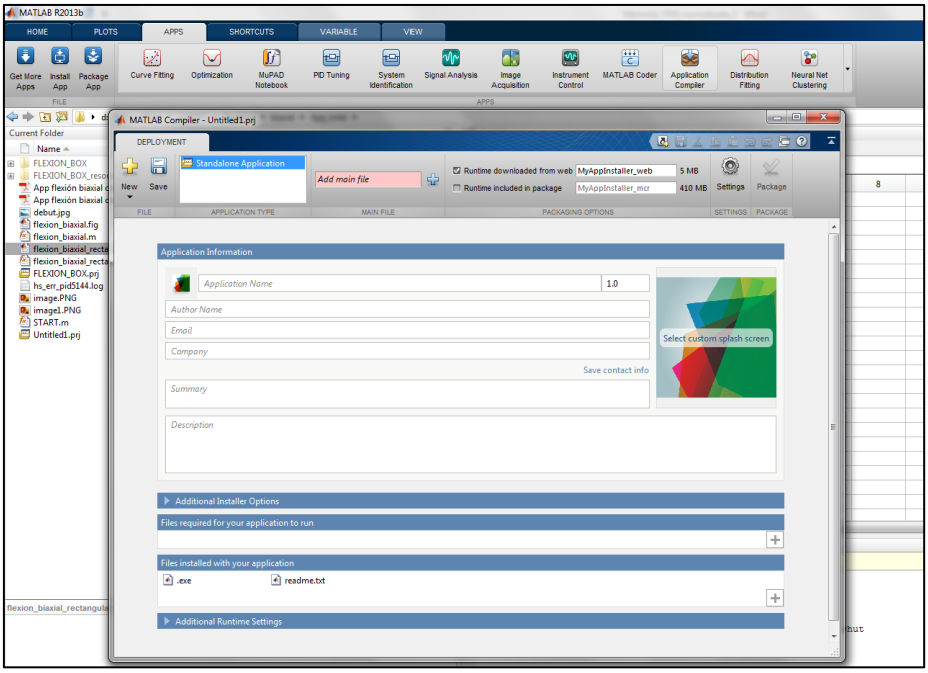

Figura 44. Generación de aplicación ejecutable mediante "Application compiler"

En la pantalla que se observa en la figura 44, hay que introducir el nombre que se quiera dar a la aplicación, los ficheros necesarios (.m y .fig) más alguna imagen, en el caso de que la app incluya alguna en el interior. Es necesario pinchar la opción "Runtime included in package" para generar el mcr, con lo que se consigue que se pueda ejecutar la app en cualquier pc aunque, no se disponga de MATLAB en el propio pc. Bastaría con instalar el archivo .mcr generado en cualquier ordenador para hacerla funcionar. Esta opción se considera muy importante, ya que con un simple instalador, con un tamaño de alrededor de 500 MB, cualquier aplicación generada con MATLAB se puede ejecutar en un pc cualquiera, sin necesidad de tener que pagar licencias.

Todo lo visto en esta introducción, se verá con más detalle en el siguiente punto, donde se explicará cómo ha sido el desarrollo de la App.

4.3.2 Creación de la App "FLEXION BOX"

Los pasos seguidos en el desarrollo de la App, objeto del presente Trabajo Fin de Máster, son los siguientes:

Paso 1. Haber finalizado los scripts que se van a introducir en la app, es decir, el script correspondiente a la resolución del problema de comprobación para la sección rectangular, así como el script para la sección poligonal de n lados.

Paso 2. Realizar un boceto en papel de la interfaz, tanto para la app de la sección rectangular como para la App de sección poligonal de n lados, para ver la distribución final que va a presentar la App. Este paso, aunque parezca algo trivial, va a ser de gran

importancia debido a que va a dar una rápida imagen de cómo se va a concebir la aplicación y, va a significar un ahorro de tiempo ya que, si directamente se dispone a realizar la interfaz de la app, sin antes haber realizado algún tipo de boceto, seguramente haya que realizar numerosos cambios hasta alcanzar la forma final, lo que conllevará más tiempo del necesario.

Paso 3. Una vez clara la distribución que va a presentar la interfaz de la App, se dispondrá a trabajar con "GUIDE". Para ello, lo primero que hay que hacer es introducir el comando "*guide*" en el command window de MATLAB y aparecerá la siguiente pantalla en la que se deberá seleccionar "*Blank Gui (Default)*", ver figura 45:

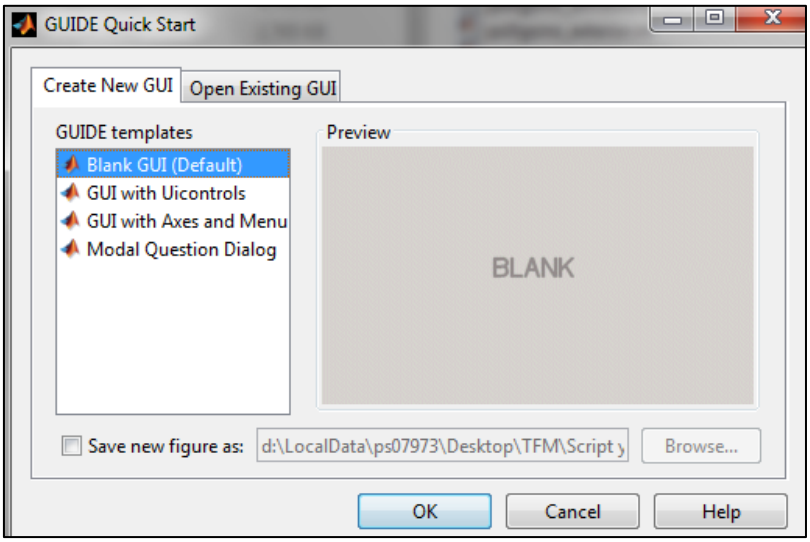

Figura 45. Pantalla inicial de GUIDE

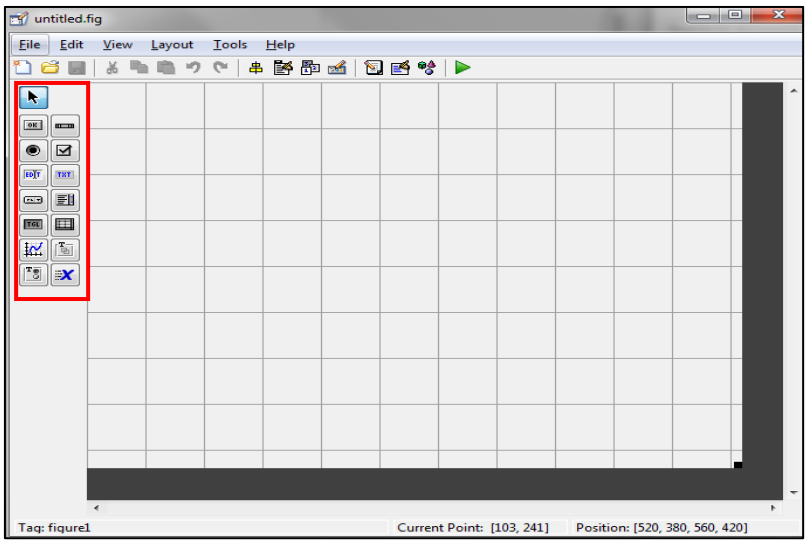

Figura 46. Pantalla inicial de un guide vacío

La pantalla que se observa en la imagen 46, es sobra la cual se van a insertar todos los botones, imágenes, desplegables, etc. Es lo que anteriormente se ha denominado figure. Es el lienzo sobre el que vamos a definir nuestra aplicación. En el recuadro que se observa en rojo, la paleta de componentes, están incluidas todos los controles (push button, edit text,...) que se pueden elegir y definir para la interfaz.

Sobre la figure, si se pulsa el botón derecho del ratón y se selecciona la acción “property inspector” se podrá acceder a todas sus propiedades para hacer cualquier cambio que se quiera, ver figura 47. En nuestro caso, se eligió un determinado color y se aumentaron las medidas de la interfaz.

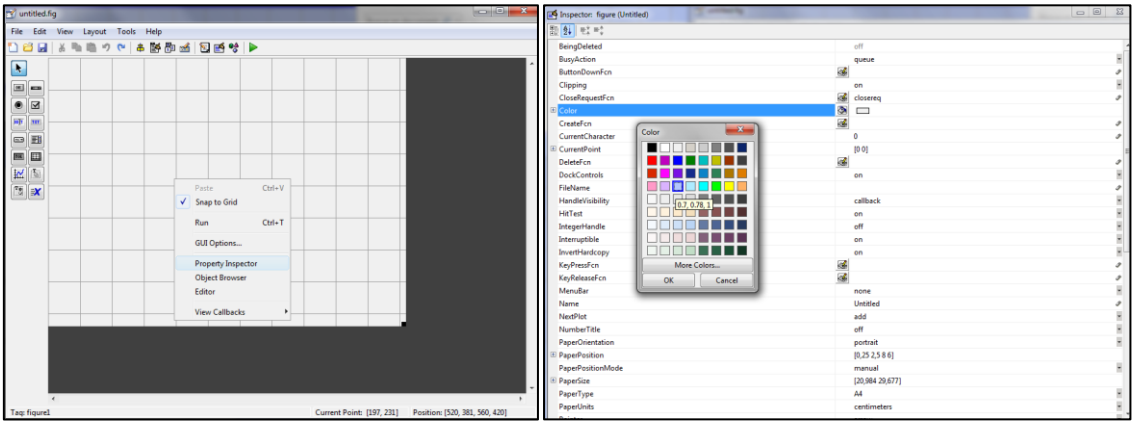

Figura 47. Selección del color de fondo del figure

Teniendo creada la figure y definidas sus propiedades, se da por finalizado el paso inicial de comenzar con la guide.

<u>Paso 4.</u> En este paso, se va a realizar la inclusión de todos los controles que se consideran necesarios para poder desarrollar el problema de comprobación y obtener los resultados adecuados. Este paso, va a haber que dividirlo en dos partes, para ver la interfaz de las dos app que se van a hacer, aunque finalmente las dos se introduzcan en una sola App final.

- <u>App sección rectangular</u>

Se va a presentar una imagen de la forma que presenta la interfaz final y se va a explicar brevemente el funcionamiento de dicha app. Ver figura 48.

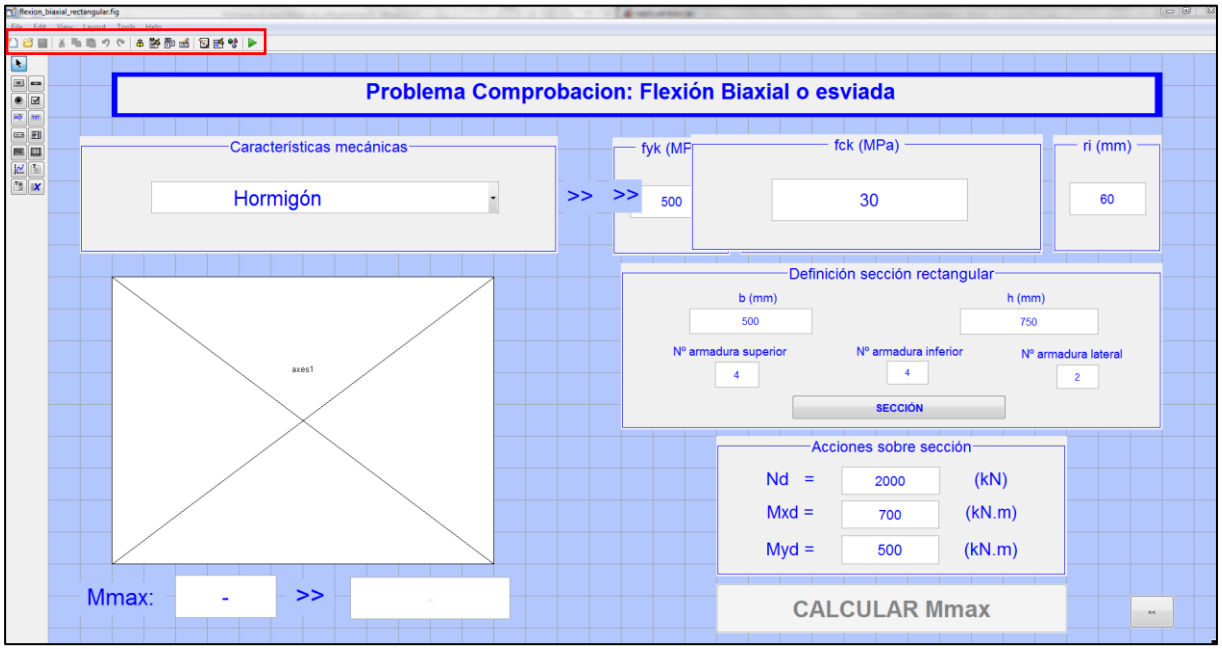

Figura 48. Imagen que presenta la forma final de la interfaz de la app para la resolución del problema de comprobación de la sección rectangular

Habiendo finalizado la inserción de elementos en el figure, es necesario pulsar el botón play, que se observa en la figura 48, recuadrado en rojo. Tras realizar esa pulsación de play, se ejecuta el guide creando el archivo .m con las funciones callback definidas en cada botón. La forma en que se presenta, una vez realizado lo dicho, es la siguiente:

Figura 49. App final para la resolución del problema rectangular

La imagen que aparece recuadrada en rojo, se ha introducido para que la interfaz sea más atractiva a nivel visual, hasta que se pulse el botón de sección, ya que sino aparecería una gran cantidad de la pantalla vacía.

