Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

TREBALL FINAL DE GRAU

Desarrollo de un software de cálculo de placas mediante métodos directos

Treball realitzat per:
DAVID CODONY GISBERT

Dirigit per:
LUIS MIGUEL CERVERA RUIZ
JOAN BAIGES AZNAR

Grau en:
ENGINYERIA CIVIL

Barcelona, 20 de juny de 2014

Departament de Resistència de Materials i Estructures a l'Enginyeria

AGRADECIMIENTOS

Me gustaría mencionar a ciertas personas que me han ayudado o han influido de alguna manera durante el proceso de realización de este trabajo.

En primer lugar, a mis dos tutores *Luis Miguel Cervera Ruiz* y *Joan Baiges Aznar*, por la dedicación y el tiempo que han invertido en este trabajo para ayudarme en momentos clave y guiarme desde el primer día.

En segundo lugar, a mi familia, que me ha apoyado durante todo este proceso de diseño, realización y redacción, y se ha ofrecido siempre para estar a mi lado.

También debo agradecer a los amigos y compañeros que en algún momento y de alguna manera se han interesado por mi trabajo y mi situación personal, y me han dado ánimos para que este trabajo haya podido alcanzar sus principales objetivos.

Por último, pero no menos importante, quería destacar al catedrático *J.Miquel Canet*, por su asesoramiento y ayuda ofrecida de manera desinteresada en momentos puntuales del trabajo.

Muchas gracias a todos.

ABSTRACT

This project is based on the design and the programming of a software able to calculate thin plates using *Matlab.* The **main goal** of this project is to deploy a useful, practical and trustworthy tool, in order to complement the teaching in this specific area of the resistance of materials.

The **mathematical resolution** of the problem is focused on the direct resolution methods for the *Kirchoff-Love* thin plate's theory, such as the *Navier* and *Levy* methods. Both are based on the decomposition of the functions used in terms of sinus *Fourier* series. In order to implement them in our software, both have been studied and rewritten in the most general way, so they can calculate any kind of load we could analytically write in terms of *Fourier* series. The kind of problems that our software would be able to solve is conditioned to the own limitations of these methods.

The **software** has been designed with a very simple, visual and practical layout, such that ables the final user to introduce the necessary data in a simple and easy – but rigorous- way. Moreover, the **user's manual** has been created in order to help the user in this task.

The **final solution** is given in terms of displacement, angles, bending and torsion moments and shearing forces. Special focus on the interaction of the user with the results has been made: On the one hand the user is able to see them tridimensionally, from any point of view. Linear analyse of the results by a section across any coordinate is also possible. On the other hand, user is also able to get the location and the value of the maxima and the minima of the solutions, and also they can be evaluated in any point inside the plate.

This project also contains an **explanation of the programming structure** in *Matlab*, a list of the entry and exit variables and a global schema that describes the internal structure of the software.

The **goodness** of the software has been proved in the different convergence analysis and the comparative studies that have been developed. Using our software for computing the solution, we have obtained exactly the same results that in the exercises solved by hand by other authors.

In **conclusion**, we can say that the desired tool has been obtained, and that it could be perfectly used by teaching staff as a complement in the learning process of the students in this subject.

RESUMEN

Este trabajo se basa en el diseño y la programación de un software de cálculo de placas delgadas mediante *Matlab*. El **objetivo principal** de este trabajo es proporcionar una herramienta útil, práctica y fiable para complementar la docencia en este ámbito de la resistencia de materiales.

La **resolución matemática** del problema se ha centrado en los métodos directos de resolución de la teoría de placas de *Kirchoff-Love*, esto es, el método de *Navier* y el de *Levy*. Ambos se basan en el desarrollo en series de senos de Fourier de las funciones que intervienen. Para poder implementarlos en un software, ambos han sido estudiados en profundidad y reescritos de la manera más general posible, de modo que admitan cualquier tipo de carga que sea descomponible analíticamente en series de senos de Fourier. La casuística de diferentes problemas que puede resolver el software viene condicionada por las propias limitaciones que tienen cada uno de los métodos.

El **software** ha sido diseñado con una interfaz práctica, visual y a la vez simple, de manera que el usuario final pueda introducir los datos necesarios de una manera fácil y rápida, pero a la vez rigurosa. De todas formas el usuario cuenta con un **manual de instrucciones** a su disposición.

La **solución final** cuenta con la flecha, los giros, los momentos y los cortantes que producen las cargas introducidas. Se ha hecho un especial énfasis en la interacción del usuario con los resultados: por un lado pueden visualizarse de forma tridimensional, desde cualquier punto de vista. También se pueden hacer análisis lineales realizando secciones por cualquier coordenada de la solución encontrada. Por otro lado, se puede obtener la ubicación y el valor de los máximos y los mínimos de la solución, y se puede calcular el valor de la solución en cualquier punto específico deseado.

El trabajo cuenta también con una **explicación de la programación** en *Matlab*, un listado de las variables de entrada y de salida y un esquema global que describe la estructura interna del programa.

La **fiabilidad** del software ha quedado demostrada con los análisis de convergencia y los estudios comparativos con ejemplos ya resueltos, en los que se han obtenido para que cualquier caso exactamente los mismos resultados.

En **conclusión**, se ha obtenido la herramienta deseada, que puede ser usada perfectamente por personal docente para completar la formación de los alumnos en éste ámbito concreto de la resistencia de materiales.

ÍNDICE

INTRODUCCIÓN Y OBJETIVOS

Este trabajo trata sobre el desarrollo mediante *Matlab* de un *software* útil para calcular placas delgadas mediante dos métodos directos: el método de Navier y el método de Levy.

El trabajo se ha estructurado en dos partes fundamentales: el **desarrollo del *software*** en sí y la **redacción de este documento**.

Por un lado, el *software* está dividido en dos secciones principales: el **preproceso** y el **postproceso**. En la primera el usuario introduce los datos necesarios (o los carga) para realizar el cálculo; la segunda muestra los resultados una vez se ha procesado la información del preproceso y se han hecho los cálculos pertinentes.

Por otro lado, éste mismo documento se divide en cuatro partes fundamentales, que son su eje principal. Dichas partes, también llamadas capítulos, son:

I. Descripción detallada de la teoría de placas clásica y de métodos de resolución conocidos.
II. Elaboración de un manual de usuario que lo guíe por dentro del software.
III. Descripción detallada del funcionamiento interno del programa y de los diferentes códigos que lo forman
IV. Validación de resultados del programa y análisis posteriores a su desarrollo.

Uno de los grandes objetivos de este trabajo es **proporcionar una herramienta** útil de cálculo de placas, sobretodo **en el ámbito docente**. La idea está en que sea un *software* **robusto** y **fiable,** que tanto el personal docente de este sector concreto de la resistencia de materiales como sus alumnos puedan explotar y utilizar para complementar la formación en este aspecto.

Otro objetivo importante es que el programa tenga una **interfaz visual, cómoda** y **práctica** para el usuario, de manera que la navegación por él, la introducción de datos de partida y la interpretación de los resultados finales sean tareas fáciles de realizar, sin necesidad incluso de leerse el manual adjunto.

NOTA: *Se le ha querido dar a este documento un cierto orden a la vez que una cierto simplicidad, de manera que la lectura por parte de la persona que lo tenga entre manos sea lo más fácil posible. Es por eso que cada capítulo que forma este documento cuenta con una numeración propia de apartados.*

NOTA: *Las imágenes y las ecuaciones que aparecen en este trabajo están todas numeradas por separado, con el número de capítulo seguido de una numeración propia.*

NOTA: *Todas las imágenes que aparecen en este trabajo provienen completamente de elaboración propia, si no existe en su pie ninguna referencia a una de las fuentes descritas en la bibliografía.*

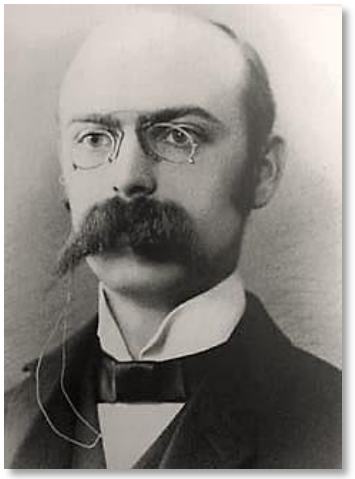
CAPÍTULO I

TEORÍA DE PLACAS DE KIRCHHOFF – LOVE

1. ECUACIÓN DIFERENCIAL DE EQUILIBRIO

1.1. INTRODUCCIÓN

I.01: *G. Kirchhoff (1824-1887).* ***Fuente [21]***



I.02: *A. E. H. Love (1863-1940).* ***Fuente [19]***

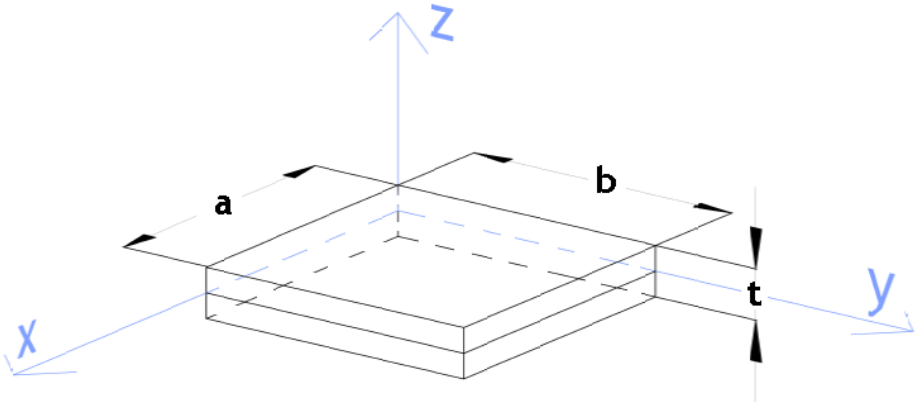
La Teoría de Placas de Kirchhoff-Love fue desarrollada el año 1888 por *Augustus Edward Hough Love* a partir de los estudios que *Gustav Kirchoff* había realizado en 1850. Hoy en día sigue siendo un modelo válido, aunque aproximado, para el cálculo de placas delgadas[1].

Las placas son paralelepípedos considerados como sólidos deformables, que presentan una dimensión mucho menor que las otras dos. Son elementos estructurales que antes de deformarse presentan una forma plana.

Se define como **plano medio** de la placa a la superficie equidistante a las dos caras del sólido que presentan las dos dimensiones mayores.

La placa presenta un cierto **espesor** (t) constante, que es la dimensión más pequeña en relación a las otras dos.

Para el desarrollo matemático es conveniente situar los ejes de coordenadas dextrógiros, en una esquina de la placa y a la altura del plano medio. El sistema de referencia usado será el siguiente:



I.03: *Placa genérica. Espesor, plano medio y situación de los ejes de coordenadas*

Este modelo de Kirchhoff-Love está basado en la teoría de la elasticidad clásica, aunque incorpora ciertas hipótesis que se deben asumir para simplificar el problema tridimensional de forma significativa.

[1] Se considera *placa delgada* aquella en la que el espesor no supera el 10% del ancho de la placa.

1.2. HIPÓTESIS

*Ver **fuentes [01]-[08]***

a) *El material que forma la placa es elástico, isótropo y homogéneo*

Es decir, se puede aplicar la teoría clásica de la elasticidad[2] que depende de los parámetros E y γ.

b) *Hipótesis de pequeñas deformaciones*

La flecha máxima admisible será como mucho del orden del 10% del espesor.

c) *La deformación por cortante se supone despreciable*

(I.01) (I.02)

Así pues, sólo se considera la deformación por flexión. Es decir, los puntos que pertenecen al plano medio sólo se pueden mover en dirección perpendicular al mismo.

$$u(x,y,0) = 0$$

$$v(x,y,0) = 0$$

d) *Hipótesis de normalidad*

Los puntos situados antes de la deformación sobre una recta perpendicular al plano medio siguen, después de la deformación, encima de la recta perpendicular al plano medio deformado.

e) *La deformación vertical se considera despreciable*

(I.03) (I.04)

Todos los puntos situados sobre recta perpendicular al plano medio tienen la misma flecha.

$$\varepsilon_z = 0$$

$$w(x,y,z) = w(x,y)$$

f) *La tensión normal al plano medio σ_z se considera despreciable*

(I.05)

Esta hipótesis nos permitirá aplicar la teoría de la elasticidad particularizada para la tensión plana.

$$\sigma_z = 0$$

Estas hipótesis permiten, entre otras cosas, tratar un problema tridimensional como si fuese un problema bidimensional, ya que los desplazamientos, deformaciones, tensiones y esfuerzos en el plano medio sólo depende de dos parámetros: x e y.

Así pues, nuestro elemento de estudio va a ser el plano medio de la placa y no la placa en sí. Pero por el momento trabajaremos con la placa entera.

[2] Ver anejo: Teoría clásica de la elasticidad.

1.3. DESPLAZAMIENTOS

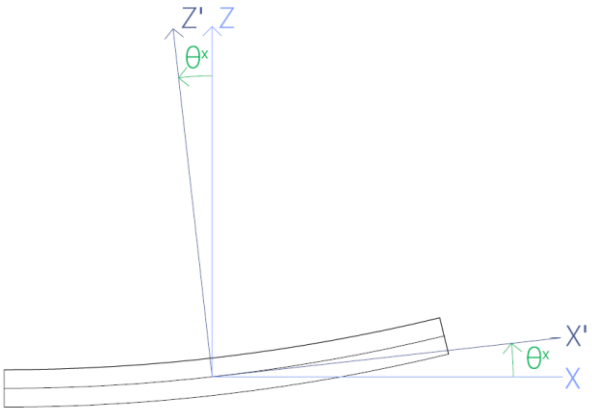
Los desplazamientos (u, v, w) se pueden concretar de una manera fácil si tenemos en cuenta las hipótesis anteriores.

a) *Desplazamiento en el eje x*

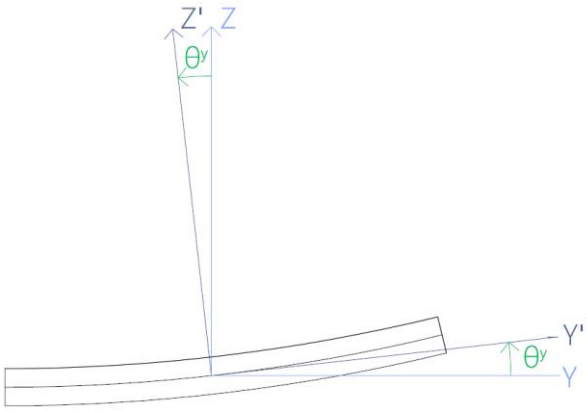
La hipótesis *c)* no permite el desplazamiento *u* en el eje x en el plano medio; es decir, cualquier desplazamiento horizontal debe darse fuera (por encima o por debajo) del plano medio. Las hipótesis *b)* y *d)* nos indican que este movimiento debe ser **lineal** con respecto a z (a medida de lo que nos alejamos del plano medio) y nos permiten aproximar el ángulo que forman las dos normales $\Theta x(x, y)$ por la derivada de la flecha *w* con respecto a x.

(I.06)

$$u(x, y, z) = -z \cdot \Theta x(x, y) = -z \cdot \frac{\partial w(x, y)}{\partial x}$$



I.04: *Desplazamiento en x:* $u\,(x, y, z)$



I.05: *Desplazamiento en y:* $v\,(x, y, z)$

b) *Desplazamiento en el eje y*

Este caso es análogo al anterior:

(I.07)

$$v(x, y, z) = -z \cdot \Theta y(x, y) = -z \cdot \frac{\partial w(x, y)}{\partial y}$$

c) *Desplazamiento en el eje z*

Es nuestra principal incógnita. Sólo se puede aplicar la hipótesis *e)*:

(I.08)

$$w(x, y, z) = w(x, y)$$

1.4. DEFORMACIONES

Las deformaciones se pueden calcular rápidamente a partir de los desplazamientos calculados anteriormente:

$$\mathcal{E}x(x,y,z) = \frac{\partial u(x,y,z)}{\partial x} = -z \cdot \frac{\partial^2 w(x,y)}{\partial x^2} \qquad (I.09)$$

$$\mathcal{E}y(x,y,z) = \frac{\partial u(x,y,z)}{\partial y} = -z \cdot \frac{\partial^2 w(x,y)}{\partial y^2} \qquad (I.10)$$

$$\mathcal{E}z(x,y,z) = \frac{\partial w(x,y)}{\partial \mathrm{z}} = 0 \qquad (I.11)$$

$$\gamma xy(x,y,z) = \frac{\partial u(x,y,z)}{\partial y} + \frac{\partial v(x,y,z)}{\partial x} = -2z \cdot \frac{\partial^2 w(x,y)}{\partial x\, \partial y} \qquad (I.12)$$

$$\gamma xz(x,y,z) = \frac{\partial u(x,y,z)}{\partial z} + \frac{\partial w(x,y)}{\partial x} = 0 \qquad (I.13)$$

$$\gamma yz(x,y,z) = \frac{\partial v(x,y,z)}{\partial z} + \frac{\partial w(x,y)}{\partial y} = 0 \qquad (I.14)$$

Se puede observar que, tal y como marcaba la hipótesis *c)*, no se ha considerado la deformación por cortante, lo que nos da unas deformaciones nulas en los planos perpendiculares al plano medio.

Las deformaciones se producen en planos paralelos al plano medio, y crecen linealmente a medida que nos alejamos de éste. En el plano medio ($z = 0$) no se produce deformación alguna.

1.5. TENSIONES

Las tensiones producidas en la placa se pueden calcular fácilmente una vez calculadas las deformaciones en la placa, a través de las hipótesis *a)* y *f)*, mediante las cuales podemos aplicar la teoría de la elasticidad clásica particularizada para la tensión plana[3].

(I.15)

$$\begin{bmatrix} \sigma x \\ \sigma x \\ \tau xy \end{bmatrix} = \frac{E}{(1-\nu)\cdot(1+\nu)} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \begin{bmatrix} \mathcal{E}x \\ \mathcal{E}y \\ \gamma xy \end{bmatrix}$$

Con:

$$\sigma z = 0; \qquad \tau xz = 0; \qquad \tau yz = 0$$

Por lo tanto se obtiene lo siguiente:

(I.16)

$$\sigma x(x,y,z) = -z \cdot \frac{E}{1-\nu^2} \cdot \left(\frac{\partial^2 w(x,y)}{\partial x^2} + \nu \cdot \frac{\partial^2 w(x,y)}{\partial y^2} \right)$$

(I.17)

$$\sigma y(x,y,z) = -z \cdot \frac{E}{1-\nu^2} \cdot \left(\frac{\partial^2 w(x,y)}{\partial y^2} + \nu \cdot \frac{\partial^2 w(x,y)}{\partial x^2} \right)$$

(I.18)

$$\sigma z(x,y,z) = 0$$

(I.19)

$$\tau xy(x,y,z) = -z \cdot \frac{E}{1+\nu} \cdot \left(\frac{\partial^2 \mathrm{w(x,y)}}{\partial \mathrm{x}\, \partial \mathrm{y}} \right)$$

(I.20)

$$\tau xz(x,y,z) = 0$$

(I.21)

$$\tau yz(x,y,z) = 0$$

Las conclusiones son análogas a las extraídas en el apartado de deformaciones.

Las tensiones en los planos perpendiculares al plano medio son nulas, y las tensiones no nulas se producen en planos paralelos al plano medio, y crecen linealmente a medida que nos alejamos de éste. En el plano medio ($z = 0$) no se produce tensión alguna.

[3] Ver anejo: Teoría clásica de la elasticidad – Tensión Plana.

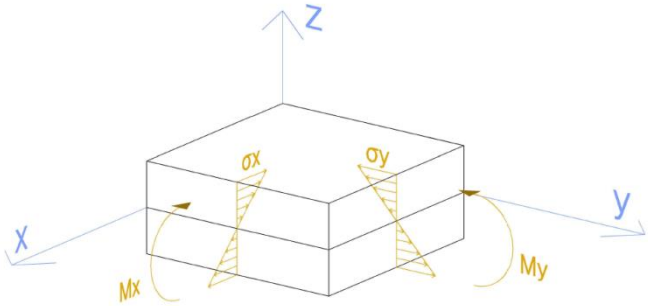
1.6. ESFUERZOS SOBRE EL PLANO MEDIO

Éste es el apartado en el que pasaremos a trabajar con el plano medio, en vez de con toda la placa.

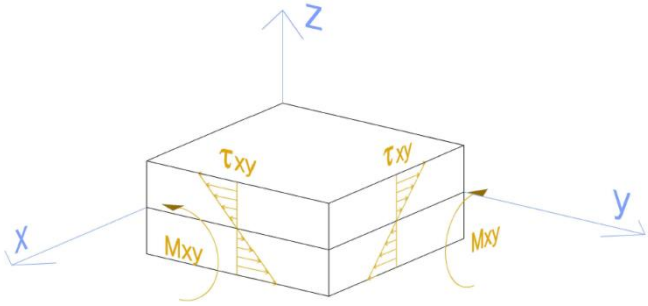
Perderemos por tanto la dependencia de las incógnitas con la coordenada *z*.

Se trata de integrar las tensiones internas que han aparecido en la placa a lo largo del eje z, de manera que se puedan obtener los esfuerzos a los que se ve sometido el plano medio.

Como se puede ver en las figuras *I.06* y *I.07*, las tensiones de la placa son estáticamente equivalentes a momentos. Las tensiones normales equivalen a **momentos de flexión** interna y las tensiones tangenciales equivalen a **momentos torsores** internos.



I.06: *Tensiones normales y momentos de flexión*



I.07: *Tensiones tangenciales y momentos de torsión*

Tal y como están definidos los momentos flectores y torsores en las figuras *I.06* y *I.07*, su formulación es la siguiente:

$$Mx(x,y) = \int_{-\frac{t}{2}}^{\frac{t}{2}} -z \cdot \sigma x(x,y,z)dz = \cdots = \qquad (I.22)$$

$$= D \cdot \left(\frac{\partial^2 w(x,y)}{\partial x^2} + \nu \cdot \frac{\partial^2 w(x,y)}{\partial y^2} \right)$$

$$My(x,y) = \int_{-\frac{t}{2}}^{\frac{t}{2}} -z \cdot \sigma y(x,y,z)dz = \cdots = \qquad (I.23)$$

$$= D \cdot \left(\frac{\partial^2 w(x,y)}{\partial y^2} + \nu \cdot \frac{\partial^2 w(x,y)}{\partial x^2} \right)$$

(I.24)

$$Mxy(x,y) = \int_{-\frac{t}{2}}^{\frac{t}{2}} -z \cdot \tau xy(x,y,z)dz = \cdots =$$

$$= D \cdot (1-\nu) \cdot \frac{\partial^2 w(x,y)}{\partial x\, \partial y}$$

En las ecuaciones *(I.22)*, *(I.23)* y *(I.24)* vistas anteriormente aparece el término D.

(I.25)

$$\boxed{D = \frac{E \cdot t^3}{12 \cdot (1-\nu)^2}}$$

D es el coeficiente que caracteriza a la placa desde el punto de vista resistente. En la formulación diferencial de placas juega un papel muy semejante al papel que juega en la formulación de vigas el término EI. Tiene unidades de FL (fuerza por distancia).

También es importante remarcar que las ecuaciones *(I.22)*, *(I.23)* y *(I.24)* nos expresan el valor de los momentos internos **en función del punto** $(\boldsymbol{x}, \boldsymbol{y})$ encima del plano medio. Es decir, este momento que nos entregan no está expresado en unidades de momento propiamente, sino de **momento por unidad de longitud** FL/L ('fuerza por distancia' por unidad de distancia).

El momento al que está sometida una fibra del plano medio (es decir, un plano vertical de la placa) se obtendría integrando estas ecuaciones con respecto a la dirección que marque la fibra observada; y éste sí que tendría unidades de momento FL.

NOTA: *Estas ecuaciones no son iguales que las que encontramos en la bibliografía (ver* ***fuentes [01]-[08]****); difieren en el signo. Esto se debe simplemente a que se ha elegido un convenio de signos y un sistema de referencia distinto. Obviamente, la correcta interpretación de estas ecuaciones está ligada a los convenios de signos citados en este mismo documento (figuras* ***I.3****,* ***I.6*** *y* ***I.7****).*

1.7. ECUACIONES DE EQUILIBRIO

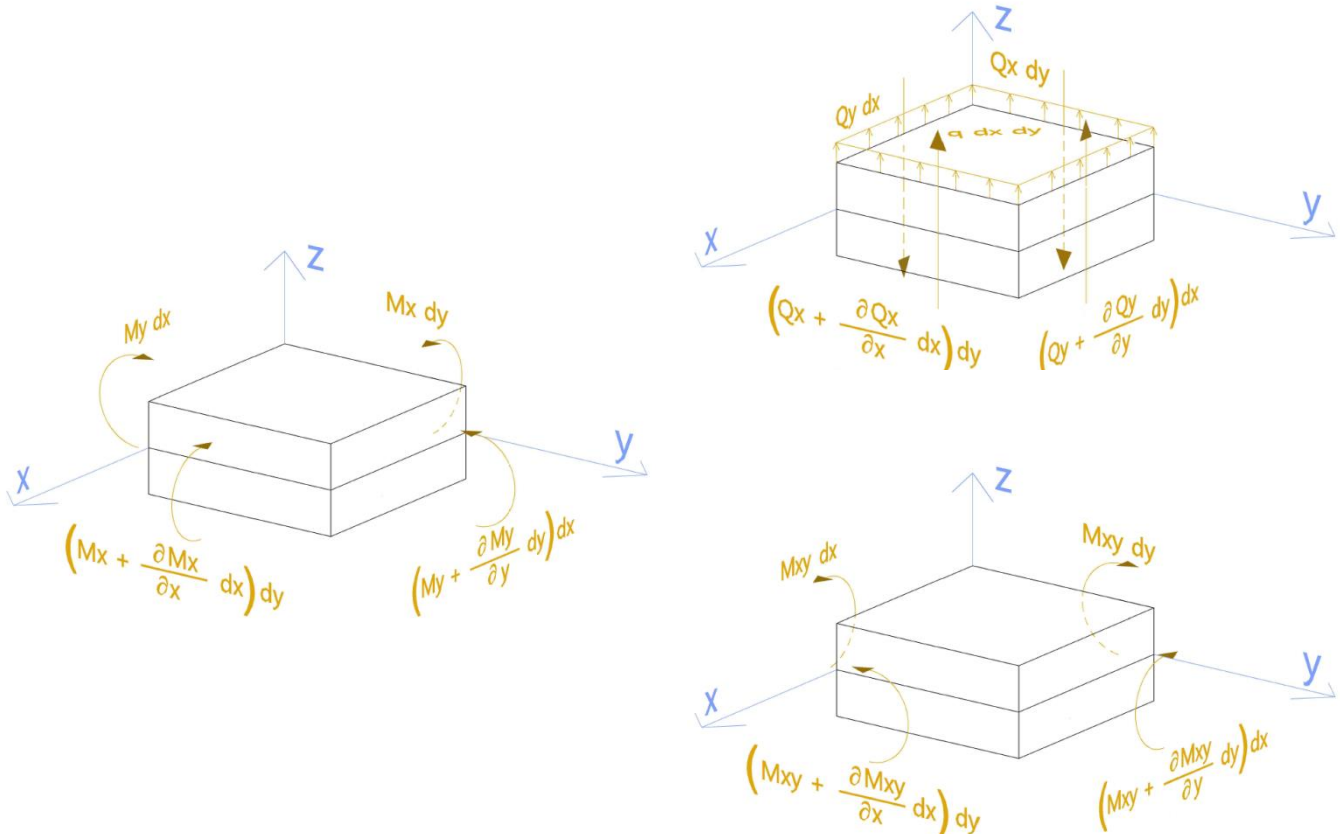
Este apartado trata de verificar el equilibrio estático entre fuerzas y momentos que se da en un diferencial genérico de placa de dimensiones dx por dy.