El funcionamiento de la app, es el siguiente:

1. Utilizar el popup menú, denominado en su etiqueta mediante "mode", e identificado en el guide mediante "Características mecánicas". Este popup menú, cuenta con dos opciones: hormigón y armadura. La forma en que se ha programado este popup menú es con un switch, de manera que si selecciona la opción hormigón se habilita una ventana, mientras que si selecciona la opción armadura se habilitan otra serie de ventanas. A continuación, se presenta el código utilizado para realizar esto.

```
switch n
    case 1
        set(handles.panel_arm1,'visible','off');
        set(handles.panel_arm2,'visible','off');
        set(handles.panel_arm3,'visible','off');
        set(handles.panel_horm,'visible','on');
        set(handles.flecha_ar,'visible','off');
        set(handles.flecha_ho,'visible','on');

    case 2
        set(handles.panel_arm1,'visible','on');
        set(handles.panel_arm2,'visible','on');
        set(handles.panel_arm3,'visible','on');
        set(handles.panel_horm,'visible','off');
        set(handles.flecha_ar,'visible','on');
        set(handles.flecha_ho,'visible','off');
```

```
end
```

En las siguientes imágenes, se presenta como funciona este popup menú.

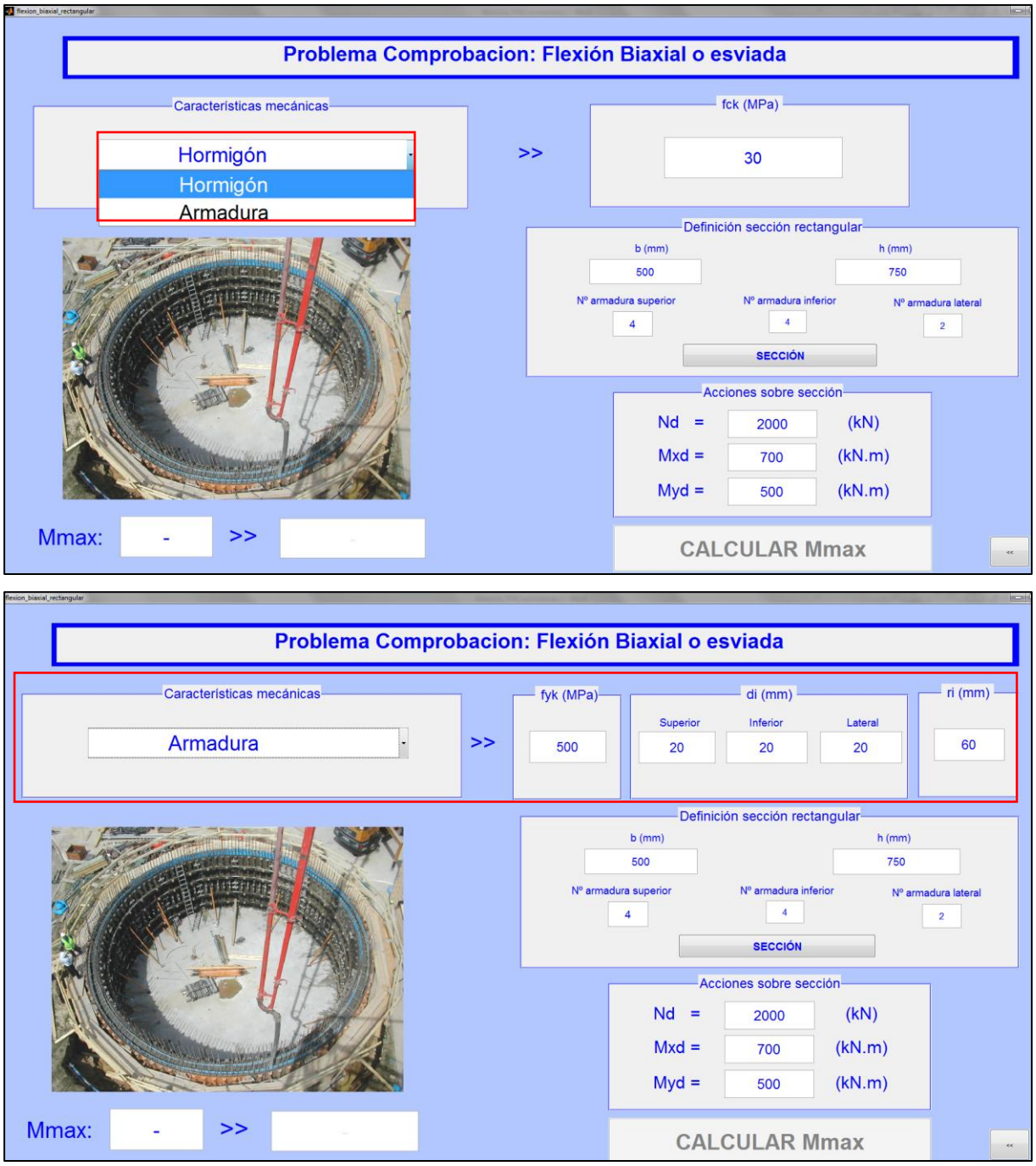

Figura 50. Opciones del popup menú recuadradas en rojo

En función de la opción seleccionada, hay que introducir los valores siguientes:

- Hormigón:
  - Resistencia característica a compresión del hormigón en MPa, fck.
- Acero:
  - Límite elástico característico del acero en MPa, fyk.
  - Diámetros de las armaduras (di) de cada una de las caras (superior, inferior y lateral) en mm.
  - Recubrimiento mecánico (ri).

2. El siguiente panel que se debe rellenar es el denominado en la interfaz como “Definición sección rectangular” y que cuenta con un total de cinco paneles edit text, que conforman las dimensiones geométricas de la sección y el número de armados de cada cara. Una vez introducidos los valores deseados, es necesario pulsar el push button “SECCIÓN”. Una vez pulsado dicho botón, aparecerá por pantalla el siguiente mensaje:

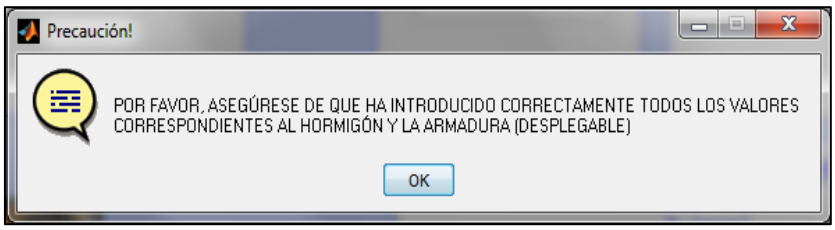

Figura 51. Mensaje de aviso para asegurar que se han introducido todos los campos

Al pulsar OK, se entiende que todo se ha rellenado de manera correcta y aparecerá por pantalla la sección rectangular junto con la armadura especificada. Ver figura.

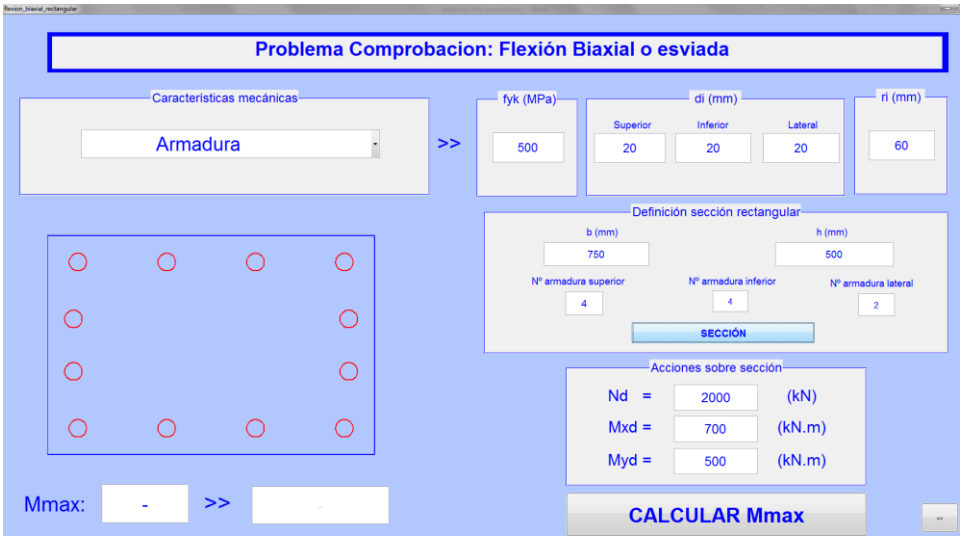

Figure 52. Salida por pantalla de la sección rectangular de hormigón armado

Otra de las cosas de las que hay que percatarse es que, hasta que no ha salido por pantalla la sección, el push button “CALCULAR Mmax” no se ha habilitado. Esto se realizó deshabilitando el botón “CALCULAR Mmax” en su “Property inspector”.

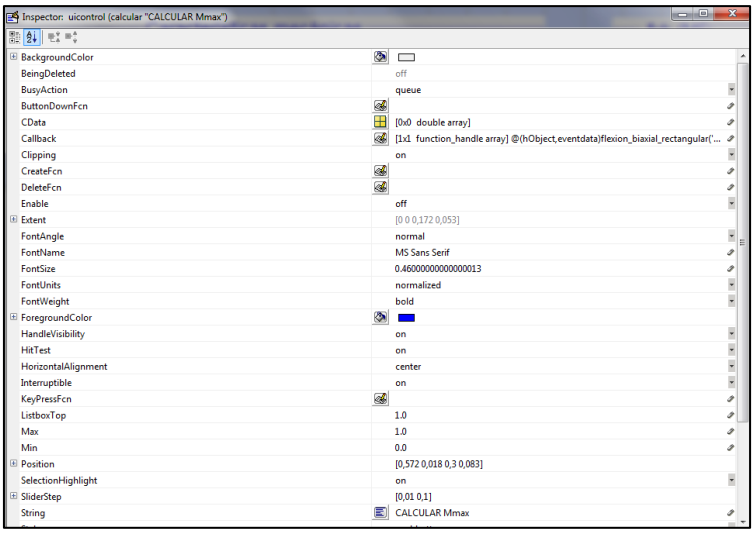

Figura 53. Property inspector del botón "CALCULAR Mmax"

3. A continuación, hay que introducir en el panel "Acciones sobre sección", los valores para las acciones aplicadas sobre la sección. Son tres valores correspondientes a la terna de valores (N_d, M_{xd}, M_{yd}). Una vez introducidos esos valores, es suficiente con pulsar el botón "CALCULAR Mmax" para que realice los cálculos correspondientes, saliendo por pantalla el valor del M_{xmax} y dando la respuesta a si resiste o no resiste la sección, como se puede ver en la figura siguiente.

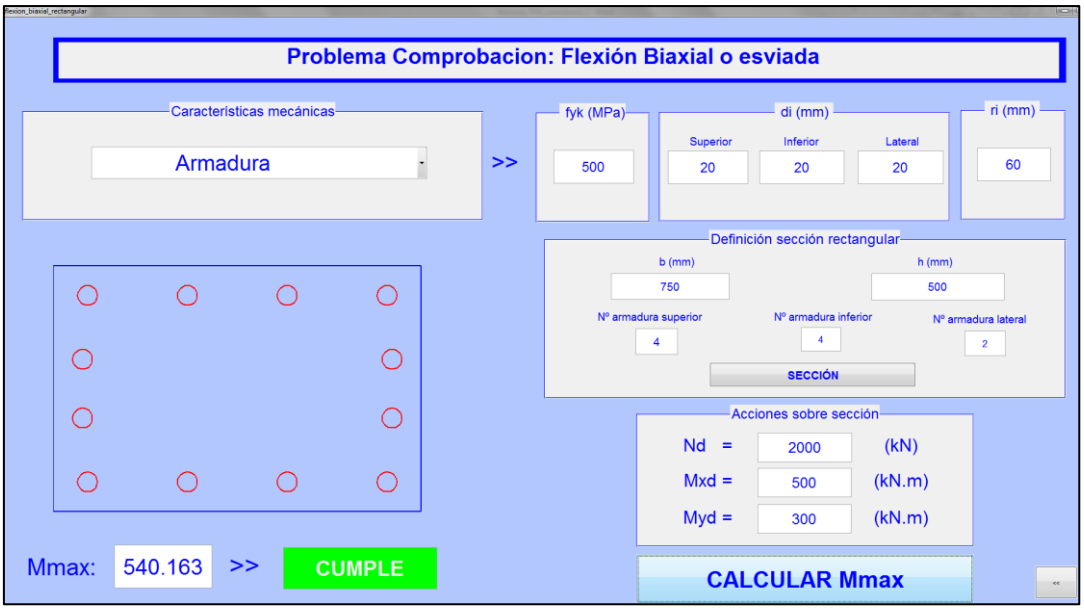

Figura 54. Obtención del resultado final tras introducir los valores de las acciones aplicadas sobre la sección

4. Por último, hay un botón en la parte inferior derecha de la pantalla (ver figura 53), que se ha hecho para volver a la pantalla inicial de la app "FLEXIÓN BOX", que es la interfaz inicial, en la que se podrá elegir entre qué tipo de problema resolver. Dicha pantalla inicial se verá una vez vista la interfaz para el caso de la resolución del problema de comprobación para el caso de una sección poligonal de un número n de lados.
   La programación de ese botón, que es algo diferente al resto, se hace de la manera siguiente, como se muestra en el script:

```
function Volver_inicial1_Callback(hObject, eventdata, handles)
close all
    clear
    START
```

Definida la llamada del botón, lo que realiza una vez pulsado el botón es cerrar todo y volver a la pantalla "START" que es la pantalla inicial dicha anteriormente.