Los esfuerzos que intervienen en las ecuaciones que vamos a desarrollar a continuación son los siguientes:

- Momentos flectores y torsores descritos anteriormente: $Mx(x,y)$, $My(x,y)$ y $Mxy(x,y)$, resultado de integrar las tensiones **internas** que se crean durante la deformación.

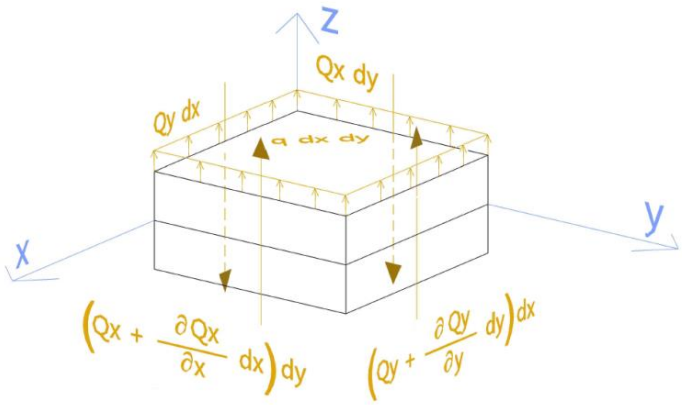
- Carga **externa** aplicada $q(x,y)$, positiva por defecto en dirección z.

- Esfuerzos cortantes **internos** $Qx(x,y)$, $Qy(x,y)$. Estos esfuerzos son presentes durante la deformación de una placa, y aunque se desprecia la deformación que éstos producen, no se pueden obviar en el cálculo estático, ya que son físicamente necesarios para que se cumpla el equilibrio de fuerzas y momentos.



I.08: *Esfuerzos y cargas que intervienen en las ecuaciones de equilibrio estático*

a) *Equilibrio de fuerzas en el eje z.*



I.09: *Fuerzas en dirección perpendicular al plano medio*

La primera condición de equilibrio que se debe imponer en el cuerpo diferencial es que la resultante de todas las fuerzas existentes es nula.

Las fuerzas paralelas al plano medio son inexistentes, gracias a las hipótesis admitidas. Así pues, sólo es necesario imponer esta condición en dirección perpendicular al plano medio:

$$\sum F(z) = 0$$

$$\downarrow$$

$$q \cdot dx \cdot dy - Qx \cdot dy + \left(Qx + \frac{\partial Qx}{\partial x} \cdot dx\right) \cdot dy - Qy \cdot dx + $$
$$+ \left(Qy + \frac{\partial Qy}{\partial y} \cdot dy\right) \cdot dx = q \cdot dx \cdot dy + \frac{\partial Qx}{\partial x} \cdot dx \cdot dy +$$
$$+ \frac{\partial Qy}{\partial y} \cdot dy \cdot dx = \left(q + \frac{\partial Qx}{\partial x} + \frac{\partial Qy}{\partial y}\right) \cdot dy \cdot dx = 0$$

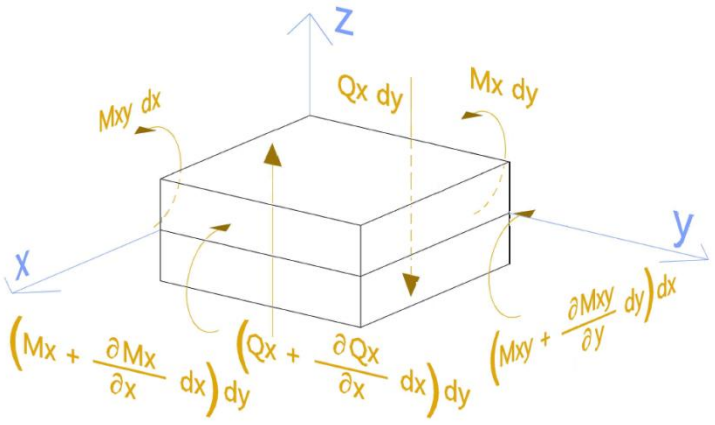
$$\downarrow$$

$$\boxed{q + \frac{\partial Qx}{\partial x} + \frac{\partial Qy}{\partial y} = 0} \qquad (I.26)$$

b) *Equilibrio de momentos en el plano XZ.*

La segunda condición que se debe imponer para conseguir el equilibrio estático del cuerpo diferencial es que se cumpla el equilibrio de momentos.

No existen momentos contenidos en un plano paralelo al plano medio, por lo que esta condición deberá ser aplicada en los planos perpendiculares a éste.



I.10: *Fuerzas y momentos que intervienen en el equilibrio de momentos en el plano XZ*

$$\sum M(xz) = 0$$

$$\downarrow$$

$$Qx \cdot dy \cdot \frac{dx}{2} + \left(Qx + \frac{\partial Qx}{\partial x} \cdot dx\right) \cdot dy \cdot \frac{dx}{2} - Mx \cdot dy +$$

$$\left(Mx + \frac{\partial Mx}{\partial x} \cdot dx\right) \cdot dy - Mxy \cdot dx + \left(Mxy + \frac{\partial Mxy}{\partial y} \cdot dy\right) \cdot dx =$$

$$= Qx \cdot dy \cdot dx + \frac{\partial Qx}{\partial x} \cdot \frac{dx}{2} \cdot dy \cdot dx + \frac{\partial Mx}{\partial x} \cdot dy \cdot dx +$$

$$\frac{\partial Mxy}{\partial y} \cdot dy \cdot dx = \left(Qx + \frac{\partial Qx}{\partial x} \cdot \frac{dx}{2} + \frac{\partial Mx}{\partial x} + \frac{\partial Mxy}{\partial y}\right) \cdot dy \cdot dx =$$

$$= \left(Qx + \frac{\partial Mx}{\partial x} + \frac{\partial Mxy}{\partial y}\right) \cdot dy \cdot dx = 0$$

$$\downarrow$$

$$\boxed{Qx + \frac{\partial Mx}{\partial x} + \frac{\partial Mxy}{\partial y} = 0} \qquad (I.27)$$

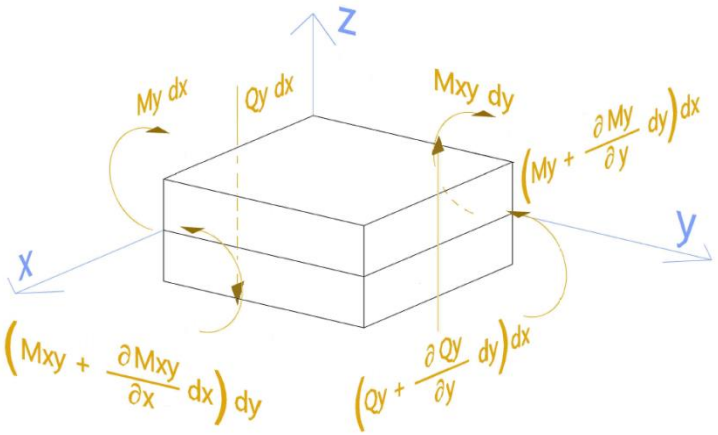
c) *Equilibrio de momentos en el plano XZ*

Este apartado se desarrolla de forma análoga al anterior, y se obtiene lo siguiente:

$$\sum M(yz) = 0$$

$$\downarrow$$

$$\boxed{Qy + \frac{\partial My}{\partial y} + \frac{\partial Mxy}{\partial x} = 0} \qquad (I.28)$$



I.11: *Fuerzas y momentos que intervienen en el equilibrio de momentos en el plano YZ*

1.8. ESFUERZOS CORTANTES

Tal y como se ha dicho en el apartado 1.7, los esfuerzos cortantes existen.

Una vez realizado un análisis de equilibrio estático, se pueden conocer los valores que toman estos cortantes en cada punto del plano medio mediante las ecuaciones *(I.27)* y *(I.28)*, de la siguiente manera:

(I.29)

$$Qx(x,y) = -\frac{\partial Mx}{\partial x} - \frac{\partial Mxy}{\partial y} =$$
$$= -D\left(\frac{\partial^3 w(x,y)}{\partial x^3} + \nu \cdot \frac{\partial^3 w(x,y)}{\partial x\,\partial y^2} + (1-\nu)\cdot \frac{\partial^3 w(x,y)}{\partial x\,\partial y^2}\right) =$$
$$= -D\cdot\left(\frac{\partial^3 w(x,y)}{\partial x^3} + \frac{\partial^3 w(x,y)}{\partial x\,\partial y^2}\right)$$

(I.30)

$$Qy(x,y) = -\frac{\partial My}{\partial y} - \frac{\partial Mxy}{\partial x} =$$
$$= -D\left(\frac{\partial^3 w(x,y)}{\partial y^3} + \nu \cdot \frac{\partial^3 w(x,y)}{\partial y\,\partial x^2} + (1-\nu)\cdot \frac{\partial^3 w(x,y)}{\partial y\,\partial x^2}\right) =$$
$$= -D\cdot\left(\frac{\partial^3 w(x,y)}{\partial y^3} + \frac{\partial^3 w(x,y)}{\partial y\,\partial x^2}\right)$$

Estos esfuerzos cortantes corresponden al criterio de signos utilizado en la figura ***I.08***.

Tal y como sucedía con los esfuerzos flectores y torsores, los esfuerzos cortantes también están definidos en cada punto del plano medio y por unidad de longitud; tienen, por tanto, unidades de **cortante por unidad de longitud** F/L.

El valor del esfuerzo cortante al que está sometida una fibra del plano medio (o un plano vertical de la placa) se encontraría integrando las ecuaciones *(I.29)* y *(I.30)* con respecto a la dirección de la fibra, y éste valor estaría expresado en unidades de cortante F.

1.9. ECUACIÓN DIFERENCIAL DE EQUILIBRIO

*Ver **fuentes [01]-[08]***

Finalmente llegamos al apartado más relevante de la teoría de Kirchhoff-Love. Aquí se va a enunciar la ecuación diferencial que relaciona las cargas aplicadas y la rigidez de la placa con la flecha observada en la deformación.

Insertamos las ecuaciones *(I.29)* y *(I.30)* en la ecuación *(I.26)* para obtener lo siguiente:

$$q = -\frac{\partial Qx}{\partial x} - \frac{\partial Qy}{\partial y} =$$

$$= D \cdot \left(\frac{\partial^4 w}{\partial x^4} + \frac{\partial^4 w}{\partial x^2\,\partial y^2}\right) + D \cdot \left(\frac{\partial^4 w}{\partial y^4} + \frac{\partial^4 w}{\partial y^2\,\partial x^2}\right) =$$

$$= D \cdot \left(\frac{\partial^4 w}{\partial x^4} + 2 \cdot \frac{\partial^4 w}{\partial x^2\,\partial y^2} + \frac{\partial^4 w}{\partial y^4}\right) = D \cdot \Delta\Delta w$$

$$\downarrow$$

$$\boxed{\Delta\Delta w(x,y) = \frac{q(x,y)}{D}} \qquad \textit{(I.31)}$$

$$con \quad \Delta(*) = \frac{\partial^2}{\partial x^2}(*) + \frac{\partial^2}{\partial y^2}(*) \qquad \textit{(I.32)}$$

Resolviendo esta ecuación diferencial podemos encontrar la flecha $w(x,y)$ que corresponde a una placa caracterizada con el parámetro D sometida a una carga $q(x,y)$. Y una vez encontrada la flecha, se pueden obtener giros, momentos flectores, momentos torsores y esfuerzos cortantes a través de sus respectivas ecuaciones ya descritas.

Más adelante hablaremos de métodos de resolución de la ecuación diferencial de equilibrio. Pero lo que está claro es que la solución física va a depender de **las condiciones de contorno** que se impongan en los extremos. Estas condiciones de contorno se reflejan matemáticamente en las constantes de integración que nos van a aparecer al resolver la ecuación diferencial.

*Ver **fuentes [01]-[08]***

1.10. CONDICIONES DE CONTORNO

En este apartado se verán las condiciones de contorno más usuales que se deben satisfacer en los extremos de la placa.

De manera análoga a la teoría de vigas de Euler-Bernoulli, cada extremo viene dado por dos condiciones de contorno.