Hasta aquí llega la explicación de cómo se desarrolla la aplicación para la sección rectangular.

- App sección poligonal

Se va a presentar una imagen de la forma que presenta la interfaz final y se va a explicar brevemente el funcionamiento de dicha app. Funciona de manera muy similar a la anterior explicada, salvo alguna pequeña modificación que se comentará. La imagen final que presenta el archivo .fig es la siguiente:

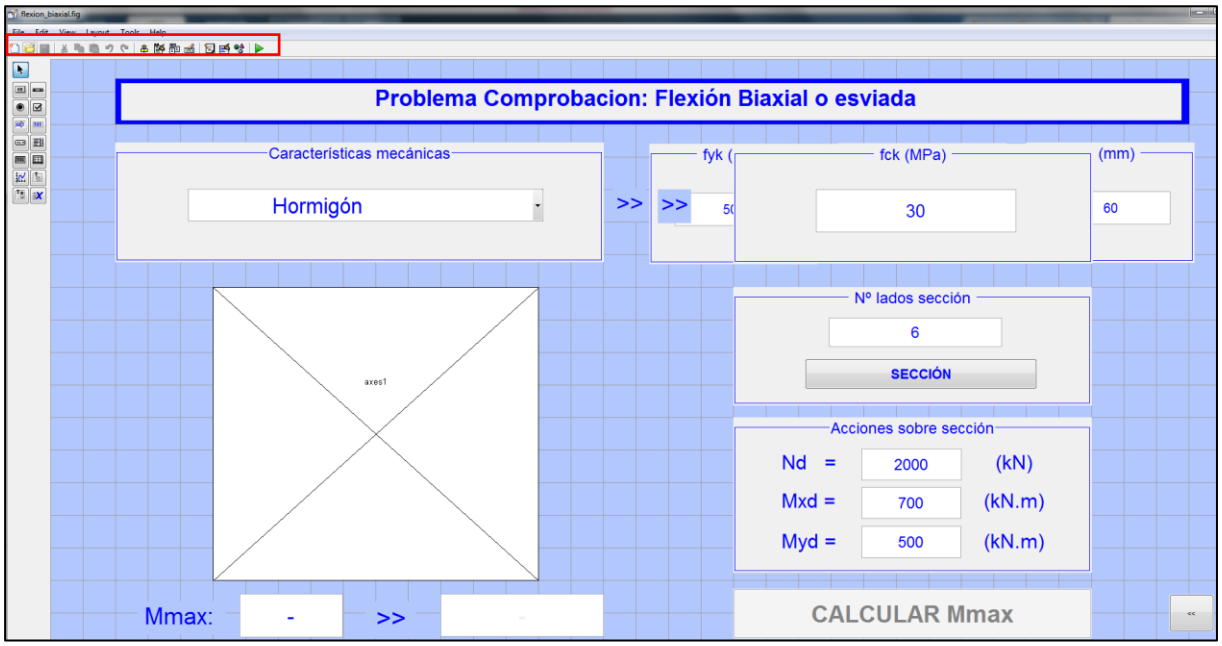

Figura 55. Imagen que presenta la forma final de la interfaz de la app para la resolución del problema de comprobación de la sección poligonal

Habiendo finalizado la inserción de elementos en el figure, es necesario pulsar el botón play, que se observa en la figura 55, recuadrado en rojo. Tras realizar esa pulsación de play, se ejecuta el guide creando el archivo .m con las funciones callback definidas en cada botón. La forma en que se presenta, una vez realizado lo dicho, es la siguiente:

Figura 56. App final para la resolución del problema de sección poligonal de n lados

La imagen que aparece recuadrada en rojo, se ha introducido para que la interfaz sea más atractiva a nivel visual, hasta que se pulse el botón de sección, ya que sino aparecería una gran cantidad de la pantalla vacía.

El funcionamiento de la app, es el siguiente:

1. Utilizar el popup menú, denominado en su etiqueta mediante "mode", e identificado en el guide mediante "Características mecánicas". Este popup menú, cuenta con dos opciones: hormigón y armadura. La forma en que se ha programado este popup menú es con un switch, de manera que si selecciona la opción hormigón se habilita una ventana, mientras que si selecciona la opción armadura se habilitan otra serie de ventanas. A continuación, se presenta el código utilizado para realizar esto.

```
switch n
    case 1
        set(handles.panel_arm1,'visible','off');
        set(handles.panel_arm2,'visible','off');
        set(handles.panel_arm3,'visible','off');
        set(handles.panel_horm,'visible','on');
        set(handles.flecha_ar,'visible','off');
        set(handles.flecha_ho,'visible','on');

    case 2
        set(handles.panel_arm1,'visible','on');
        set(handles.panel_arm2,'visible','on');
        set(handles.panel_arm3,'visible','on');
        set(handles.panel_horm,'visible','off');
        set(handles.flecha_ar,'visible','on');
        set(handles.flecha_ho,'visible','off');
end
```

En las siguientes imágenes, se presenta como funciona este popup menú.

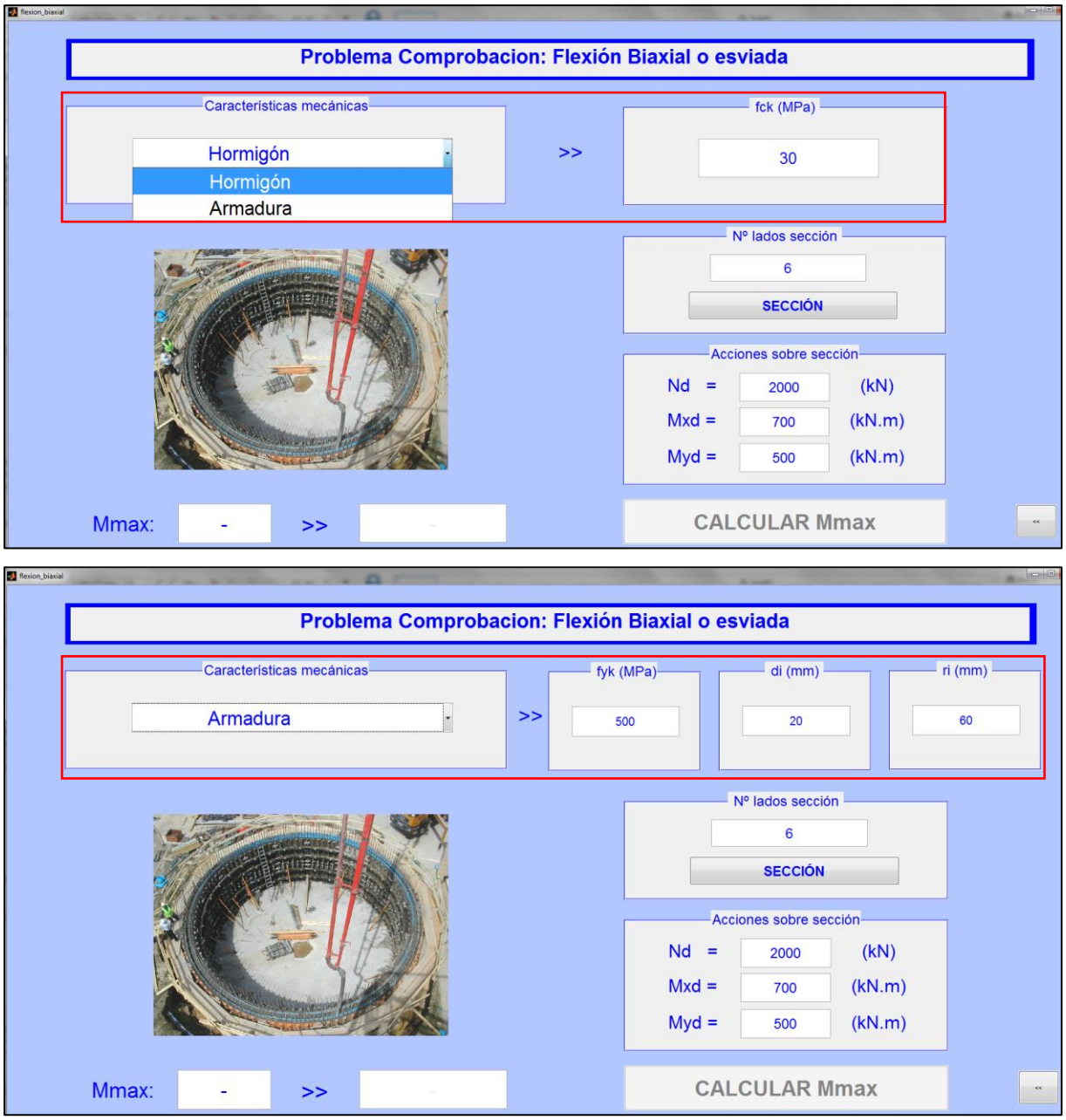

Figura 57. Opciones del popup menú recuadradas en rojo

En función de la opción seleccionada, hay que introducir los valores siguientes:

- Hormigón:
  - Resistencia característica a compresión del hormigón en MPa, fck.
- Acero:
  - Límite elástico característico del acero en MPa, fyk.
  - Diámetros de las armaduras (di) en mm. En este caso, como ya se explicó anteriormente, para polígonos con un número de lados menor que veinte, se han utilizado dos armados por lado, mientras que para polígonos con un número de lados mayor que veinte se ha utilizado un armado por lado.
  - Recubrimiento mecánico (ri).

2. El siguiente panel que se debe rellenar es el denominado en la interfaz como "Nº lados sección", que cuenta con un paneles edit text, en el que hay que introducir

el número de lados que se quiere que tenga la sección. Una vez introducidos los valores deseados, es necesario pulsar el push button “SECCIÓN”. Una vez pulsado dicho botón, aparecerá por pantalla el siguiente mensaje:

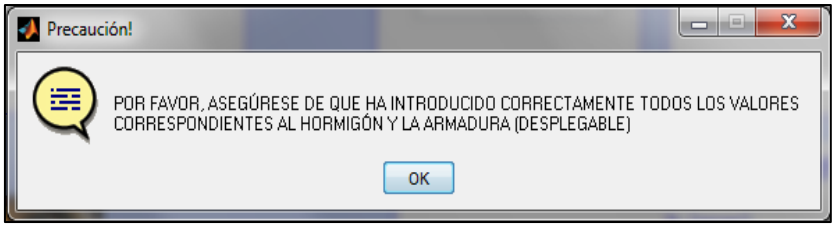

Figura 58. Mensaje de aviso para asegurar que se han introducido todos los campos

Al pulsar OK, se entiende que todo se ha rellenado de manera correcta y aparecerá por pantalla la sección poligonal de n lados junto con la armadura especificada. Ver figura 59, para ver el ejemplo de seis y de veinte lados.

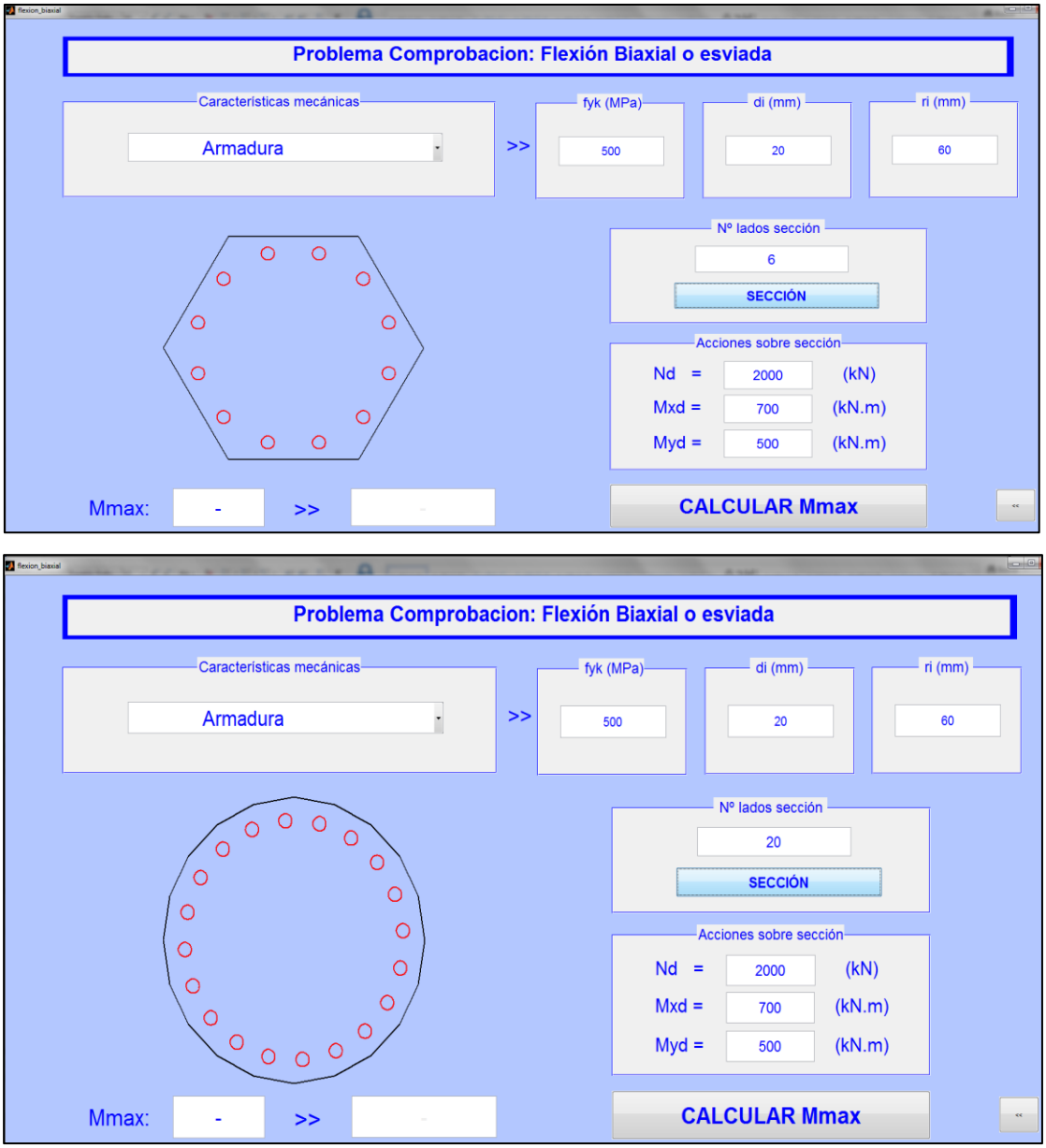

Figure 59. Salida por pantalla de la sección poligonal de hormigón armado

Otra de las cosas de las que hay que percatarse es que, hasta que no ha salido por pantalla la sección, el push button "CALCULAR Mmax" no se ha habilitado. Esto se realizó deshabilitando el botón "CALCULAR Mmax" en su "Property inspector".

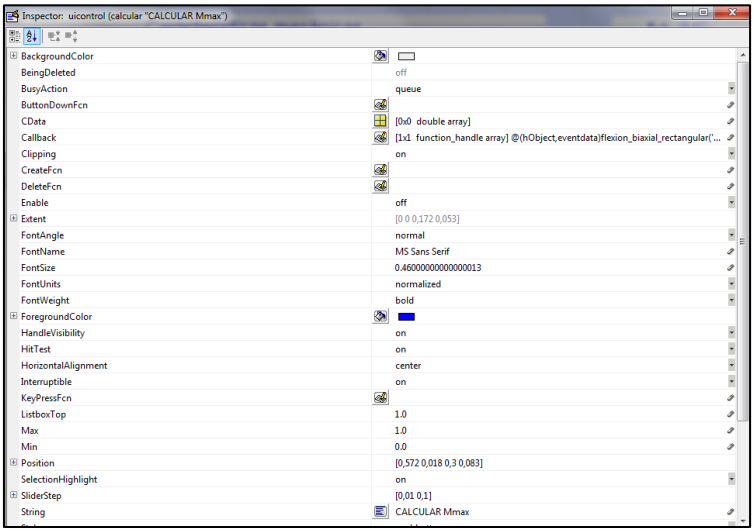

Figura 60. Property inspector del botón "CALCULAR Mmax"

3. A continuación, hay que introducir en el panel "Acciones sobre sección", los valores para las acciones aplicadas sobre la sección. Son tres valores correspondientes a la terna de valores (N_d, M_{xd}, M_{yd}). Una vez introducidos esos valores, es suficiente con pulsar el botón "CALCULAR Mmax" para que realice los cálculos correspondientes, saliendo por pantalla el valor del M_{xmax} y dando la respuesta a si resiste o no resiste la sección, como se puede ver en la figura siguiente.

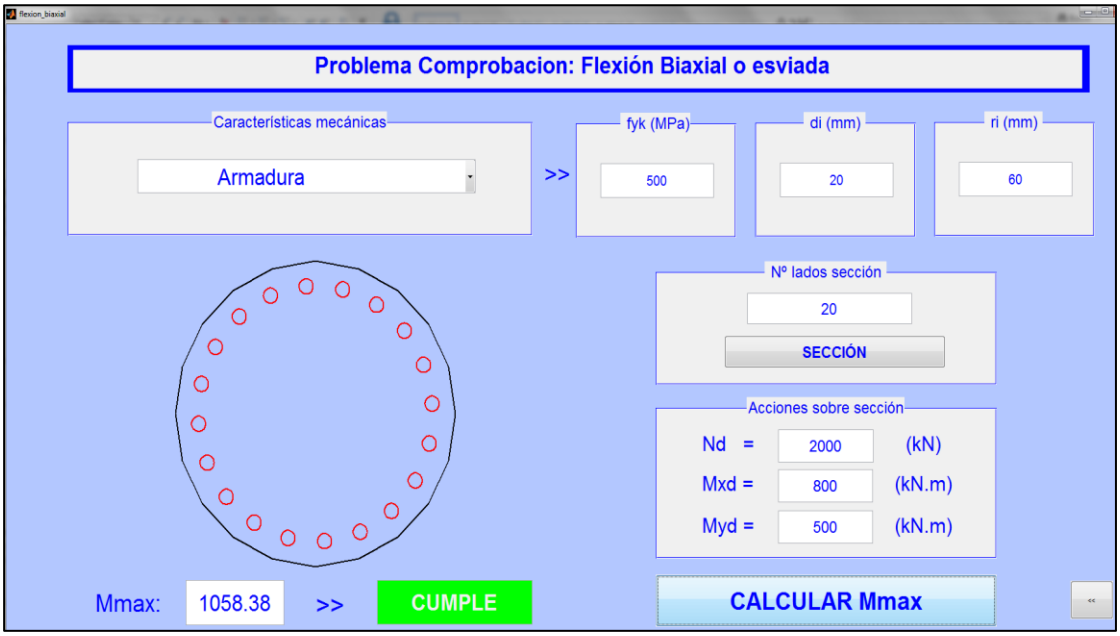

Figura 61. Obtención del resultado final tras introducir los valores de las acciones aplicadas sobre la sección

4. Por último, hay un botón en la parte inferior derecha de la pantalla (ver figura 60), que se ha hecho para volver a la pantalla inicial de la app "FLEXIÓN BOX", que es la interfaz inicial, en la que se podrá elegir entre qué tipo de problema resolver. La

programación de ese botón, que es algo diferente al resto, se hace de la manera siguiente, como se muestra en el script:

```
function Volver_inicial1_Callback(hObject, eventdata, handles)
close all
    clear
    START
```

Definida la llamada del botón (Volver_inicial1), lo que realiza una vez pulsado el botón es cerrar todo y volver a la pantalla "START" que es la pantalla inicial dicha anteriormente.

Hasta aquí llega la explicación de cómo se desarrolla la aplicación para la sección poligonal de un número n de lados.

Paso 5. Integración de los scripts creados inicialmente dentro de las funciones que se han ido creando para cada elemento de control que presenta la interfaz.

Como se ha dicho, al arrastrar un determinado control a la interfaz y posteriormente, pulsar el botón de play, se genera el código propio de cada control, ya sea un push button, un popup menú, etc. Pues bien, dentro de cada código, si se requiere, habrá que introducir el script necesario para que se lleve a cabo el programa de manera satisfactoria.

Se va a mostrar a continuación, la forma en que se programa cada elemento, explicando si necesita o no la inclusión del código del script realizado. Esto se hará en el orden que hay que ir rellenado y desarrollando la app:

1. App rectangular

El orden de desarrollo de la app, es el mostrado en la imagen, con los números del 1 al 8:

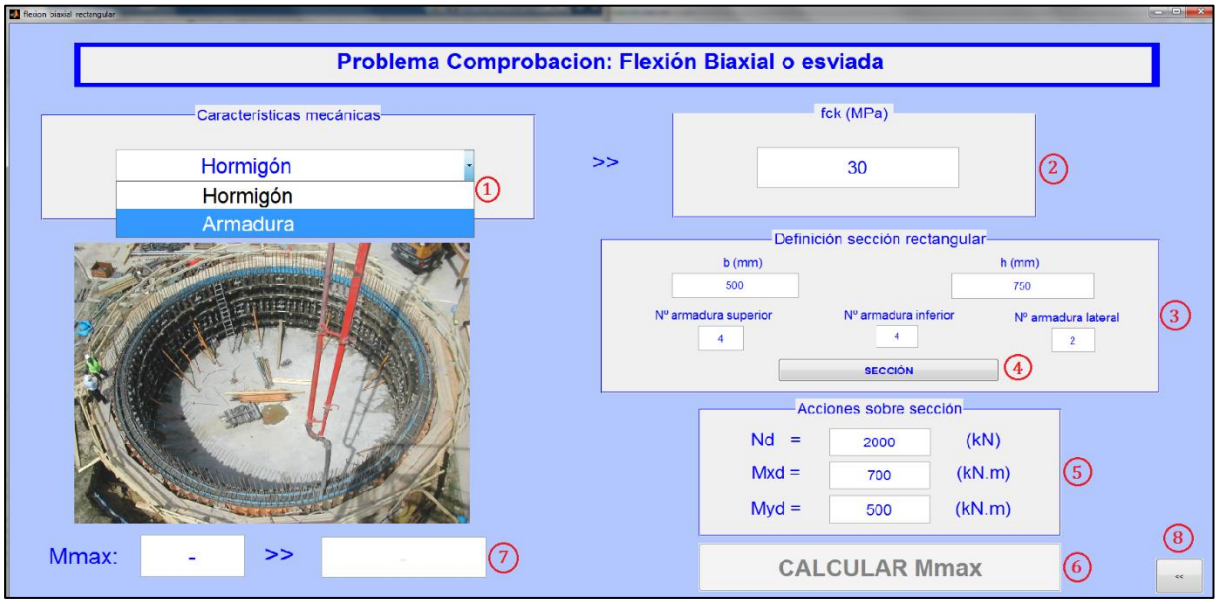

Figura 62. Orden de realización app

- Primer paso. Consiste en elegir la opción del popup menú: hormigón o acero. En función de la opción elegida, aparecerán unos edit text para rellenar. La manera de programar el popup menú es la siguiente:
    - Popup menú. Ya se ha enseñado anteriormente, que el popup menú se programa con un switch que en caso de escoger una opción u otra, habilita unos paneles y deshabilita otros. Su programación es la siguiente:

```
n = get(handles.mode,'value');
    switch n
        case 1
            set(handles.panel_arm1,'visible','off');
            set(handles.panel_arm2,'visible','off');
            set(handles.panel_arm3,'visible','off');
            set(handles.panel_horm,'visible','on');
            set(handles.flecha_ar,'visible','off');
            set(handles.flecha_ho,'visible','on');

        case 2
            set(handles.panel_arm1,'visible','on');
            set(handles.panel_arm2,'visible','on');
            set(handles.panel_arm3,'visible','on');
            set(handles.panel_horm,'visible','off');
            set(handles.flecha_ar,'visible','on');
            set(handles.flecha_ho,'visible','off');
    end
```

La opción uno, es la del hormigón, que habilita el panel del hormigón (en el script panel_hor) y también habilita unas flechas que se han puesto únicamente con carácter visual. Al elegir la opción uno, se deshabilitan el resto de paneles correspondientes al acero (panel_arm1, panel_arm2, panel_arm3 y flecha_ar).

Si se elige la opción dos, que es la armadura, sucede el caso contrario.

- Segundo paso. Está basado en rellenar Edit Text, para introducir los datos de entrada del problema, propios del hormigón y de la armadura.

  - Edit text. Estos paneles, habilitados en función de la opción que se haya elegido en el popup menú, sirven para introducir los datos de entrada. A parte del código automático que genera GUIDE no es necesario introducir nada referente al script realizado, a excepción, de la transformación de texto a valor de los valores introducidos en cada panel. Esta transformación, se puede realizar dentro de la programación del código generado por código o más tarde en la función principal donde se vayan a utilizar los datos. En nuestro caso, para evitar confusiones se ha realizado dicha transformación en la función principal donde se utilizan dichos datos. Esto se realiza con el comando “strdouble”, que permite la conversión de una cadena de caracteres a números. Se muestra un ejemplo de dicha programación:

```
fck = get(handles.fck,'string');
fck = str2double(fck);
```

    El panel en el que se introduce el valor de la resistencia característica a compresión del hormigón, se inserta como cadena de caracteres y hay que hacer la conversión a número, mediante esas dos líneas de código.

    El resto de paneles seguirán la misma forma.

- Tercer paso. Consiste en rellenar otros nuevos Edit Text, de manera que se introducen los datos geométricos y el número de armaduras de la sección. Su programación se realiza de igual manera a la vista en el paso dos.

- Cuarto paso. Una vez rellenados los Edit Text, hay que pulsar el push button "SECCIÓN". Al pulsar este botón, se desarrolla una parte del script creado por nosotros en el cual se genera la sección junto con sus armados. Dicha sección es la que posteriormente se va a utilizar para realizar los cálculos del M_{xmax}. Por lo tanto, dentro de este botón, irán las transformaciones de los caracteres introducidos en los Edit Text a números, así como, la parte del código en que se genera la sección de hormigón armado. Todo el código introducido, se podrá ver en el apartado anexos, debido a que se considera demasiado extenso para adjuntarlo aquí.

- Quinto paso. Rellenar los Edit Text con los valores correspondientes a las acciones que se aplican sobre la sección.

- Sexto paso. Es necesario pulsar otro push button denominado "CALCULAR Mmax", que se habilita en el momento que aparece por pantalla la sección de hormigón armado. Dentro del código generado por GUIDE, se introducen las conversiones de caracteres a números de los Edit Text que contienen los valores de las acciones que se aplican sobre la sección, así como, la parte del script que desarrolla el cálculo del problema de comprobación, que contiene la discretización, la integración de las secciones y las funciones "calcula" y esfuerzos" explicadas en el apartado 4.2.1 y 4.2.2.