Se toma como ejemplo el contorno $x=a$ (ídem para $y=b$).

a) Contorno empotrado

El contorno empotrado se caracteriza por tener flecha nula y giro perpendicular Θx nulo:

(I.33)

$$\begin{cases} w(x,y) = 0 \\ \dfrac{\partial w(x,y)}{\partial x} = 0 \end{cases} \quad \text{Para } x = a; \quad \text{para } \forall y$$

b) Contorno simplemente apoyado

El contorno simplemente apoyado se caracteriza por tener flecha nula y momento perpendicular Mx nulo:

(I.34)

$$\begin{cases} w(x,y) = 0 \\ D \cdot \left(\dfrac{\partial^2 w(x,y)}{\partial x^2} + \nu \cdot \dfrac{\partial^2 w(x,y)}{\partial y^2} \right) = 0 \end{cases} \quad \text{Para } x = a; \quad \text{para } \forall y$$

La segunda condición se puede simplificar si tenemos en cuenta que la curvatura en sentido del eje Y es nula en $x = a$, puesto que la flecha en $x = a$ también es nula:

(I.35)

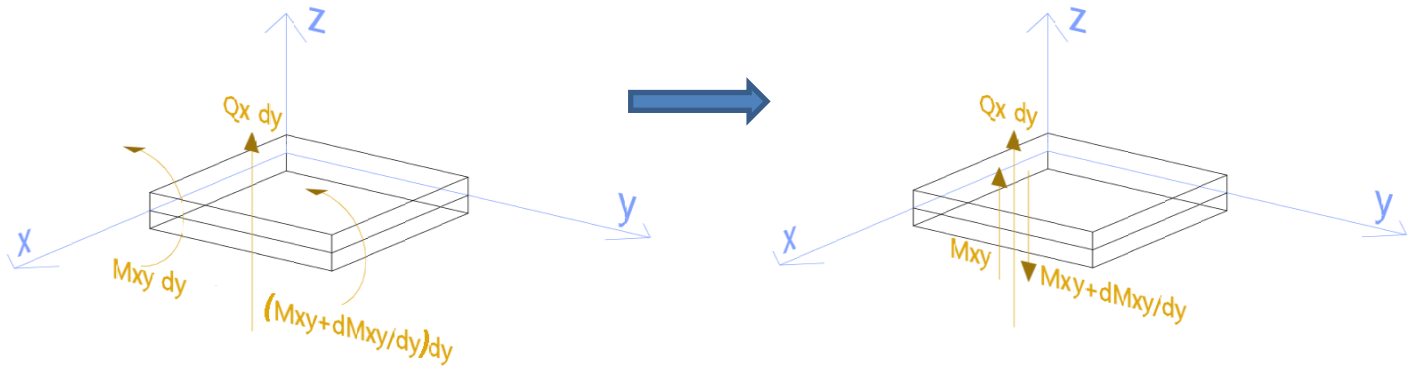
$$\begin{cases} w(x,y) = 0 \\ \dfrac{\partial^2 w(x,y)}{\partial x^2} = 0 \end{cases} \quad \text{Para } x = a; \quad \text{para } \forall y$$

c) *Contorno libre*

El contorno libre se caracteriza por tener momento perpendicular Mx nulo y fuerzas verticales nulas.

Este caso es más complejo que los otros dos anteriores, puesto que primero hay que calcular la expresión de las fuerzas verticales en el contorno.

Tomando un diferencial de placa podemos sumar las fuerzas verticales en el contorno; se puede ver que, en cada punto del contorno, el momento torsor equivale a un par de fuerzas verticales con resultante *0* colocadas a una distancia *dy/2*, por lo que en un diferencial de placa tenemos una de estas fuerzas por cada momento torsor. En resumen, obtenemos lo siguiente:



I.12: *Fuerzas verticales en el contorno x=a. Interpretación del papel del momento torsor en ellas*

$$\sum F_v = Qx \cdot dy \cdot + Mxy \cdot dy - \left(Mxy + \frac{\partial Mxy}{\partial y}\right) dy = \left(Qx - \frac{\partial Mxy}{\partial y}\right) dy \qquad (I.36)$$

Se definen entonces los siguientes esfuerzos cortantes:

$$Qx' = -\frac{\partial Mxy}{\partial y} = -D \cdot (1 - \nu) \cdot \frac{\partial^2 w(x,y)}{\partial x\, \partial y^2} \qquad (I.37.a)$$

$$Vx = Qx + Qx' = Qx - \frac{\partial Mxy}{\partial y} = -D \cdot \left(\frac{\partial^3 w(x,y)}{\partial x^3} + (2 - \nu) \cdot \frac{\partial^3 w(x,y)}{\partial x\, \partial y^2}\right) \qquad (I.37.b)$$

Y las condiciones de contorno que hay que imponer son las siguientes:

$$\begin{cases} Mx(x,y) = 0 \\ Vx(x,y) = 0 \end{cases} \quad \text{Para } x = a; \quad \text{para } \forall y \qquad (I.38)$$

2. RESOLUCIÓN POR MÉTODOS DIRECTOS

*Ver **fuentes [02]-[03]-[06]***

2.1. MÉTODOS DE RESOLUCIÓN: VISIÓN GLOBAL

Existen numerosos métodos para calcular las deformaciones y las tensiones a las que se ve sometida una placa.

Se pueden agrupar en cuatro grandes grupos:

I.13: *Recopilatorio de los métodos de resolución de placas más usados*

Los **métodos directos** buscan una expresión analítica de la flecha a partir de la ecuación *I.31*. Por tanto, sólo sirven para calcular placas con las características mencionadas en el punto 1 de éste capítulo. Básicamente existen dos: el ***método de Navier*** y el ***método de Levy***. Ambos encuentran la función expresada en términos de series de Fourier. Son el objeto de este trabajo, por lo que se verán con gran detalle en el próximo punto. Este tipo de métodos encuentran la solución exacta; sin embargo esta solución puede ser muy compleja y encontrarla puede suponer un gran coste computacional.

El segundo grupo de métodos buscan una función aproximada a partir de **conceptos energéticos**.

Ya sea mediante el *teorema de los trabajos virtuales* (en el que el trabajo realizado por los esfuerzos internos debe ser igual al de las cargas aplicadas) o aplicando la idea de la minimización de la energía, como los métodos de *Ritz* o de *Galerkin*; este tipo de métodos tienen un coste computacional parecido a los métodos directos, mientras que la solución muchas veces no presenta tanta precisión. Es por eso que forman un conjunto de métodos no tan interesante como los demás.

Los **métodos aproximados** se caracterizan por dar una solución aproximada con un coste computacional realmente bajo. Esta solución puede ser suficiente para ciertos proyectos o anteproyectos. El *método de Grashof* se basa en analizar dos bigas perpendiculares en las que las flechas en su punto de intersección se imponen iguales. El *método de Marcus* aplica un coeficiente reductor que tiene en cuenta la influencia favorable del momento torsor. Por último, el *método de Brunner* aplica el método de Cross para pórticos en placas.

Los **métodos numéricos** basan su cálculo en una malla predefinida. El *método de las diferencias finitas* usa una malla regular, mientras que el *método de los elementos finitos* usa mallas mucho más complejas, que pueden adaptarse mejor a las geometrías inusuales. La bondad de los resultados depende de la bondad de la malla. Estos métodos (en especial el MEF) son interesantes ya que pueden calcular placas con geometrías especiales. Por último, el *método de la banda finita* combina los métodos numéricos con los directos.

Por su precisión y simplicidad, en este trabajo se van a estudiar y desarrollar tan sólo los **métodos directos**; esto es, *Navier* y *Levy*. Este estudio se puede visualizar en los apartados siguientes.

*Ver **fuentes [01]-[08]***

2.2. MÉTODO DE NAVIER

El método de Navier es capaz de resolver placas rectangulares que se encuentren simplemente apoyadas en sus 4 contornos. Esto es, que cumpla la ecuación *(I.31)* con las condiciones de contorno *(I.35)* aplicadas en los cuatro contornos.

*Ver **fuentes [05],[12]-[14]***

La idea básica es escribir la función carga *q(x,y)* y la función flecha *w(x,y)* en forma de suma de **series de senos de Fourier** en las dos direcciones *x* e *y*, y aplicar la ecuación *(I.31)* y las condiciones de contorno *(I.35)* a estas nuevas expresiones.

Se obtienen las siguientes expresiones para la carga:

(I.39)

$$q(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} q_{mn} \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right)$$

Como *q(x,y)* es conocida, conocemos los coeficientes q_{mn}

(I.40)

$$q_{mn} = \frac{4}{ab} \cdot \int_{a_1}^{a_2}\int_{b_1}^{b_2} q(x,y) \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right) \cdot dx \cdot dy$$

Haciendo el mismo procedimiento para la flecha se obtiene:

(I.41)

$$w(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} w_{mn} \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right)$$

Se puede comprobar fácilmente, al tratarse de una composición sinusoidal, que la expresión *(I.41)*, por construcción, cumple automáticamente las condiciones de contorno *(I.35).*

Así pues, solo nos queda imponer la ecuación *(I.31)*, sustituyendo en ella la misma expresión para la flecha *(I.41).* Obtenemos:

(I.42)

$$\Delta\Delta w(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} w_{mn} \cdot \Delta\Delta\left[sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right)\right] =$$

$$= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} w_{mn} \cdot \pi^4 \cdot \left[\frac{m^2}{a^2} + \frac{n^2}{b^2}\right] \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right) =$$

$$= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{q_{mn}}{D} \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right) = \frac{q(x,y)}{D}$$

Por lo que se obtiene la relación entre q_{mn} y w_{mn}:

$$w_{mn} = \frac{1}{D \cdot \pi^4 \cdot \left[\frac{m^2}{a^2} + \frac{n^2}{b^2}\right]} \cdot q_{mn} \qquad (I.43)$$

La ecuación *(I.40)* se inserta en la *(I.43)* para obtener la expresión final de la flecha *(I.44)*:

$$w(x,y) = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} w_{mn} \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right) \qquad (I.44)$$

Con:

$$w_{mn} = \frac{4}{ab} \cdot \frac{1}{D \cdot \pi^4 \cdot \left[\frac{m^2}{a^2} + \frac{n^2}{b^2}\right]} \cdot \int_{a_1}^{a_2} \int_{b_1}^{b_2} q(x,y) \cdot sin\left(\frac{m\pi x}{a}\right) \cdot sin\left(\frac{n\pi y}{b}\right) \cdot dx \cdot dy$$

La ecuación *(I.44)* nos da, finalmente, la expresión analítica de la flecha para cualquier punto (*x*,*y*) donde:

a es la longitud de la placa en dirección *x*

b es la longitud de la placa en dirección *y*

a_1 y a_2 son las coordenadas *x* de aplicación de la carga

b_1 y b_2 son las coordenadas *y* de aplicación de la carga

D es la rigidez de la placa

m y *n* son índices mudos de los dos sumatorios, que van desde 1 hasta ∞. En la realidad estas sumas infinitas nunca van a ser realizadas; se va a elegir un número definido *M* de sumas en *x* y un número *N* en *y*, y se va a admitir cierto error. *M y N* deben ser elegidas por la persona que use la fórmula *(I.44)*. A medida que estos dos coeficientes aumenten también lo va a hacer la precisión de los resultados obtenidos.

A partir de las ecuaciones *(I.22), (I.23), (I.24), (I.29), (I.30)* y *(I.37)* se pueden obtener los esfuerzos que provoca esta flecha.

*Ver **fuentes [01]-[08]***

2.3. MÉTODO DE LEVY

El método de Levy es capaz de resolver placas rectangulares que se encuentren simplemente apoyadas en dos de sus contornos enfrentados. Los otros dos contornos pueden estar apoyados, empotrados, libres... Por lo tanto, se debe cumplir la ecuación *(I.31)* con las condiciones de contorno pertinentes aplicadas en estos contornos.

La idea básica es escribir la flecha *w(x,y)* como la suma de una solución homogénea $w_h(x,y)$ y una particular $w_p(x,y)$. De esta manera, la ecuación *(I.31)* que debe cumplir *w(x,y)* no debe cumplirse para ambas soluciones; únicamente para una de ellas, la solución particular. Las condiciones de contorno se deberán imponer, por el contrario, a la suma de las dos soluciones *w(x,y).*

a) Solución homogénea $w_h(x,y)$

La solución homogénea debe cumplir, entonces, la ecuación *(I.45)*:

(I.45)

$$\Delta\Delta w_h(x,y) = 0$$

y la condición de contorno *(I.35)* únicamente en los bordes simplemente apoyados. Si tomamos éstos contornos como *x=0* y *x=a*, podemos escribir $w_h(x,y)$ descompuesta en senos de Fourier en la dirección *x* únicamente de la siguiente manera:

*Ver **fuentes [05],[12]-[14]***

(I.46)

$$w_h(x,y) = \sum_{m=1}^{\infty} Y_m(y) \cdot sin\left(\frac{m\pi x}{a}\right)$$

De manera análoga al método de Navier, podemos ver que la condición de contorno *(I.35)* se cumple automáticamente, aunque esta vez sólo para *x=0* y *x=a*. El problema ahora es encontrar para cada iteración *m* la función Y_m. Ésta se va a encontrar precisamente imponiendo en *(I.46)* la ecuación *(I.45).*

De hacerlo, se obtiene la siguiente expresión:

(I.47)

$$\sum_{m=1}^{\infty} \left[Y_m(y)^{IV} - 2 \cdot \left(\frac{m\pi}{a}\right)^2 \cdot Y_m(y)^{II} + \left(\frac{m\pi}{a}\right)^4 \cdot Y_m(y) \right] \cdot sin\left(\frac{m\pi x}{a}\right) = 0$$

Que en particular se cumple para cada *x* y para cada *m*, por lo que al final la ecuación que debemos resolver es la *(I.48)*:

$$Y_m(y)^{IV} - 2 \cdot \left(\frac{m\pi}{a}\right)^2 \cdot Y_m(y)^{II} + \left(\frac{m\pi}{a}\right)^4 \cdot Y_m(y) = 0 \qquad (I.48)$$

Esta ecuación tiene una solución analítica bien conocida. Su expresión es la siguiente:

$$Y_m(y) = A_m \cdot cosh(\varphi) + B_m \cdot \varphi \cdot sinh(\varphi) + C_m \cdot sinh(\varphi) + D_m \cdot \varphi \cdot cosh(\varphi) \qquad (I.49)$$

Con

$$\varphi = \frac{m\pi}{a} \cdot (y - \frac{b}{2})$$

A_m, B_m, C_m y D_m son $4 \cdot M$ incógnitas que van a ser encontradas más adelante imponiendo las condiciones de contorno pertinentes a los dos extremos $y=0$ e $y=b$ a la flecha $w(x,y)$ en cada iteración.

b es la longitud de la placa en la dirección y

Por tanto, las ecuaciones *(I.46)* y *(I.49)* nos describen de forma detallada la estructura que tiene la solución homogénea $w_h(x,y)$.

NOTA: *En la mayoría de documentos (**fuentes [01]-[08]**) aparece la ecuación (I.49) expresada en términos de y, y no de y+b/2. Esto es porque se considera el eje y pasando por el centro de la placa. En este documento se ha hablado en todo momento de ejes colocados en una esquina de la placa, y se ha considerado conveniente mantener la misma notación y sistema de referencia[4] en todas las partes que lo componen.*

b) Solución particular $w_p(x,y)$

Vamos ahora a por la solución particular. Sabemos que debe cumplir la ecuación *(I.31)* y que no debe cumplir ninguna condición de contorno en especial.

No parece mala idea, pues, tomar como solución particular una que ya hemos obtenido anteriormente: la solución de Navier *(I.44)*. Precisamente cumple la ecuación *(I.31)* y nos sirve para utilizarla como

[4] Ver figura I.3: *Placa genérica. Espesor, plano medio y situación de los ejes de coordenadas*

solución particular en el método de Levy.

Muchos autores (***fuentes [01-08]***) aprovechan que la solución homogénea de Levy está desarrollada en una sola serie de Fourier en el eje *x* para describir de igual forma la solución particular. La verdad es que no es necesario que ambas estén desarrolladas de la misma forma. Describir la solución particular con una sola serie Fourier no nos permite calcular de una manera fácil flechas cuando se aplican cargas no constantes en *y*. De hecho, la mayoría de autores describen el método de Levy, y por consiguiente su solución particular, para cargas constantes tanto en *x* como en *y*.

En este trabajo se ha querido desarrollar un software capaz de resolver el mayor número de cargas posibles. Es por eso que se ha desarrollado el **método de Levy con doble desarrollo** (en *x* y en *y*) **para su solución particular**; para abarcar el caso más general. Si es verdad que el doble desarrollo pierde en velocidad de convergencia y aumenta el tiempo de cálculo, por otro lado nos permite calcular flechas con cualquier carga aplicada que se pueda escribir en términos de doble serie de senos de Fourier.

Es por eso que la solución particular que se ha elegido para el método de Levy ha sido la misma ecuación *(I.44)* del método de Navier.

Ya hemos obtenido la solución homogénea $w_h(x,y)$ *(I.46)* y *(I.49)* y la solución particular $w_p(x,y)$ *(44)*. Sólo queda sumarlas para obtener la solución general *w(x,y)* y aplicarle las condiciones de contorno para *y=0* e *y=b* pertinentes. Si las sumamos y sacamos factor común del seno que desarrolla la flecha en dirección *x* obtenemos:

$$w(x,y)=\sum_{m=1}^{\infty}\left[\begin{array}{c} A_m\cdot cosh(\varphi)+B_m\cdot\varphi\cdot sinh(\varphi)+C_m\cdot sinh(\varphi)+\ D_m\cdot\varphi\cdot cosh(\varphi)+ \\ \frac{4}{abD\pi^4}\cdot\sum_{n=1}^{\infty}\frac{1}{\left[\frac{m^2}{a^2}+\frac{n^2}{b^2}\right]}\cdot\int_{a_1}^{a_2}\int_{b_1}^{b_2}q(x,y)\,sin\left(\frac{m\pi x}{a}\right)sin\left(\frac{n\pi y}{b}\right)dxdy\cdot sin\left(\frac{n\pi y}{b}\right) \end{array}\right]\cdot sin\left(\frac{m\pi x}{a}\right)$$

(I.50)

Con

$$\varphi=\frac{m\pi}{a}\cdot(y-\frac{b}{2})$$

Esta compleja expresión *(I.50)* es el resultado de sumar la homogénea y la particular de Levy. Nuevamente, las series deberán ser truncadas hasta *M* y *N* términos respectivamente.

A partir de las ecuaciones *(I.22), (I.23), (I.24), (I.29), (I.30), y (I.37)* se pueden obtener los esfuerzos que provoca esta flecha. Derivando la expresión *(I.50)* respecto a *x* e *y* se pueden obtener también los giros provocados por esta flecha en cada dirección.

Por tanto, somos capaces de imponer las condiciones de contorno descritas en el punto *1.10* en los contornos *y=0* e *y=b* (dos para cada contorno y cada iteración).

Estas condiciones de contorno se van a tener que cumplir para toda *x.* Nótese que, sean cual sean las condiciones de contorno a aplicar, siempre que se derive la flecha *w(x,y)* respecto a *x* se va a hacer dos veces: es decir, en las condiciones de contorno aparecerá o bien la flecha sin derivar o la derivada segunda de la flecha respecto a *x* (también aparecerán derivadas respecto a *y*, pero en este punto no nos centramos en ellas). Es decir, que las ecuaciones que formen el sistema de 4 ecuaciones con 4 incógnitas a encontrar para cada *m* van a estar formadas por una ecuación extensa que depende de *y* multiplicada por el seno de $\boldsymbol{m\pi x/a}$.

Como el sistema de ecuaciones se debe cumplir para toda *x*, este seno puede ser eliminado de las 4 ecuaciones que lo forman. De esta manera, obtendremos un sistema que sólo depende de *y.* Evaluando cada ecuación en su coordenada correspondiente (*y=0* o *y=b*) obtendremos finalmente un sistema de 4 ecuaciones numéricas con 4 incógnitas (A_m, B_m, C_m y D_m) para cada iteración *m*.

Sistema *m* de ecuaciones

$$\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} A_m \\ B_m \\ C_m \\ D_m \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \begin{matrix} \\ \\ \leftrightarrows \text{ evaluadas en } y=0 \\ \\ \leftrightarrows \text{ evaluadas en } y=b \end{matrix}$$

I.14: *Sistema de ecuaciones a resolver para cada iteración m*

Este sistema se puede resolver sin ningún tipo de problema matemático. Entonces, para cada iteración *m* encontraremos los coeficientes A_m, B_m, C_m y D_m. Con ellos y con la ecuación *(I.50)* **hemos hallado la flecha *w(x,y)***, y por consiguiente podremos encontrar los giros y esfuerzos asociados a esta flecha.

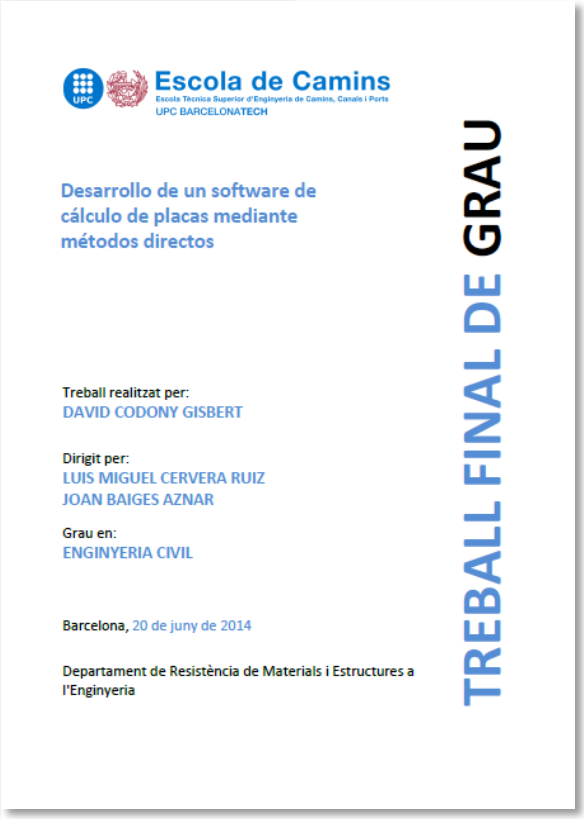
CAPÍTULO II

PLACALCULATOR 1.0 – MANUAL DE USUARIO

1. ANTES DE EMPEZAR

Placalculator 1.0 es un software de cálculo de placas basado en la resolución de placas delgadas mediante métodos directos.

Forma parte del trabajo de fin de grado en ingeniería civil "*Desarrollo de un software de cálculo de placas mediante métodos directos"* realizado por *David Codony Gisbert* en 2014.


Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Desarrollo de un software de cálculo de placas mediante métodos directos

TREBALL FINAL DE GRAU

Treball realitzat per:
DAVID CODONY GISBERT

Dirigit per:
LUIS MIGUEL CERVERA RUIZ
JOAN BAIGES AZNAR

Grau en:
ENGINYERIA CIVIL

Barcelona, 20 de juny de 2014

Departament de Resistència de Materials i Estructures a l'Enginyeria

II.01: *Portada del trabajo de fin de grado*

Dicho software se ha diseñado mediante el programa *Matlab* de *MathWorks, Inc.*

Se le ha querido dar un enfoque docente, de manera que sea una herramienta útil para los alumnos estudiantes de resistencia de materiales.

Incorpora el cálculo de placas delgadas rectangulares sometidas a cargas puntuales, liniales o repartidas de manera rectangular. Dicho cálculo se puede realizar mediante dos métodos directos de cálculo (Navier y Levy).

Se le ha intentado proporcionar un *layout* cómodo y práctico para el usuario, de manera que la navegación por él, la introducción de datos de partida y la interpretación de los resultados finales sean tareas fáciles de realizar, aun así sin leer este manual. La intención es que el usuario sepa en todo momento donde está, que debe hacer y que no se pierda por dentro del programa.

Así pues, como se podrá ver más adelante, *Placalculator 1.0* tiene una interfaz simple, pero a la vez seria y rigurosa.

El programa consta de las opciones de *guardado* y *cargado* de problemas, hecho que facilita al usuario final la visualización de ejemplos ya resueltos o el guardado de los mismos.

El programa, así como este manual, está dividido en dos secciones principales: el **preproceso** y el **postproceso**. En la primera el usuario introduce los datos necesarios (o los carga) para realizar el cálculo; la segunda muestra los resultados una vez se ha procesado la información del preproceso y se han hecho los cálculos pertinentes.

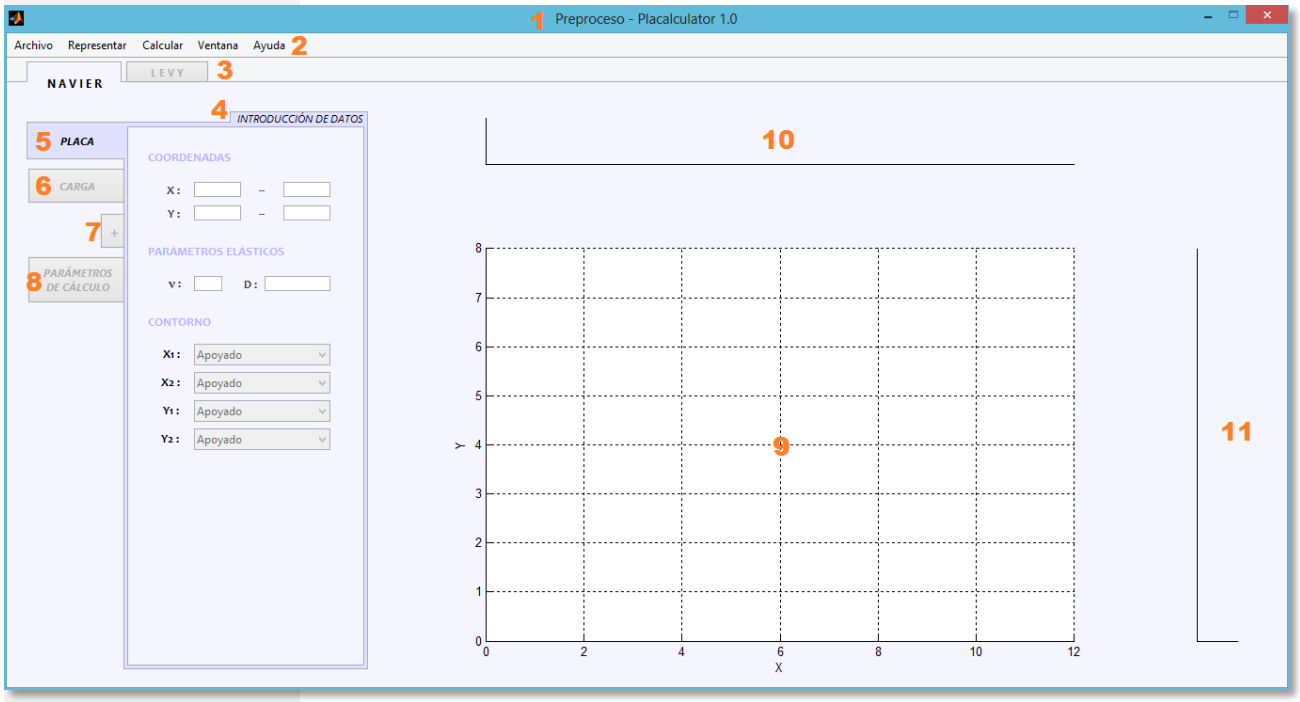
En este manual **no se va a hablar de unidades**: el usuario debe ser consistente con él mismo y saber que los resultados se van a dar **en las mismas unidades** que haya elegido él en el preproceso.