- Sétimo paso. Aparece por pantalla el valor del momento máximo calculado alrededor del eje x, M_{xmax}. Son dos Edit Text, en uno aparece el valor del momento y en el otro dirá si cumple o no, comparando M_{xd} con M_{xmax}. En este caso, el valor obtenido para Mxmax, hay que pasarlo a cadena de caracteres para que aparezca en el Edit Text. El otro Edit Text aparecerá "CUMPLE" si la sección resiste o "NO CUMPLE" si la sección no resiste. Se han programado de la siguiente manera:

```
set(handles.result,'string',Mxmax);

if(Mxmax>Mxd)
    set(handles.cumple,'string','CUMPLE');
    set(handles.cumple,'BackgroundColor','green');
else
    set(handles.cumple,'string','NO CUMPLE');
    set(handles.cumple,'BackgroundColor','red');
end
```

2. *App poligonal de n lados*

El desarrollo es el mismo que se ha visto para el caso de la sección rectangular.

Paso 6. Generación de una sola aplicación que englobe las dos creadas y vistas anteriormente. Desde esta app general se podrá acceder a cualquiera de las dos app, ya sea la de sección rectangular o la de sección poligonal de n lados.

Se llegó a la conclusión de que, no era práctico contar con dos aplicaciones por separado y que podía ser posible desarrollar una sola aplicación desde la cual, se pudiera acceder a cualquiera de las dos.

La pantalla inicial desde la cual, se va a poder llamar a las otras dos aplicaciones, se denominó “START” y, es un archivo .m que se programó como un script normal.

Se definió una imagen de fondo, cuya única función es hacer de fondo de pantalla. Se introdujo, además, un static text, mediante programación. A mayores, se añadieron dos push button, para acceder a la aplicación rectangular o a la aplicación poligonal de n lados. Por último, se añadió un menú para acceder a dos ayudas (una para cada app), para que viendo esos dos pdf, sea posible ejecutar la app sin ningún tipo de problema. La imagen que se muestra, a continuación, muestra la presentación de la pantalla inicial, desde la que se podrá acceder tanto a las ayudas, como a las dos aplicaciones.

Figura 63. Pantalla principal de la app final “FLEXIÓN BOX”

La manera de llamar a las apps desde la pantalla principal, se muestra, a continuación:

```
function [] = rectangular(varargin)
    close all
    clear
    flexion_biaxial_rectangular

function [] = poligonal(varargin)
    close all
    clear
    flexion_biaxial
```

Anteriormente los push button han sido definidos junto con todas sus propiedades, teniendo como tag, “rectangular” el botón para acceder a la sección rectangular y, “poligonal” el botón para acceder a la aplicación de sección polygonal de n lados.

Por otro lado, el menú para acceder a las ayudas, ha sido programado de la siguiente manera:

```
S.m1=uimenu(S.figdiag,'label','Help');
S.m1a=uimenu(S.m1,'label','SECCION RECTANGULAR');
S.m1b=uimenu(S.m1,'label','SECCION POLIGONAL');
S.m1b1=uimenu(S.m1a,'label','Help','call',{@Help_rect});
S.m2a=uimenu(S.m1b,'label','Help','call',{@Help_poli});


function [] = Help_rect(varargin)
 winopen('App flexión biaxial con sección rectangular.pdf')

function [] = Help_poli(varargin)
 winopen('App flexión biaxial con sección poligonal.pdf')
```

Se definen en primer lugar el menú desplegable y, posteriormente, se hace la llamada a las funciones para que abran el pdf que corresponda.

4.3.3 FLEXIÓN BOX

Habiendo terminado de desarrollar la app completa. El último paso consiste en crear un archivo mcr, que es un ejecutable para que dicha app se pueda instalar en cualquier ordenador, disponga o no del programa MATLAB.

Para poder crear dicho ejecutable, hay que seguir los siguientes pasos:

1. Meter en una misma carpeta todos los archivos de las app (.m y .fig), las imágenes que van dentro de cada app y los ficheros pdf de las ayudas.
2. Acceder a MATLAB, ir a la pestaña APPS y seleccionar la casilla “Application Compiler”.
3. Seleccionar la pestaña “Runtime included in package” para que genere el archivo ejecutable .mcr.
4. Rellenar la información requerida como: nombre de aplicación, nombre del autor, versión, etc.
5. Introducir todos los ficheros incluidos en la carpeta, en el campo “Files installed with your application”.
6. Pulsar el botón “Package”, entonces tardará un rato.
7. Al finalizar, estará el ejecutable y la app creada.

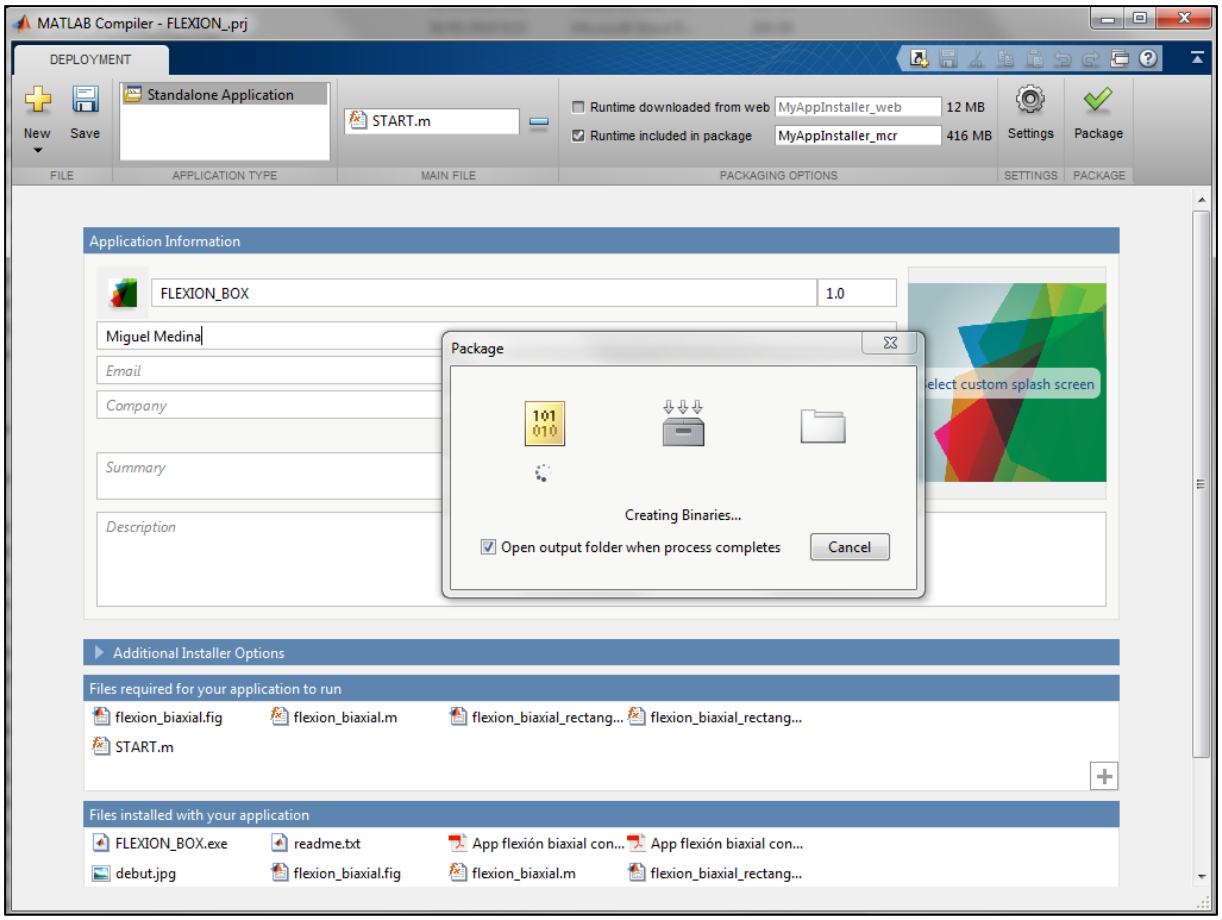

Figura 64. Creacción de.mcr mediante Application Compiler

Con esto quedaría realizada la App "FLEXIÓN BOX" objeto de este TFM.

Capítulo 5

CONCLUSIONES Y LÍNEAS FUTURAS

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 CONCLUSIONES

El proceso de trabajo ha sido, inicialmente, el estudio y la comprensión de los capítulos de hormigón estructural de los apuntes de la asignatura Estructuras, de segundo curso del máster de Ingeniería Industrial. Más en concreto, del capítulo siete, titulado "Análisis de la sección en flexión", dentro del cual, se estudian los diferentes tipos de flexión (simple, compuesta o esviada) a los que se puede ver sometida una sección. También, se estudiaron y resolvieron manualmente ejemplos, expuestos en este capítulo, tanto del problema de comprobación, como del problema de dimensionamiento.

A partir de ahí, se comenzó a trabajar en la resolución del problema de comprobación para el caso de flexión esviada para sección rectangular y, posteriormente, para una sección poligonal, que tuviera un número cualquiera de lados, incluso un número de lados tan grande que se asemejase a la sección circular. Todo esto, se fue traduciendo a leguaje de código implementándolo en MATLAB.

En cuanto al software MATLAB, era conocido pero a nivel usuario. Gracias a este trabajo, se han aumentado los conocimientos en cuanto a su manejo y programación. También, se ha descubierto una herramienta con un elevado potencial para un futuro como es GUIDE, a partir de la cual, se ha desarrollado la aplicación ejecutable. GUIDE es una herramienta poco conocida y divulgada y, que facilita mucho la programación dentro del entorno de MATLAB.

Como conclusiones cabe destacar desde un punto de vista positivo, el haber podido desarrollar con MATLAB una aplicación que resuelva el problema de dimensionamiento para una sección de hormigón armado de forma rectangular o poligonal, utilizando métodos poco conocidos, por mi parte, como son el campo de la discretización e integración de Gauss-Legendre (método de Hammer) de secciones. A su vez, decir que MATLAB nos ha permitido desarrollar el código para resolver el problema y crear, por medio de GUIDE, una interfaz desde cero. Por lo tanto, en cuanto a MATLAB en general y GUIDE en particular, el resultado del trabajo ha sido plenamente satisfactorio, ya que, poder contemplar y ver evolucionar una aplicación, desde los inicios, como son la creación del script, hasta las etapas finales como son la generación de los comandos, obtención de resultados,...merece mucho la pena.

Por otro lado, nos resulta amargo el no poder haber desarrollado esta aplicación para todo tipo de secciones normalizadas y utilizadas en estructuras, como son vigas doble T, corona circular (tubos), etc.

5.2 LÍNEAS FUTURAS

Habiéndonos adentrado en el mundo del hormigón armado en general y, del análisis a flexión de las secciones, en particular, resultaría muy interesante realizar lo mismo que se ha realizado en este TFM pero para el caso del problema de dimensionamiento de una sección de hormigón armado, es decir, resolver el problema de dimensionamiento para una sección de hormigón armado sometida a flexión biaxial.

Sería también, de gran interés pedagógico, desarrollar la resolución para el caso del problema de dimensionamiento en secciones de todo tipo (no sólo las vistas en este TFM) y poder traspasarlas a una nueva aplicación creada por GUIDE o cualquier otro sistema. Para que en determinados problemas prácticos, pudiera ser utilizado por cualquier alumno y así poder conseguir un sistema de comprobación rápido y visual.

Otro posible trabajo futuro relacionado con el tema tratado, sería realizar una comparación entre los diferentes métodos que hay para resolver el problema de comprobación en flexión esviada, como son los ábacos adimensionales, método de las excentricidades, etc…y poder analizar con cuál se obtienen unos resultados más cercanos a la realidad y cuánto error hay con cada método.

Capítulo 6
BIBLIOGRAFÍA

6 BIBLIOGRAFÍA

Castillo, F. G. (2007). MÉTODO DE ELEMENTOS FINITOS. Preprocesp y postproceso de resultados. Madrid.

Jiménez Montoya, P., García Meseguer, A., & Morán Cabré, F. (1987). Hormigón armado. GUSTAVO GILI.

Mansilla, D. L. (2018). ESTUDIO DEL COMPORTAMIENTO TERMO-MECÁNICO SECCIONAL DE COLUMNAS RECTANGULARES DE HORMIGÓN ARMADO SOMETIDAS A LA ACCIÓN DEL FUEGO. FLEXO-COMPRESIÓN RECTA Y ESVIADA. Valencia: Universidad Politécnica de Valencia.

Martín, E. H. (2007). HORMGÓN ARMADO Y PRETENSADO. Granada: Grupo de Investigación TEP-190 Ingeniería e Infraestructuras.

Morcillo, A. D. (2000). Método de mallado y algoritmos adaptativos en dos y tres dimensiones para la resolución de problemas electromagnéticos cerrados mediante el método de los elementos finitos. Valencia.

Páez, A. (1986). HORMIGÓN ARMADO. Barcelona: EDITORIAL REVERTÉ S.A.

Rathod, H., Nagaraja, K., & Venkatesudu, B. (s.f.). Symmetric Gauss Legendre quadrature formulas for composite numerical integration over a triangular surface. Bangalore, India.

The MathWorks, I. (2001). MATLAB. The Language of Technical Computing.

The MathWorks, I. (2019). https://es.mathworks.com/help/matlab/math/delaunay-triangulation.html. Obtenido de https://es.mathworks.com/help/matlab/math/delaunay-triangulation.html.