El **separador decimal** en todo momento va a ser el **punto** (.)

2. PREPROCESO

2.1. VISIÓN GLOBAL

El preproceso es la pantalla inicial que se muestra al iniciar el programa:



II.02: *Pantalla de preproceso. Partes que lo componen numeradas de naranja del 1 al 11*

Vamos a hacer un breve resumen de los componentes de la pantalla. Más adelante se va a explicar detalladamente cada uno de ellos.

Se puede ver que aparece la palabra "*preproceso*" en el **título** 1 de la ventana.

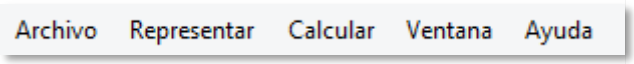
Arriba de la ventana tenemos la **barra de menú** 2 con sus diferentes pestañas. La más relevante quizás es la pestaña de cálculo.

Justo debajo tenemos la **barra de selección de método de cálculo** 3 (*Navier* o *Levy*).

Las partes 9 *(ejes x-y)* , 10 *(ejes x-z)* y 11 *(ejes y-z)* forman la **representación gráfica** del problema.

La sección 4 se llama "*introducción de datos*". Es quizás la parte fundamental del preproceso. Contiene las 3 pestañas que el usuario debe rellenar para poder realizar los cálculos: la pestaña que caracteriza la **placa** 5, la que define la **carga** 6 y la que determina los **parámetros de cálculo** 8. También consta de un botón para **añadir cargas** 7, en el caso que haya más de una.

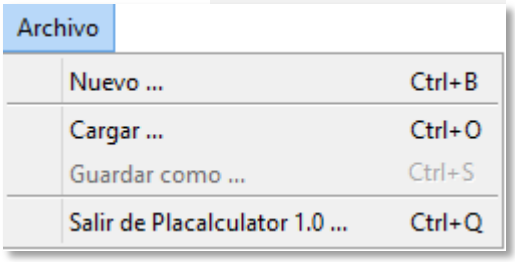
2.2. BARRA DE MENÚ



II.03: *Barra de menú*

Nos muestra 5 pestañas: *"archivo", "representar", "calcular", "ventana"* y *"ayuda"*.

a) *Archivo*



II.04: *Barra de menú > Archivo*

Nos permite realizar las operaciones básicas de cualquier software.

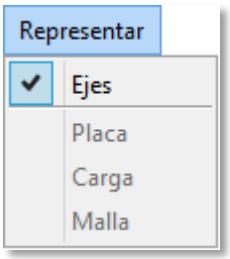
Nuevo... nos permite abrir un nuevo documento en blanco. Se pierden los datos que no hayan sido guardados.

Cargar... nos permite cargar ejercicios ya resueltos y guardados, como por ejemplo los ejercicios de validación del capítulo IV.

Guardar como... nos permite guardar únicamente **ejercicios ya resueltos**. Por tanto, es una opción desactivada antes de hacer ningún cálculo.

Salir de Placalculator 1.0... cierra el programa.

b) *Representar*



II.05: *Barra de menú > Representar*

Nos indica qué se está representando en las zonas **9** **10** y **11**. Esta pestaña **no va a modificar** en ningún momento la representación gráfica. Únicamente sirve para saber qué es lo que el programa nos está dibujando. En ella salen los principales componentes de la representación gráfica: Los ejes coordenados, la placa, la malla y la(s) carga(s) (a medida que añadamos cargas van a ir apareciendo en esta pestaña).

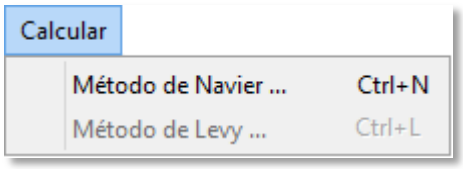
Los componentes que aparecen pueden estar **iluminados con un tic (∨)** o **apagados**.

Si aparecen iluminados, significa que los datos correspondientes han sido rellenados correctamente en el apartado **4**. Por tanto van a ser dibujados (los ejes coordenados siempre están iluminados y dibujados). Si no es así, significa que los datos no se han rellenado

correctamente o que aún no se han rellenado. De manera que no se van a dibujar.

Nótese, por tanto, que el cálculo no se va a poder realizar hasta que todos los componentes de esta pestaña aparezcan iluminados.

c) Calcular



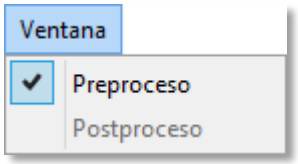
II.06: *Barra de menú > Calcular*

Esta es la pestaña más importante del menú. Nos permite iniciar el cálculo tras rellenar todos los datos necesarios. Siempre habrá una de las dos opciones iluminada, Método de Navier... o Método de Levy..., en función de la elección que hayamos hecho en **3**. La otra opción estará siempre desactivada.

Si algún dato no está bien rellenado en **4**, nos saldrá un mensaje de error avisándonos del problema y no vamos a poder calcular nada. Por el contrario, si no hay ningún problema, nos saldrá una pregunta de confirmación que nos permitirá calcular nuestra placa según el método indicado.

Al acabar el cálculo podremos abrir la ventana de postproceso.

d) Ventana

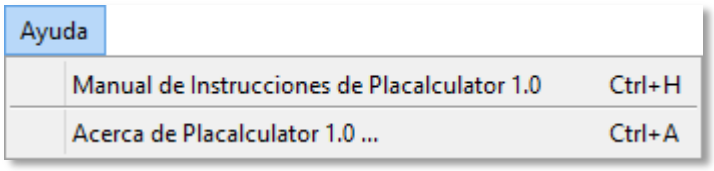


II.07: *Barra de menú > Ventana*

Esta pestaña nos indica que estamos en el preproceso y nos permite cambiar de ventana para ver el postproceso.

Únicamente va a ser posible ver el postproceso si el cálculo ya se ha realizado. Por lo que siempre que abramos el programa de nuevo o abramos una hoja en blanco, esta opción va a estar desactivada.

e) Ayuda



II.08: *Barra de menú > Ayuda*

Muestra la ayuda del programa.

Manual de instrucciones de Placalculator 1.0 abre este mismo documento.

Acerca de Placalculator 1.0... abre una ventana informativa que explica de manera resumida el objeto de este software, el autor y sus tutores, la institución donde se realizó, etc.

2.3. BARRA DE SELECCIÓN DE MÉTODO DE CÁLCULO

II.09: *Barra de selección del método de cálculo*

Nos da la opción a elegir entre plantear un problema que va a ser resuelto por *Navier,* o por *Levy*. Es lo primero que hay que elegir si queremos rellenar el apartado **4** en vez de cargar un ejercicio ya resuelto, puesto que ésta elección va a modificar los parámetros de **4**.

a) *Navier*

Como ya hemos dicho antes, nos permite entrar los datos necesarios para realizar el cálculo por Navier.

Por tanto bloquea las condiciones de contorno en el apartado **5** para los 4 apoyos de la placa y los deja marcados como apoyados.

En la *barra de menú* > *calcular* se ilumina el apartado Método de Navier… .

b) *Levy*

Nos modifica **4** para que se pueda aplicar posteriormente un cálculo por el método de Levy.

Es decir, bloquea las condiciones de contorno para $x=0$ y $x=a$ y los marca como apoyados. Por otro lado desbloquea las condiciones en y, que ahora pueden ser definidas por el usuario.

En la *barra de menú* > *calcular* se ilumina el apartado Método de Levy… .

2.4. PLACA

Esta pestaña forma parte de **4** (*introducción de datos*).

En ella podremos decirle al programa todas las características de nuestra placa.

Consta de tres partes: *"coordenadas", "parámetros elásticos"* y *"contorno"*.

a) *Coordenadas*

II.10: *Introducción de datos > placa*

En este apartado debemos introducir las coordenadas de inicio y de fin de la placa, tanto en *x* como en *y*. Una vez introducidas, la placa aparece dibujada en **9**.

Es importante que *x1* sea estrictamente menor que *x2* y que *y1* sea estrictamente menor que *y2*. De lo contrario, no se va a dibujar ninguna placa y no se van a poder realizar los cálculos.

Las unidades, como ya se ha comentado en la introducción de este capítulo, deben ser elegidas por el usuario, pero deben ser las mismas en todo el preproceso. En este apartado concreto deben ser unidades de longitud [L].

b) *Parámetros Elásticos*

Este apartado pide los parámetros que caracterizan a la placa. Concretamente el parámetro *nu* [-] y la rigidez de la placa [F·L].

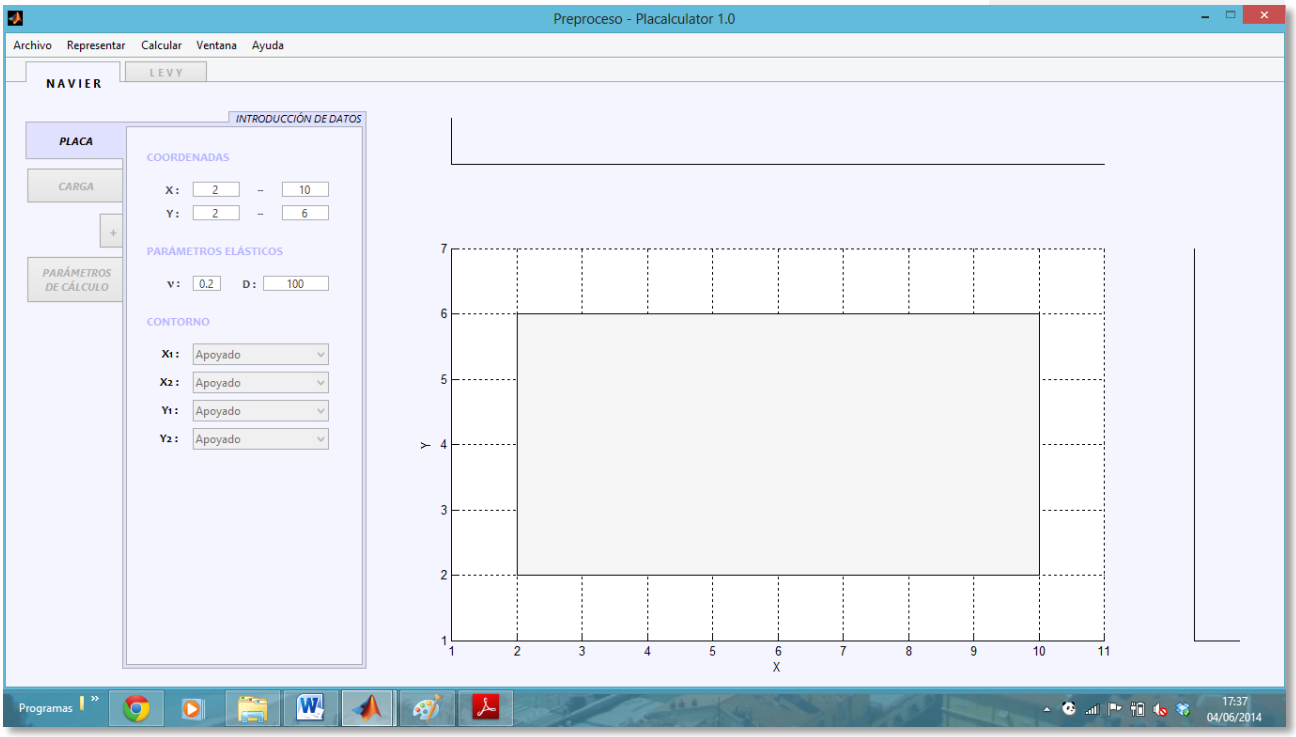
El programa no pide el espesor de la placa porque ya está incluido en la rigidez. Si no se conoce la rigidez de la placa pero sí su espesor, se puede proceder a calcular la rigidez de la placa mediante la ecuación *(I.25)* del capítulo I.

La rigidez debe entrarse como un número real. No se puede escribir como una operación matemática.

c) *Contorno*

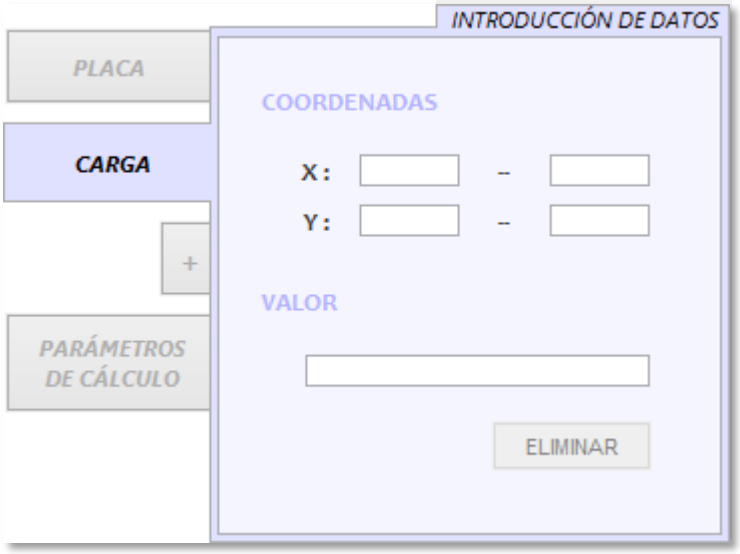
Nos permite elegir las condiciones de contorno de nuestra placa. En el caso de Navier, estarán todas fijadas como Apoyado. Sin embargo, en Levy podremos elegir las características de los contornos *y1* e *y2*. Podremos elegir entre Apoyado, Empotrado y Libre.

x1 y *x2* van a estar fijados en cualquier caso.



11: *Ejemplo rellenado correctamente. Representación gráfica de la placa en la parte derecha de la ventana*

2.5. CARGA



II.12: *Introducción de datos > Carga*

En esta pestaña se definen las características de la carga aplicada a la placa de manera puntual, lineal o repartida en un rectángulo.

Antes de explicar esta sección me gustaría hacer mención a la pestaña que aparece justo debajo de ésta: la pestaña *(+)*.

Esta pestaña sirve para añadir otra carga, en el caso de que ésta existiese.

Se pueden añadir hasta un máximo de 5 cargas. Para cada una de ellas va a aparecer una pantalla como la que vemos en la figura ***II.12***. Todas las cargas que se añadan en el preproceso deben estar bien rellenadas. Si queremos eliminar una carga, no basta con dejar en blanco sus datos. Debemos eliminarla pulsando el botón ELIMINAR.

Sólo se puede eliminar la última carga añadida. Hay que tener en cuenta que si sólo tenemos una carga, no podemos borrarla.

a) *Coordenadas*

En este apartado debemos introducir las coordenadas [L] de inicio y de fin de la región donde queremos que actúe nuestra carga, tanto en *x* como en *y*. Una vez introducidas, ésta región aparece dibujada en **9**. Para poder lanzar el cálculo, la carga debe estar aplicada toda dentro de la placa.

- *Carga repartida: x1* y *x2* deben ser diferentes. Ídem para *y1* e *y2*.

- *Carga lineal:* Basta con definir *x1=x2=x** (o *y1=y2=y**). Entonces el software considerará que entre *y1* e *y2* (o *x1* y *x2)* se aplica una carga lineal en *x=x** (o *y=y**).

- *Carga puntual:* Basta con definir *x1=x2=x** a la vez que *y1=y2=y**. El software considerará que se aplica una carga puntual en el punto (*x*,y*)*.

b) *Valor*

Se debe escribir la expresión de la carga *q(x,y)* (positiva en dirección ascendente del eje *Z*. Tanto para *Navier* como para *Levy*, se admiten cargas con dependencia de *x* y de *y*. Las variables *x* e *y* pueden escribirse en mayúsculas o minúsculas indistintamente. No se pueden utilizar otras letras que no sean *x* o *y* (aparte de las que designen una operación matemática, como exp( ), log( ), etc).

Se deben usar los símbolos típicos para escribir operaciones matemáticas (+ para la suma, - para la resta, * para el producto, / para la división, ^ para la potencia, etc).

Asimismo, también se puede escribir un valor numérico constante que no dependa de *x* o de *y*.

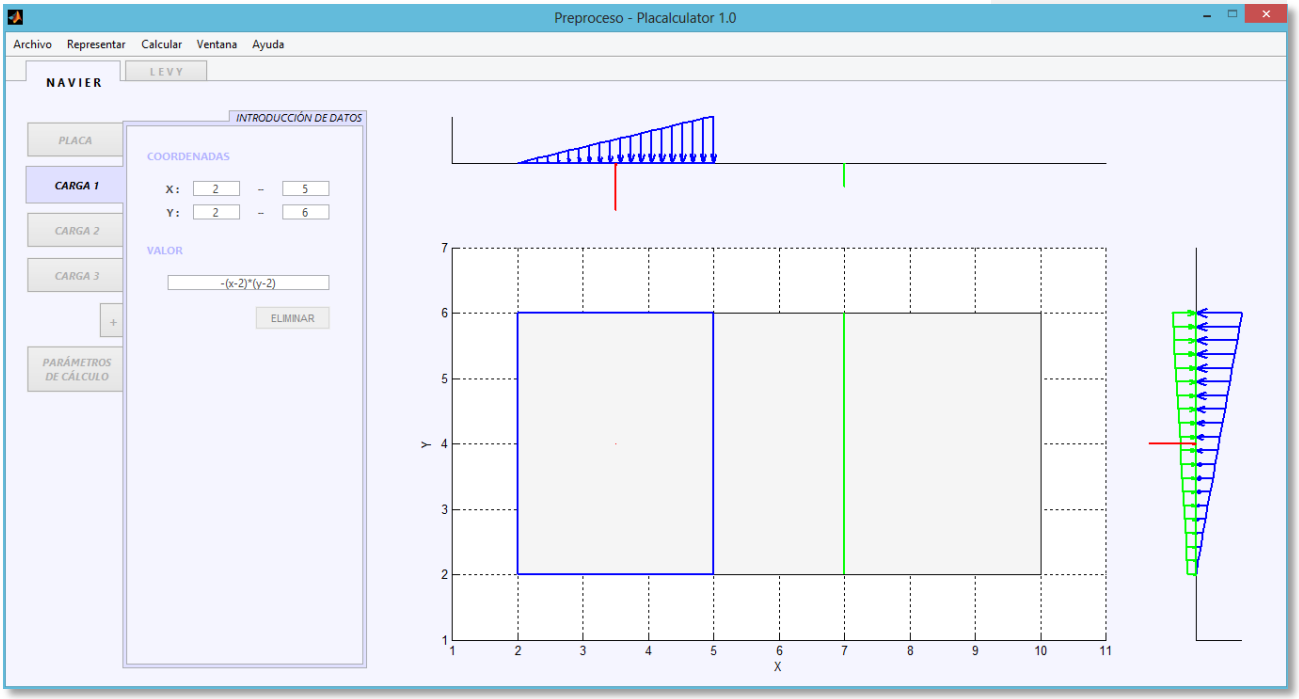
- *Carga repartida: q(x,y)* debe tener unidades de [F/L^2].

- *Carga lineal: q(x,y)* debe tener **unidades de [F/L].**

- *Carga puntual: q(x,y)* debe tener **unidades de [F].**

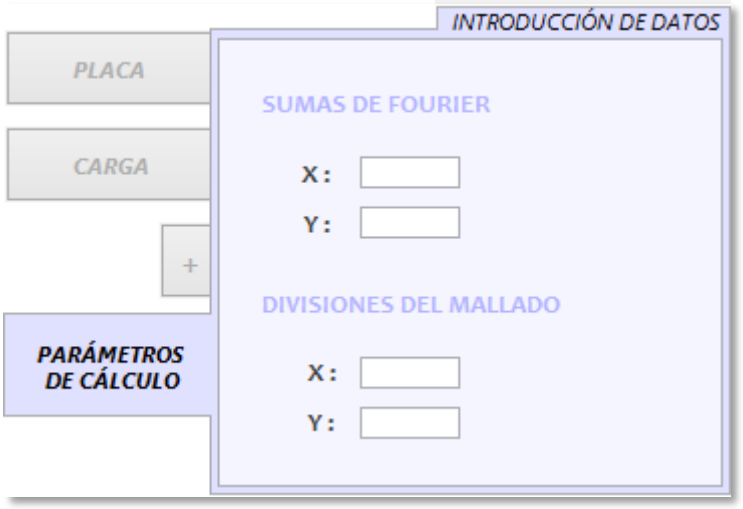
Una vez introducida, la carga debería aparecer representada en **10** y **11** (representación de los ejes *XZ* e *YZ*). De no ser así, algún dato se ha introducido mal y el cálculo no va a poder ser realizado.

NOTA: *Cuantas más cargas definamos, más largo se hará el proceso de cálculo.*



II.13: *Ejemplo rellenado correctamente. Representación gráfica de las cargas en la parte derecha de la ventana*

2.6. PARÁMETROS DE CÁLCULO



II.14: *Introducción de datos > Parámetros de Cálculo*

En esta pestaña se definen las características del algoritmo de cálculo.

Tanto para *Navier* como para *Levy* vamos a necesitar determinar el número de sumas de Fourier y vamos a tener que diseñar una malla.

a) Sumas de Fourier

Tal y como se explica en el capítulo I, los sumatorios que aparecen en las ecuaciones *(I.44)* y *(I.50)* de ese capítulo van de *1* a *infinito*. Para poder realizar ese cálculo se necesita truncar las series hasta un número finito de sumas. Para la coordenada *x* se definen *M* sumas y para la coordenada *y* se definen *N*.

Estos parámetros son los que hay que escribir en este apartado.

Son números enteros (como mínimo 1) que debe elegir el usuario del software.

El valor de estos coeficientes **va a influir directamente en la precisión** de los resultados. Cuanto mayores sean, más precisos van a ser los resultados obtenidos.

NOTA: *Cuantas más sumas de Fourier elijamos, más se va a alargar el proceso de cálculo.*

NOTA: *El punto 2 del capítulo 4 recoge un análisis de convergencia por aumento del número de sumas consideradas en x y en y, hecho sobre un ejercicio a modo de ejemplo. En él se puede ver que el error relativo de la solución va disminuyendo a medida que aumentamos el número de sumas. Y por tanto la solución va obteniendo un grado de precisión cada vez más elevado.*

b) Divisiones del mallado

Para el cálculo en sí, la malla es totalmente innecesaria. En la teoría explicada en el capítulo I del trabajo de fin de grado no se habla de métodos numéricos ni de diferencias finitas. Tan sólo se habla de métodos de resolución directos, que son el objeto del proyecto.

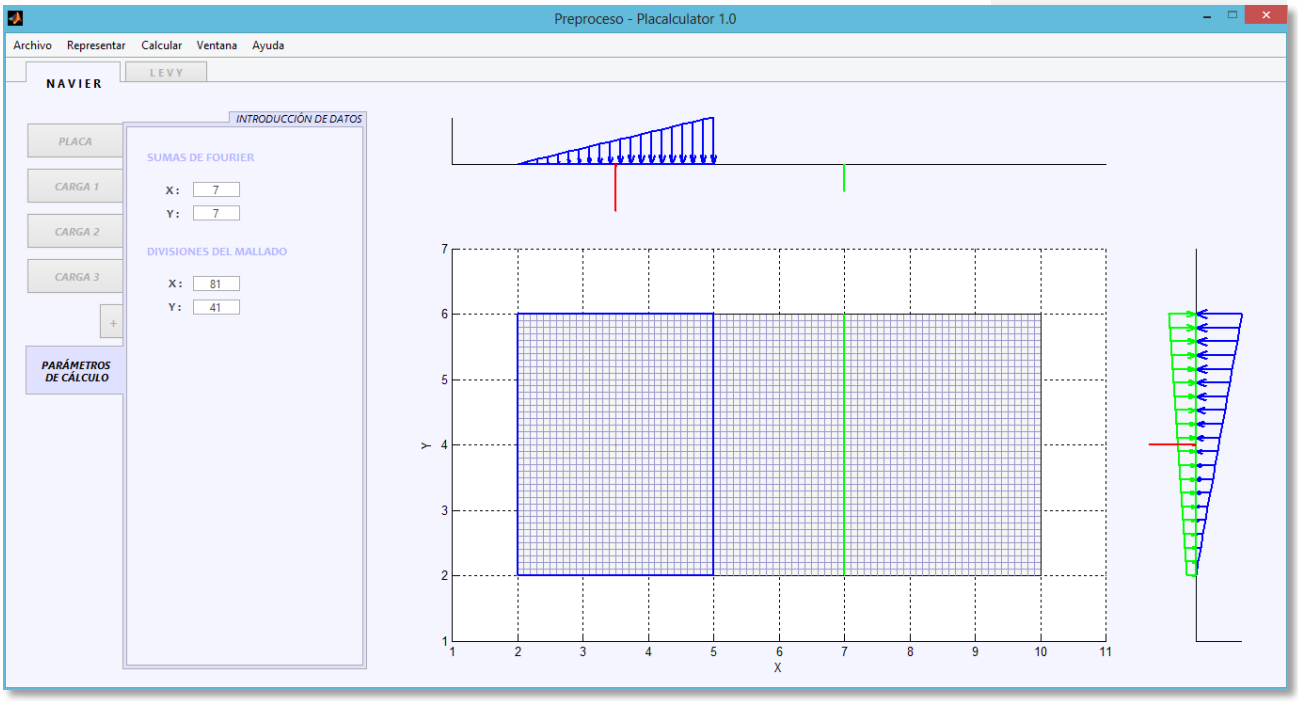
La malla es útil, sin embargo, **a la hora de dibujar** la solución final. Y es que no es tan importante la función solución *w(x,y)* como su **representación gráfica**. En el postproceso se van a usar los puntos definidos por esta malla para evaluar las funciones solución. Estos valores se van a unir entre sí con líneas rectas. Si buscamos una representación gráfica de la solución bonita y suavizada, usaremos una malla extensa. Si no, no es necesario.

En ningún caso la malla va a marcar la precisión de los resultados.

El número de divisiones del mallado rectangular deberá ser un número real, mayor o igual que *3* en cualquier caso.

Si se entran bien los datos del mallado, éste va a aparecer representado en **9**.