The MathWorks, Inc. (2019). https://es.mathworks.com/discovery/matlab-gui.html. Obtenido de https://es.mathworks.com/discovery/matlab-gui.html.

Torrano, S., & Martí, P. (s.f.). ANÁLISIS DE SECCIONES DE HORMIGÓN ARMADO DE FORMA CUALQUIERA SOMETIDAS A FLEXOCOMPRESIÓN. Cartagena, España: Escuela Técnica Superior de Ingeniería Industrial. Universidad Politécnica de Cartagena.

Zienkiewicz, O. (s.f.). El método de los elementos finitos. Reverté.

Capítulo 7

ANEXOS

7 ANEXOS. SCRIPT APP

Como anexo se introducirá el script de cada app (rectangular o poligonal), así como el de la app "START", que es la encargada de llamar a cualquiera de las dos.

- **App rectangular**

```
function varargout = flexion_biaxial_rectangular(varargin)
% FLEXION_BIAXIAL_RECTANGULAR MATLAB code for
flexion_biaxial_rectangular.fig
%      FLEXION_BIAXIAL_RECTANGULAR, by itself, creates a new
FLEXION_BIAXIAL_RECTANGULAR or raises the existing
%      singleton*.
%
%      H = FLEXION_BIAXIAL_RECTANGULAR returns the handle to a new
FLEXION_BIAXIAL_RECTANGULAR or the handle to
%      the existing singleton*.
%
%
FLEXION_BIAXIAL_RECTANGULAR('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in FLEXION_BIAXIAL_RECTANGULAR.M with the
given input arguments.
%
%      FLEXION_BIAXIAL_RECTANGULAR('Property','Value',...) creates a new
FLEXION_BIAXIAL_RECTANGULAR or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before flexion_biaxial_rectangular_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
flexion_biaxial_rectangular_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
flexion_biaxial_rectangular

% Last Modified by GUIDE v2.5 02-Jun-2019 17:53:08

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@flexion_biaxial_rectangular_OpeningFcn, ...
                   'gui_OutputFcn',
@flexion_biaxial_rectangular_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before flexion_biaxial_rectangular is made visible.
function flexion_biaxial_rectangular_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to flexion_biaxial_rectangular (see
VARARGIN)
global cont

cont=0;

a=imread('image1.png');
axes(handles.axes1);
imshow(a);

% Choose default command line output for flexion_biaxial_rectangular
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes flexion_biaxial_rectangular wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = flexion_biaxial_rectangular_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function fck_Callback(hObject, eventdata, handles)
% hObject    handle to fck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fck as text
%        str2double(get(hObject,'String')) returns contents of fck as a
double
```

```
% --- Executes during object creation, after setting all properties.
function fck_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fck (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



% --- Executes on selection change in mode.
function mode_Callback(hObject, eventdata, handles)
% hObject    handle to mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns mode contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from mode
n = get(handles.mode,'value');
    switch n
        case 1
            set(handles.panel_arm1,'visible','off');
            set(handles.panel_arm2,'visible','off');
            set(handles.panel_arm3,'visible','off');
            set(handles.panel_horm,'visible','on');
            set(handles.flecha_ar,'visible','off');
            set(handles.flecha_ho,'visible','on');

        case 2
            set(handles.panel_arm1,'visible','on');
            set(handles.panel_arm2,'visible','on');
            set(handles.panel_arm3,'visible','on');
            set(handles.panel_horm,'visible','off');
            set(handles.flecha_ar,'visible','on');
            set(handles.flecha_ho,'visible','off');
    end



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

% Hint: popupmenu 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 ri_Callback(hObject, eventdata, handles)
% hObject    handle to ri (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ri as text
%        str2double(get(hObject,'String')) returns contents of ri as a
double


% --- Executes during object creation, after setting all properties.
function ri_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ri (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 di1_Callback(hObject, eventdata, handles)
% hObject    handle to di1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of di1 as text
%        str2double(get(hObject,'String')) returns contents of di1 as a
double


% --- Executes during object creation, after setting all properties.
function di1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to di1 (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 di2_Callback(hObject, eventdata, handles)
% hObject    handle to di2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of di2 as text
%        str2double(get(hObject,'String')) returns contents of di2 as a
double


% --- Executes during object creation, after setting all properties.
function di2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to di2 (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 di3_Callback(hObject, eventdata, handles)
% hObject    handle to di3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of di3 as text
%        str2double(get(hObject,'String')) returns contents of di3 as a
double


% --- Executes during object creation, after setting all properties.
function di3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to di3 (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 fyk_Callback(hObject, eventdata, handles)
% hObject    handle to fyk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fyk as text
%        str2double(get(hObject,'String')) returns contents of fyk as a
double


% --- Executes during object creation, after setting all properties.
function fyk_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fyk (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 b_Callback(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
%        str2double(get(hObject,'String')) returns contents of n as a
double


% --- Executes during object creation, after setting all properties.
function b_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (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 h_Callback(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
%        str2double(get(hObject,'String')) returns contents of n as a
double


% --- Executes during object creation, after setting all properties.
function h_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (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 ns_Callback(hObject, eventdata, handles)
% hObject    handle to ns (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ns as text
%        str2double(get(hObject,'String')) returns contents of ns as a
double


% --- Executes during object creation, after setting all properties.
function ns_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ns (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 ni_Callback(hObject, eventdata, handles)
% hObject    handle to ni (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ni as text
%        str2double(get(hObject,'String')) returns contents of ni as a
double


% --- Executes during object creation, after setting all properties.
function ni_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ni (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 nl_Callback(hObject, eventdata, handles)
% hObject    handle to nl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nl as text
%        str2double(get(hObject,'String')) returns contents of nl as a
double


% --- Executes during object creation, after setting all properties.
function nl_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nl (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


% --- Executes on button press in calcul_fig.
function calcul_fig_Callback(hObject, eventdata, handles)
% hObject    handle to calcul_fig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global cont Ai Es fyd fck fcd fyk di1 di2 di3 ri b h ns ni nl acero


cont=cont+1;

if cont==1
    j=helpdlg('POR FAVOR, ASEGÚRESE DE QUE HA INTRODUCIDO CORRECTAMENTE
TODOS LOS VALORES CORRESPONDIENTES AL HORMIGÓN Y LA ARMADURA
(DESPLEGABLE)','Precaución!');
    waitfor(j);
end

b = get(handles.b,'string');
b = str2double(b);

h = get(handles.h,'string');
h = str2double(h);

ns = get(handles.ns,'string');
ns = str2double(ns);

ni = get(handles.ni,'string');
ni = str2double(ni);

nl = get(handles.nl,'string');
nl = str2double(nl);
```

```
    fck = get(handles.fck,'string');
    fck = str2double(fck);

    fcd=fck/1.5;

    fyk = get(handles.fyk,'string');
    di1  = get(handles.di1,'string');
    di2  = get(handles.di2,'string');
    di3  = get(handles.di3,'string');
    ri  = get(handles.ri,'string');
    fyk = str2double(fyk);
    di1 = str2double(di1);
    di2 = str2double(di2);
    di3 = str2double(di3);
    ri = str2double(ri);

Ai=[[ns di1 ri] %Poner una matriz con numero de armaduras en dirección
vertical y numero de armaduras en direccion horizontal
    [ni di2 ri]
    [nl di3 ri]
    [nl di3 ri]
    ];


    fyd=fyk/1.15;
    Es=2.0*10^5;


acero=armadura();
seccion(1,b/2+b/5,h/2+h/10);

axis off
hold off

set(handles.calcular,'enable','on');

function v=armadura()
    global Ai ns ni nl di1 di2 di3 ri b h
    na=sum(Ai(:,1));
    v=zeros(na,3);

    i=1; % armadura superior

    ns=Ai(i,1); % numero de armaduras alineadas
    di1=Ai(i,2); % diametro
    ri=Ai(i,3); % recubrimiento mecanico
    da=(b-(2*ri+ns*di1))/(ns-1);%distancia entre armados
    k=1;l=0;
    v(k,:)=[b/2-ri-di1/2 h/2-ri di1];%posición de los armados
    for j=2:ns
        v(k+j-1,:)=[v(k+j-2,1)-(di1+da) h/2-ri di1];
        l=l+1;
    end
    k=k+l;

    i=2; % armadura inferior
```

```
    ni=Ai(i,1); % numero de armaduras alineadas
    di2=Ai(i,2); % diametro
    ri=Ai(i,3); % recubrimiento mecanico
    da=(b-(2*ri+ni*di2))/(ni-1);
    k=k+1;l=0;
    v(k,:)=[b/2-ri-di2/2 -h/2+ri di2];
    for j=2:ni
        v(k+j-1,:)=[v(k+j-2,1)-(di2+da) -h/2+ri di2];
        l=l+1;
    end
    k=k+l;

    i=3; % armadura piel izquierda

    di1=Ai(1,2);
    r1=Ai(1,3);

    di2=Ai(2,2);
    r2=Ai(2,3);

    nl=Ai(i,1); % numero de armaduras alineadas
    di3=Ai(i,2); % diametro
    ri=Ai(i,3); % recubrimiento mecanico
    da=(h-(r1+di1+r2+di2+nl*di3))/(nl+1);
    k=k+1;l=0;
    v(k,:)=[-b/2+ri h/2-(r1+di1+da)-di3/2 di3];
    for j=2:nl
        v(k+j-1,:)=[-b/2+ri v(k+j-2,2)-(di3+da) di3];
        l=l+1;
    end
    k=k+l;

    i=4; % armadura piel derecha

    nl=Ai(i,1); % numero de armaduras alineadas
    di3=Ai(i,2); % diametro
    ri=Ai(i,3); % recubrimiento mecanico
    da=(h-(r1+di1+r2+di2+nl*di3))/(nl+1);
    k=k+1;l=0;
    v(k,:)=[b/2-ri h/2-(r1+di1+da)-di3/2 di3];
    for j=2:nl
        v(k+j-1,:)=[b/2-ri v(k+j-2,2)-(di3+da) di3];
        l=l+1;
    end

function seccion(esc,x0,y0)

    global b h di acero
    % dibuja seccion
    H0=[[b/2 h/2]
        [-b/2 h/2]
        [-b/2 -h/2]
        [b/2 -h/2]
        ];
    H0(:,1)=H0(:,1)+x0;
    H0(:,2)=H0(:,2)+y0;
    H0=esc*H0;

    %figure;
```

```
    hold off

    n0=size(acero);
    n0=n0(1);
    for i=1:n0
        a=acero(i,:);
        di=a(3);
        t=0:pi/30:2*pi;
        x=a(1)+di*cos(t);y=a(2)+di*sin(t);
        x=x+x0;y=y+y0;
        x=esc*x;y=esc*y;
%         hold on;
        plot(x,y,'r','LineWidth',1.5);
        hold on;

    end


    line(H0(1:2,1),H0(1:2,2),'LineWidth',2);
    line(H0(2:3,1),H0(2:3,2),'LineWidth',2);
    line(H0(3:4,1),H0(3:4,2),'LineWidth',2);
    x=zeros(1,2);y=zeros(1,2);
    x(1)=H0(1,1);x(2)=H0(4,1);
    y(1)=H0(1,2);y(2)=H0(4,2);
    line(x,y,'LineWidth',2);
%     axis off


    axis equal


function Nd_Callback(hObject, eventdata, handles)
% hObject    handle to Nd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nd as text
%        str2double(get(hObject,'String')) returns contents of Nd as a
double


% --- Executes during object creation, after setting all properties.
function Nd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nd (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 Mxd_Callback(hObject, eventdata, handles)
% hObject    handle to Mxd (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Mxd as text
%        str2double(get(hObject,'String')) returns contents of Mxd as a
double


% --- Executes during object creation, after setting all properties.
function Mxd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Mxd (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 Myd_Callback(hObject, eventdata, handles)
% hObject    handle to Myd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Myd as text
%        str2double(get(hObject,'String')) returns contents of Myd as a
double


% --- Executes during object creation, after setting all properties.
function Myd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Myd (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




% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)
% hObject    handle to calcular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% global eltos nodos Li Wi Nd Ai  Es fyd fck fcd fyk di1 ri na r Mxd Myd
emax s ec2 ecu2 n
```

```
global b h ns ni nl eltos nodos Li Wi Nd Es fyd fck fcd fyk di1 di2 di3
ri Mxd Myd emax s ec2 ecu2 n

Nd = get(handles.Nd,'string');
Nd = str2double(Nd);
Mxd = get(handles.Mxd,'string');
Mxd = str2double(Mxd);
Myd = get(handles.Myd,'string');
Myd = str2double(Myd);

b = get(handles.b,'string');
b = str2double(b);

h = get(handles.h,'string');
h = str2double(h);

ns = get(handles.ns,'string');
ns = str2double(ns);

ni = get(handles.ni,'string');
ni = str2double(ni);

nl = get(handles.nl,'string');
nl = str2double(nl);

fck = get(handles.fck,'string');
fck = str2double(fck);

fcd=fck/1.5;

fyk = get(handles.fyk,'string');
di1  = get(handles.di1,'string');
di2  = get(handles.di2,'string');
di3  = get(handles.di3,'string');
ri  = get(handles.ri,'string');
fyk = str2double(fyk);
di1 = str2double(di1);
di2 = str2double(di2);
di3 = str2double(di3);
ri = str2double(ri);

% Ai=[2 di1 ri];

Ai=[[ns di1 ri] %Poner una matriz con numero de armaduras en direccin
vertical y numero de armaduras en direccion horizontal
    [ni di2 ri]
    [nl di3 ri]
    [nl di3 ri]
    ];

% ni=2;
fyd=fyk/1.15;
Es=2.0*10^5;

[Li,Wi]=integraHammer(); % integracion de Hammer
[nodos,eltos]=discretiza(); % discretizacion area bruta
[n,ec2,ecu2]=initHormigon(fck); % diagrama parabola-rectangulo Hormigon
```

```
emax=ecu2;

options = optimoptions('fsolve','Display','off');
s0=[0 0 0];
s=fsolve(@(s)calcula(s),s0,options);

vv=esfuerzos(s);
Mxmax=vv(1);

set(handles.result,'string',Mxmax);

if(Mxmax>Mxd)
    set(handles.cumple,'string','CUMPLE');
    set(handles.cumple,'BackgroundColor','green');
else
    set(handles.cumple,'string','NO CUMPLE');
    set(handles.cumple,'BackgroundColor','red');
end
guidata(hObject, handles);

function [Li,Wi]=integraHammer()
    % metode de integracion de Hammer eltos triangulares

    Li=1/2*[
        [1 1 0]
        [1 0 1]
        [0 1 1]
        ]; % coordenadas
    Wi=1/6*[1 1 1]; % pesos


function [n,ec2,ecu2]=initHormigon(fck)
    % diagrama parabola-rectangulo

    if(fck<50)
        n=2;
        ec2=2/1000;
        ecu2=3.5/1000;
    else
        n=1.4+23.4*((90-fck)/100)^4;
        ec2=(2.0+0.085*(fck-50)^0.53)/1000;
        ecu2=(2.6+35*((90-fck)/100)^4)/1000;
    end

function v0=ecmax(s)
    % deformacion maxima en el hormigon
global eltos nodos
    v0=0;
    n0=size(eltos);
    n0=n0(1); % numero de elementos
    for k=1:n0
        vk=eltos(k,:); % elemento k
        nk=nodos(eltos(k,:),:); % nodos del elemento k
        for j=1:3
            x=nk(j,1);
            y=nk(j,2);
            v0=max(v0,defLong(x,y,s));
        end
    end
```

```
function [nodos,eltos]=discretiza()
global b h
% discretizacion
        x0=-b/2:b/10:b/2;
        y0=-h/2:h/10:h/2;

        [X,Y] = meshgrid(x0,y0);

        nx=size(x0);
        nx=nx(2);
        x=reshape(X,nx^2,1);

        ny=size(y0);
        ny=ny(2);
        y=reshape(Y,ny^2,1);
        DT=delaunayTriangulation(x,y); % triangulacion de Delaunay

        nodos=DT.Points; % nodos
        eltos=DT.ConnectivityList; % elementos

function v0=calcula(s)
global eltos nodos acero Nd emax Myd
    % calcula e0 para Nd
    e0=s(1);
    ax=s(2);
    ay=s(3);

    v=zeros(1,3);
    x=0;
    y=0;
    n0=size(eltos);
    n0=n0(1); % numero de elementos
    for k=1:n0
        vk=eltos(k,:); % elemento k
        nk=nodos(eltos(k,:),:); % nodos del elemento k
        vJ=calculaJ(nk);
        v0=intHammer(nk,s);
        v=v+v0*vJ;
    end

    % aportacion del acero
    n0=size(acero);
    n0=n0(1); % numero de armaduras
    for k=1:n0
        x=acero(k,1);
        y=acero(k,2);
        d=acero(k,3);
        A=pi*(d/2)^2; % area
        ts=tensionAcero(x,y,s);
        v=v+A*ts*[1 y x];
    end

    v(1)=v(1)/1000; % axil (kN)
    v(2)=v(2)/10^6; % flector Mx (kNm)
    v(3)=v(3)/10^6; % flector My (kNm)

    v0=zeros(1,2);
    v0(1)=v(1)-Nd;
```

```
    v1=ecmax(s);
    v0(2)=v1-emax;

    v0(3)=v(3)-Myd;

function vJ=calculaJ(nk)
% calcula el jacobiano
% vk: elemento k
% nk: nodos del elemento k
    vJ=zeros(2);
    vJ(1,1)=[1 0 -1]*nk(:,1);
    vJ(1,2)=[1 0 -1]*nk(:,2);

    vJ(2,1)=[0 1 -1]*nk(:,1);
    vJ(2,2)=[0 1 -1]*nk(:,2);

    vJ=det(vJ);

function v=tensionAcero(x,y,s)
global fyd Es ec
    % tension en el Acero
    v=0;

    ec=defLong(x,y,s);
    if(ec<-fyd/Es)
        v=-fyd;
    elseif(-fyd/Es<ec)&&(ec<fyd/Es)
        v=Es*ec;
    else
        v=fyd;
    end

function v=intHammer(nk,s)
global Li Lj x y Wi
% coordenadas (x,y) a partir de (L1,L2,L3)

    v=zeros(1,3);
    for i=1:3
            Lj=Li(i,:);
            x=Lj*nk(:,1);
            y=Lj*nk(:,2);
            v=v+Wi(i)*tensionHormigon(x,y,s)*[1 y x];
    end

function v=tensionHormigon(x,y,s)
global ec2 ec fcd n
    % tension en el Hormigon
    v=0;

    ec=defLong(x,y,s);
    if((ec>0)&&(ec<=ec2))
        v=fcd*(1-(1-ec/ec2)^n);
    elseif(ec>ec2)
        v=fcd;
    end

function v=defLong(x,y,s)
    % plano de deformacion
```

```
    % (x,y): coordenadas en la seccion
    % e0: deformacion axial
    % ax: giro en torno al eje x
    % ay: giro en torno al eje y

    e0=s(1);
    ax=s(2);
    ay=s(3);

    v=e0+ay*x+ax*y;

function v0=esfuerzos(s)
global eltos nodos acero

    % (Nc,Mc): esfuerzos en el hormigón
        e0=s(1);
        ax=s(2);
        ay=s(3);

        v=zeros(1,3);
        x=0;
        y=0;
        n0=size(eltos);
        n0=n0(1); % numero de elementos
        for k=1:n0
            vk=eltos(k,:); % elemento k
            nk=nodos(eltos(k,:),:); % nodos del elemento k
            vJ=calculaJ(nk);
            v0=intHammer(nk,s);
            v=v+v0*vJ;
        end

        % aportacion del acero
        n0=size(acero);
        n0=n0(1); % numero de armaduras
        for k=1:n0
            x=acero(k,1);
            y=acero(k,2);
            d=acero(k,3);
            A=pi*(d/2)^2; % area
            ts=tensionAcero(x,y,s);
            v=v+A*ts*[1 y x];
        end

        v(1)=v(1)/1000; % axil (kN)
        v(2)=v(2)/10^6; % flector Mx (kNm)
        v(3)=v(3)/10^6; % flector My (kNm)

        v0=[v(2) v(3)];

function result_Callback(hObject, eventdata, handles)
% hObject    handle to result (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of result as text
%        str2double(get(hObject,'String')) returns contents of result as
a double
```

```
% --- Executes during object creation, after setting all properties.
function result_CreateFcn(hObject, eventdata, handles)
% hObject    handle to result (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 cumple_Callback(hObject, eventdata, handles)
% hObject    handle to cumple (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cumple as text
%        str2double(get(hObject,'String')) returns contents of cumple as
a double


% --- Executes during object creation, after setting all properties.
function cumple_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cumple (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


% --- Executes on button press in Volver_inicial1.
function Volver_inicial1_Callback(hObject, eventdata, handles)
% hObject    handle to Volver_inicial1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
    clear
    START
```

- **App poligonal**

```
function varargout = flexion_biaxial(varargin)
% FLEXION_BIAXIAL MATLAB code for flexion_biaxial.fig
%      FLEXION_BIAXIAL, by itself, creates a new FLEXION_BIAXIAL or
raises the existing
%      singleton*.
%
%      H = FLEXION_BIAXIAL returns the handle to a new FLEXION_BIAXIAL
or the handle to
%      the existing singleton*.
%
%      FLEXION_BIAXIAL('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in FLEXION_BIAXIAL.M with the given input
arguments.
%
%      FLEXION_BIAXIAL('Property','Value',...) creates a new
FLEXION_BIAXIAL or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before flexion_biaxial_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to flexion_biaxial_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help flexion_biaxial

% Last Modified by GUIDE v2.5 03-Jun-2019 18:51:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @flexion_biaxial_OpeningFcn, ...
                   'gui_OutputFcn',  @flexion_biaxial_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before flexion_biaxial is made visible.
```

```
function flexion_biaxial_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to flexion_biaxial (see VARARGIN)
global cont

cont=0;

a=imread('image.png');
axes(handles.axes1);
imshow(a);

% Choose default command line output for flexion_biaxial
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes flexion_biaxial wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = flexion_biaxial_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function fck_Callback(hObject, eventdata, handles)
% hObject    handle to fck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fck as text
%        str2double(get(hObject,'String')) returns contents of fck as a
double


% --- Executes during object creation, after setting all properties.
function fck_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fck (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


% --- Executes on selection change in mode.
function mode_Callback(hObject, eventdata, handles)
% hObject    handle to mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns mode contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from mode
n = get(handles.mode,'value');
    switch n
        case 1
            set(handles.panel_arm1,'visible','off');
            set(handles.panel_arm2,'visible','off');
            set(handles.panel_arm3,'visible','off');
            set(handles.panel_horm,'visible','on');
            set(handles.flecha_ar,'visible','off');
            set(handles.flecha_ho,'visible','on');

        case 2
            set(handles.panel_arm1,'visible','on');
            set(handles.panel_arm2,'visible','on');
            set(handles.panel_arm3,'visible','on');
            set(handles.panel_horm,'visible','off');
            set(handles.flecha_ar,'visible','on');
            set(handles.flecha_ho,'visible','off');
    end


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

% Hint: popupmenu 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 ri_Callback(hObject, eventdata, handles)
% hObject    handle to ri (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ri as text
%        str2double(get(hObject,'String')) returns contents of ri as a
double
```

```
% --- Executes during object creation, after setting all properties.
function ri_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ri (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 di_Callback(hObject, eventdata, handles)
% hObject    handle to di (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of di as text
%        str2double(get(hObject,'String')) returns contents of di as a
double


% --- Executes during object creation, after setting all properties.
function di_CreateFcn(hObject, eventdata, handles)
% hObject    handle to di (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 fyk_Callback(hObject, eventdata, handles)
% hObject    handle to fyk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fyk as text
%        str2double(get(hObject,'String')) returns contents of fyk as a
double


% --- Executes during object creation, after setting all properties.
function fyk_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fyk (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 n_Callback(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
%        str2double(get(hObject,'String')) returns contents of n as a
double


% --- Executes during object creation, after setting all properties.
function n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (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



% --- Executes on button press in calcul_fig.
function calcul_fig_Callback(hObject, eventdata, handles)
% hObject    handle to calcul_fig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Ai vector_arm Es fyd fck fcd fyk di ri na r cont

cont=cont+1;

if cont==1
    j=helpdlg('POR FAVOR, ASEGÚRESE DE QUE HA INTRODUCIDO CORRECTAMENTE
TODOS LOS VALORES CORRESPONDIENTES AL HORMIGÓN Y LA ARMADURA
(DESPLEGABLE)','Precaución!');
    waitfor(j);
end

na = get(handles.n,'string');
na = str2double(na);

if na<3
```

```
    message = 'ERROR! EL NUMERO DE LADOS HA DE SER AL MENOS 3! INTENTELO
DE NUEVO';
    title = 'error';
    msgbox(message,title,'error')
    return
else
    r=400;

    fck = get(handles.fck,'string');
    fck = str2double(fck);

    fcd=fck/1.5;

    fyk = get(handles.fyk,'string');
    di  = get(handles.di,'string');
    ri  = get(handles.ri,'string');
    fyk = str2double(fyk);
    di = str2double(di);
    ri = str2double(ri);

    Ai=[2 di ri];

    ni=Ai(1);
    fyd=fyk/1.15;
    Es=2.0*10^5;

    if na<19
        [xc,yc] = armado(2*na);
        vector_arm=zeros(ni*na+1,3);
    else
        [xc,yc] = armado(20);
        vector_arm=zeros(20+1,3);
    end

    xc=r*xc; yc=r*yc;

    if na==3
        m=2.7;
    elseif na==4
        m=2;
    elseif na==5
        m=1.7;
    elseif na>5 && na<=18
        m=1.1;
    else
        m=1;
    end

    k1=(m*ri/(cos(pi/(2*na))));
p2x=xc(1)-k1;
k1=p2x/xc(1);

xo = k1*xc;
yo = k1*yc;

xo=xo.';
yo=yo.';

vector_arm(:,1)=xo;
```

```
vector_arm(:,2)=yo;
vector_arm(:,3)=di;


n_arm=size(vector_arm);
        n_arm=n_arm(1);
        for cont=1:n_arm
            a=vector_arm(cont,:);
            di=a(3);
            angulo=0:pi/30:2*pi;
            x_arm=a(1)+di*cos(angulo);y_arm=a(2)+di*sin(angulo);
            % x=x+x0;y=y+y0;
            plot(x_arm,y_arm,'r','LineWidth',1.5);
            axis off
            hold on
        end


[xc,yc] = seccion(na);

xc=r*xc; yc=r*yc;

k2=(ri/(cos(pi/(2*na))));
p2x=xc(1)-k2;
k2=p2x/xc(1);

xo = k2*xc;
yo = k2*yc;


plot(xc,yc,'k','LineWidth',1.5)
% fill(xc,yc,'y')

axis square
axis off

hold off

vector_pol=zeros(na+1,3);

vector_pol(:,1)=xo;
vector_pol(:,2)=yo;
vector_pol(:,3)=di;


set(handles.calcular,'enable','on');

end

     function [x,y] = armado( N )

% POLIGONO: Coordenadas de um poligono regular de N lados
%
% SINTAXE: [x,y] = poligono( N )
%
% onde N representa o numero de lados e x, y
% correspondem as coordenadas dos vertices.
%
```

```
x = [];
y = [];

i = sqrt( -1 );

delta_theta = 2*pi/N;

if N==6
    theta=(0:delta_theta:2*pi)-pi/N;
elseif N==8
    theta=(0:delta_theta:2*pi)-pi/N;
elseif N==10
    theta=(0:delta_theta:2*pi)-pi/N;
else
    theta=(0:delta_theta:2*pi)+pi/((N/2)-2);
end

z = exp( i*theta );

x = real( z );
y = imag( z );


    function [x,y] = seccion( N )

% POLIGONO: Coordenadas de um poligono regular de N lados
%
% SINTAXE: [x,y] = poligono( N )
%
% onde N representa o numero de lados e x, y
% correspondem as coordenadas dos vertices.
%

x = [];
y = [];

i = sqrt( -1 );

delta_theta = 2*pi/N;

theta =(0:delta_theta:2*pi);

z = exp( i*theta );

x = real( z );
y = imag( z );



function Nd_Callback(hObject, eventdata, handles)
% hObject    handle to Nd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nd as text
```

```
%        str2double(get(hObject,'String')) returns contents of Nd as a
double


% --- Executes during object creation, after setting all properties.
function Nd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nd (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 Mxd_Callback(hObject, eventdata, handles)
% hObject    handle to Mxd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Mxd as text
%        str2double(get(hObject,'String')) returns contents of Mxd as a
double


% --- Executes during object creation, after setting all properties.
function Mxd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Mxd (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 Myd_Callback(hObject, eventdata, handles)
% hObject    handle to Myd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Myd as text
%        str2double(get(hObject,'String')) returns contents of Myd as a
double


% --- Executes during object creation, after setting all properties.
function Myd_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to Myd (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



% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)
% hObject    handle to calcular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global eltos nodos Li Wi Nd Ai  Es fyd fck fcd fyk di ri na r Mxd Myd
emax s ec2 ecu2 n

Nd = get(handles.Nd,'string');
Nd = str2double(Nd);
Mxd = get(handles.Mxd,'string');
Mxd = str2double(Mxd);
Myd = get(handles.Myd,'string');
Myd = str2double(Myd);

na = get(handles.n,'string');
na = str2double(na);
r=400;

fck = get(handles.fck,'string');
fck = str2double(fck);

fcd=fck/1.5;

fyk = get(handles.fyk,'string');
di  = get(handles.di,'string');
ri  = get(handles.ri,'string');
fyk = str2double(fyk);
di = str2double(di);
ri = str2double(ri);

Ai=[2 di ri];

ni=2;
fyd=fyk/1.15;
Es=2.0*10^5;

[Li,Wi]=integraHammer(); % integracion de Hammer
[nodos,eltos]=discretiza(na,r); % discretizacion area bruta

[n,ec2,ecu2]=initHormigon(fck); % diagrama parabola-rectangulo Hormigon
emax=ecu2;

options = optimoptions('fsolve','Display','off');
s0=[0 0 0];
s=fsolve(@(s)calcula(s),s0,options);
```

```
vv=esfuerzos(s);
Mxmax=vv(1);

set(handles.result,'string',Mxmax);

if(Mxmax>Mxd)
    set(handles.cumple,'string','CUMPLE');
    set(handles.cumple,'BackgroundColor','green');
else
    set(handles.cumple,'string','NO CUMPLE');
    set(handles.cumple,'BackgroundColor','red');
end
guidata(hObject, handles);

function [Li,Wi]=integraHammer()
    % metode de integracion de Hammer eltos triangulares

    Li=1/2*[
        [1 1 0]
        [1 0 1]
        [0 1 1]
        ]; % coordenadas
    Wi=1/6*[1 1 1]; % pesos


function [n,ec2,ecu2]=initHormigon(fck)
    % diagrama parabola-rectangulo

    if(fck<50)
        n=2;
        ec2=2/1000;
        ecu2=3.5/1000;
    else
        n=1.4+23.4*((90-fck)/100)^4;
        ec2=(2.0+0.085*(fck-50)^0.53)/1000;
        ecu2=(2.6+35*((90-fck)/100)^4)/1000;
    end

function v0=ecmax(s)
    % deformacion maxima en el hormigon
global eltos nodos
    v0=0;
    n0=size(eltos);
    n0=n0(1); % numero de elementos
    for k=1:n0
        vk=eltos(k,:); % elemento k
        nk=nodos(eltos(k,:),:); % nodos del elemento k
        for j=1:3
            x=nk(j,1);
            y=nk(j,2);
            v0=max(v0,defLong(x,y,s));
        end
    end

function [nodos,eltos]=discretiza(n,r)

% discretizacion
    t = (1/(2*n):1/n:1)'*2*pi;
```

```
    x=[];
    y=[];
    for i=1:4
        x=cat(1,x,r/i*cos(t));
        y=cat(1,y,r/i*sin(t));
    end

    DT=delaunayTriangulation(x,y); % triangulacion de Delaunay
%     figure;
%     triplot(DT)

    nodos=DT.Points; % nodos
    eltos=DT.ConnectivityList; % elementos

function v0=calcula(s)
global eltos nodos vector_arm Nd emax Myd
    % calcula e0 para Nd
    e0=s(1);
    ax=s(2);
    ay=s(3);

    v=zeros(1,3);
    x=0;
    y=0;
    n0=size(eltos);
    n0=n0(1); % numero de elementos
    for k=1:n0
        vk=eltos(k,:); % elemento k
        nk=nodos(eltos(k,:),:); % nodos del elemento k
        vJ=calculaJ(nk);
        v0=intHammer(nk,s);
        v=v+v0*vJ;
    end

    % aportacion del acero
    n0=size(vector_arm);
    n0=n0(1); % numero de armaduras
    for k=1:n0
        x=vector_arm(k,1);
        y=vector_arm(k,2);
        d=vector_arm(k,3);
        A=pi*(d/2)^2; % area
        ts=tensionAcero(x,y,s);
        v=v+A*ts*[1 y x];
    end

    v(1)=v(1)/1000; % axil (kN)
    v(2)=v(2)/10^6; % flector Mx (kN·m)
    v(3)=v(3)/10^6; % flector My (kN·m)

    v0=zeros(1,2);
    v0(1)=v(1)-Nd;

    v1=ecmax(s);
    v0(2)=v1-emax;

    v0(3)=v(3)-Myd;
```

```
function vJ=calculaJ(nk)
% calcula el jacobiano
% vk: elemento k
% nk: nodos del elemento k
    vJ=zeros(2);
    vJ(1,1)=[1 0 -1]*nk(:,1);
    vJ(1,2)=[1 0 -1]*nk(:,2);

    vJ(2,1)=[0 1 -1]*nk(:,1);
    vJ(2,2)=[0 1 -1]*nk(:,2);

    vJ=det(vJ);

function v=tensionAcero(x,y,s)
global fyd Es ec
    % tension en el Acero
    v=0;

    ec=defLong(x,y,s);
    if(ec<-fyd/Es)
        v=-fyd;
    elseif(-fyd/Es<ec)&&(ec<fyd/Es)
        v=Es*ec;
    else
        v=fyd;
    end

function v=intHammer(nk,s)
global Li Lj x y Wi
% coordenadas (x,y) a partir de (L1,L2,L3)

    v=zeros(1,3);
    for i=1:3
            Lj=Li(i,:);
            x=Lj*nk(:,1);
            y=Lj*nk(:,2);
            v=v+Wi(i)*tensionHormigon(x,y,s)*[1 y x];
    end

function v=tensionHormigon(x,y,s)
global ec2 ec fcd n
    % tension en el Hormigon
    v=0;

    ec=defLong(x,y,s);
    if((ec>0)&&(ec<=ec2))
        v=fcd*(1-(1-ec/ec2)^n);
    elseif(ec>ec2)
        v=fcd;
    end

function v=defLong(x,y,s)
    % plano de deformacion
    % (x,y): coordenadas en la seccion
    % e0: deformacion axial
    % ax: giro en torno al eje x
    % ay: giro en torno al eje y

    e0=s(1);
```

```
    ax=s(2);
    ay=s(3);

    v=e0+ay*x+ax*y;

function v0=esfuerzos(s)
global eltos nodos vector_arm

    % (Nc,Mc): esfuerzos en el hormigón
    e0=s(1);
    ax=s(2);
    ay=s(3);

    v=zeros(1,3);
    x=0;
    y=0;
    n0=size(eltos);
    n0=n0(1); % numero de elementos
    for k=1:n0
        vk=eltos(k,:); % elemento k
        nk=nodos(eltos(k,:),:); % nodos del elemento k
        vJ=calculaJ(nk);
        v0=intHammer(nk,s);
        v=v+v0*vJ;
    end

    % aportacion del acero
    n0=size(vector_arm);
    n0=n0(1); % numero de armaduras
    for k=1:n0
        x=vector_arm(k,1);
        y=vector_arm(k,2);
        d=vector_arm(k,3);
        A=pi*(d/2)^2; % area
        ts=tensionAcero(x,y,s);
        v=v+A*ts*[1 y x];
    end

    v(1)=v(1)/1000; % axil (kN)
    v(2)=v(2)/10^6; % flector Mx (kN·m)
    v(3)=v(3)/10^6; % flector My (kN·m)

    v0=[v(2) v(3)];

function result_Callback(hObject, eventdata, handles)
% hObject    handle to result (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of result as text
%        str2double(get(hObject,'String')) returns contents of result as
a double


% --- Executes during object creation, after setting all properties.
function result_CreateFcn(hObject, eventdata, handles)
% hObject    handle to result (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 cumple_Callback(hObject, eventdata, handles)
% hObject    handle to cumple (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cumple as text
%        str2double(get(hObject,'String')) returns contents of cumple as
a double


% --- Executes during object creation, after setting all properties.
function cumple_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cumple (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



% --- Executes on button press in Volver_inicial2.
function Volver_inicial2_Callback(hObject, eventdata, handles)
% hObject    handle to Volver_inicial2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
clear
START
```

- **App START**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%           VERIF HORM ARM - MMC                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [] = START()

clc
clear all
```

```
warning off
evalin('base','clear all)');
evalin( 'base', 'clearvars *' )

S.Screen=get(0,'ScreenSize');
S.figdiag=figure('Units','Pixels','Position',[0 0 S.Screen(3)*0.45
S.Screen(4)*0.7],'Name','FLEXION BOX','numbertitle',
'off','Menubar','none','color',[0 0 0],'Resize','off');
set(S.figdiag, 'menubar', 'none');

S.m1=uimenu(S.figdiag,'label','Help');
S.m1a=uimenu(S.m1,'label','SECCION RECTANGULAR');
S.m1b=uimenu(S.m1,'label','SECCION POLIGONAL');
S.m1b1=uimenu(S.m1a,'label','Help','call',{@Help_rect});
S.m2a=uimenu(S.m1b,'label','Help','call',{@Help_poli});
S.m2=uimenu(S.figdiag,'label','About','call',{@About});



S.Posicion=get(gcf,'Position');
S.xr=S.Screen(3)-S.Posicion(3);
S.xp=round(S.xr/2);
S.yr=S.Screen(4)-S.Posicion(4);
S.yp=round(S.yr/2);
set(gcf,'Position',[S.xp S.yp S.Posicion(3) S.Posicion(4)]);
S.axes=axes('Units','Normalized','Position',[0 0 1 1]);
[S.x,S.map]=imread('debut.jpg','jpg');
image(S.x),colormap(S.map),axis off,hold on

S.tx(4)=text(150,1250,'¿QUE SECCION QUIERE
ANALIZAR?','Fontname','Arial','Fontsize',18,'Fontweight','Bold','color',
[0 0 1]);

uicontrol('Style','pushbutton','Units','normalized','Position',[.110 .16
.47 .07],'String','FLEXION BIAXIAL RECTANGULAR','Foregroundcolor',[0 0
0],'Fontsize',10,'Fontweight','Bold','Callback','clear all; close
all','call',{@rectangular});
uicontrol('Style','pushbutton','Units','normalized','Position',[.110
.065 .47 .07],'String','FLEXION BIAXIAL POLIGONAL','Foregroundcolor',[0
0 0],'Fontsize',10,'Fontweight','Bold','Callback','clear all; close
all','call',{@poligonal});



function [] = rectangular(varargin)
    close all
    clear
    flexion_biaxial_rectangular

function [] = poligonal(varargin)
    close all
    clear
    flexion_biaxial

function [] = Help_rect(varargin)
 winopen('App flexión biaxial con sección rectangular.pdf')

function [] = Help_poli(varargin)
 winopen('App flexión biaxial con sección poligonal.pdf')

 function [] = About(varargin)
```

```
image=imread('debut.jpg');
msgbox({'     ************************************',...
    '     FLEXION BOX',...
    '     ',...
        '     Version 1.0',...
    '     ',...
    '     Last update: 08/05/2019',...
    '     ',...
    '     ************************************'},'FLEXION
BOX','custom',image)
```