NOTA: *Cuantas más divisiones del mallado elijamos, más se va a alargar el proceso de cálculo.*

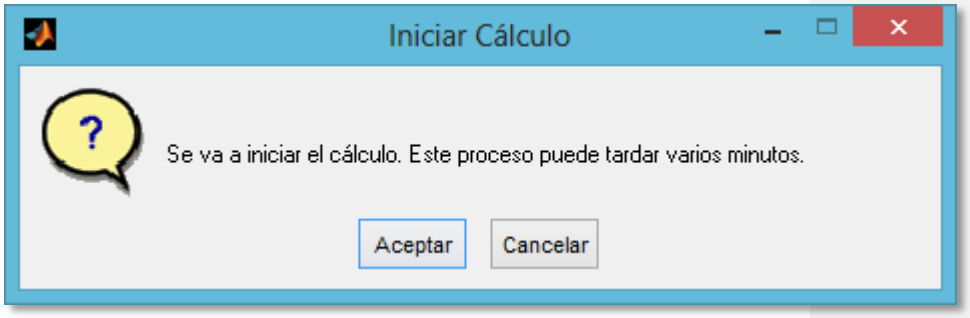


II.15: *Ejemplo rellenado correctamente. Representación gráfica de la malla en la parte derecha de la ventana*

2.7. LANZAMIENTO DEL CÁLCULO

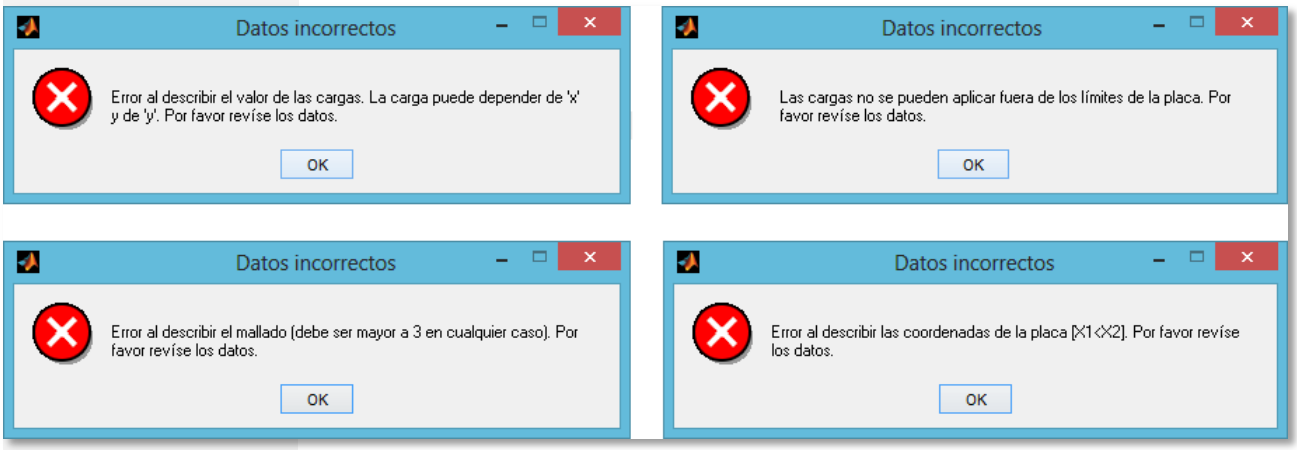
Una vez hayamos entrado todos los datos correctamente, podremos lanzar el cálculo. Para ello, debemos dirigirnos al botón calcular dentro de la barra de menú **2** y seleccionar Método de Navier... o Método de Levy... según sea nuestro caso. Como ya hemos explicado en el apartado *2.2c*, sólo una de las dos opciones podrá ser pulsada. Así que no hay posibilidad de confusión en este aspecto.

Al clicar en la opción correspondiente debería abrirse un mensaje de aviso como el de la figura ***II.16***.



II.16: *Mensaje de confirmación para el lanzamiento del cálculo.*

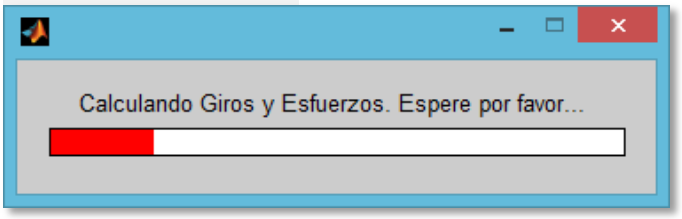
Si no se han seguido debidamente todos los pasos y cumplido las restricciones que marca este manual, no aparecerá esta pantalla, sino un mensaje de error como los de la figura ***II.17***.



II.17: *Recopilatorio de algunos de los mensajes de error posibles al lanzar el cálculo*

De aparecer un mensaje como los representados arriba, los datos introducidos deben ser revisados, puesto que no se han introducido correctamente.

En la mayoría de los casos el mismo software te dice a qué se debe el error. Simplemente hay que darle a OK para cerrar la ventana y, tras solucionar el problema, volver a lanzar el cálculo.



II. 18: *Barra de progreso*

Una vez nos aparezca un mensaje como el de la figura ***II.16*** podemos lanzar el cálculo. Al darle al botón aceptar saldrá una figura con una barra de progreso como la de la figura ***II.18***, que se irá llenando a medida que pase el tiempo.

a) Tiempo de cálculo

El **tiempo de cálculo** depende de diversos factores, como son el tipo de procesador del ordenador, el tipo y número de actividades que el pc esté desarrollando en paralelo… Es por eso que no se ha estudiado con detalle esta materia.

Sin embargo, se puede decir que el cálculo se separa en dos mitades bastante claras: todo el proceso desde que empieza el cálculo hasta que encuentra las funciones solución de flecha, giros, momentos… es la primera. La segunda empieza en este punto, cuando el programa tiene que evaluar las funciones solución en los puntos de la malla.

Pues bien, dejando la informática a un lado, básicamente hay **tres tipos de factores**; los que influyen en la primera mitad, los que influyen en la segunda y los que influyen en las dos.

El tipo de método de resolución es del primer tipo. *Levy* va a tardar siempre más que *Navier* a calcular la misma placa, puesto que los cálculos que realiza el segundo se corresponden con la solución particular del primero. *Levy* deberá resolver también el sistema de ecuaciones de la solución homogénea.

Otro factor que se encuentra dentro del primer grupo es el número de cargas y la tipología. Según aumente el número de cargas y según la complejidad de la expresión *q(x,y)*, el software tardará más en calcular los coeficientes w_{mn}.

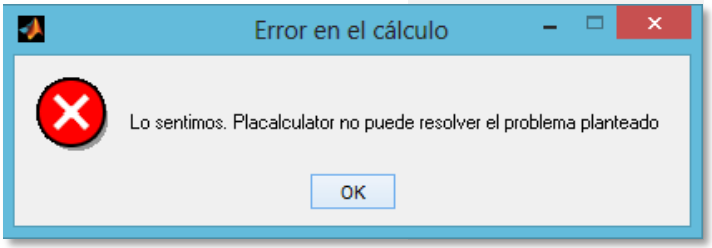
La malla forma parte del segundo grupo. Sólo afecta al tiempo de cálculo una vez hay que evaluar en ella las funciones encontradas. Es por eso que si nuestro objetivo tan solo es buscar precisión en un cierto punto, y no la representación gráfica, no necesitamos una malla extensa.

Los coeficientes *M* y *N* forman parte del tercer grupo. Como mayores sean, mayores serán también las series, y por tanto más largas serán las expresiones finales de flechas giros y esfuerzos. Pero no sólo eso, sino que también se va a incrementar el tiempo de evaluación de éstas en los nodos de la malla (puesto que *x* e *y* aparecen más veces en una misma expresión).

b) *Cálculo interrumpido*

Tras este breve comentario sobre coste computacional, también hay que remarcar que llegados a este punto **puede ser que el cálculo falle** (aunque se haya podido lanzar el mismo). Uno de los motivos puede ser que quizás no sea posible descomponer la carga (aunque esté bien escrita) en series de Fourier de manera analítica (por ejemplo, la función $q(x)=exp(x^2)$, que no tiene primitiva).

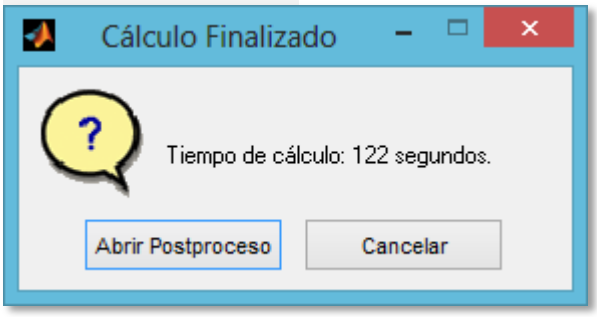
En ese caso, la barra de progreso va a desaparecer y vamos a ver un mensaje de error como el de la figura ***II.19***.



II. 19: *Mensaje de error ocurrido durante el proceso de cálculo*

El software está diseñado para que no se quede trabado tras cualquier error que pueda surgir en el proceso de cálculo. A priori, si eso sucediese siempre se va a detener el cálculo automáticamente y se va a lanzar un mensaje como el mostrado. Si la barra no avanza, seguramente se debe a que el cálculo es complejo y largo de hacer, no a que el programa no funcione correctamente. Sólo queda esperar a que resuelva el problema o cerrar el programa de manera brusca, a través del administrador de tareas (o similar) del pc del usuario.

c) *Éxito en el cálculo*



II.20: *Cálculo finalizado*

Si finalmente no hay ningún tipo de interrupción en el cálculo, al finalizar se va a mostrar una ventana como la de la figura ***II.20***.

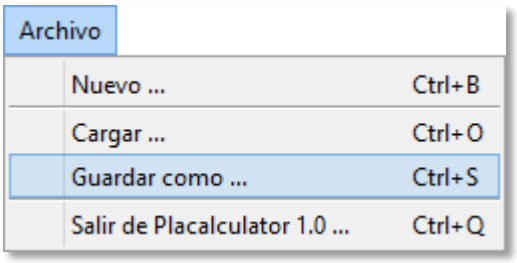
En ella podemos ver el tiempo [*seg*] que ha tardado el software en calcular nuestro problema. También se nos da la opción de ir al postproceso para ver los resultados.

Si queremos simplemente volver al preproceso, podemos darle al botón cancelar. Si el usuario elije esta opción, se va a cerrar la ventana de la figura ***II.20***. Podemos ver que en la barra del título 1, una vez están hechos los cálculos, aparece el método de resolución usado (véase figura ***II.21***).

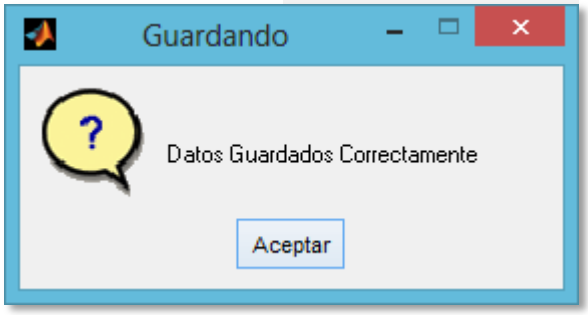
II.21: *Título de la ventana de preproceso modificado tras el cálculo*

Siempre que veamos el nombre del método usado en el nombre de la ventana principal sabremos que los cálculos han sido ejecutados. En este caso, se nos habrán **activado** dos opciones de la barra de menú que antes aparecían desactivadas: La opción guardar como... de la pestaña archivo (figura ***II.22***) y la opción postproceso de la pestaña ventana (figura ***II.24***). En efecto, como ya hemos explicado anteriormente, son dos opciones que sólo se activan tras realizar el cálculo.

La primera sirve para guardar los datos, tanto del *postproceso* como del *preproceso*, para ser abiertos en un futuro. Podremos elegir nombre y ubicación del archivo que queremos guardar. Tras guardar veremos una ventana como la de la figura ***II.23***.



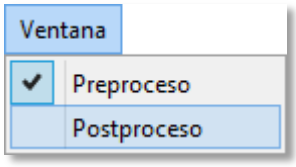
II.22: *Botón de* guardar como... *activado tras realizar el cálculo*



II.23: *Mensaje tras guardar los datos*

El archivo que obtenemos tiene una extensión *.mat (Microsoft Access Table).*

Finalmente podemos mostrar la ventana de *postproceso* clicando el botón postproceso de la pestaña ventana (figura ***II.24***), que ahora se muestra activo.



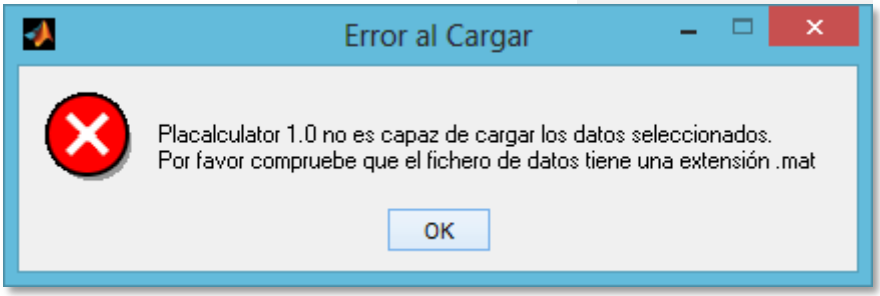
II.24: *Botón de* postproceso *activado tras realizar el cálculo*

2.8. CARGADO DE DATOS

Una alternativa a todo este proceso es simplemente cargar datos que ya estuviesen introducidos en el preproceso y calculados anteriormente por uno de los dos métodos mencionados.

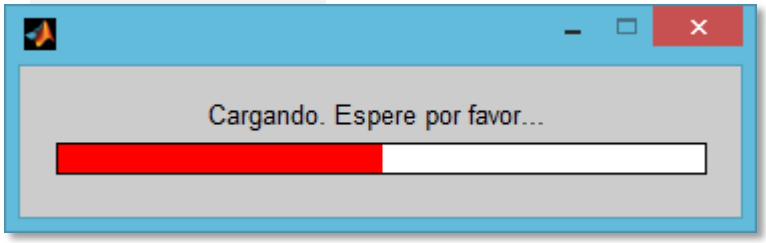
Para ello, el usuario únicamente debe clicar el botón cargar... en la pestaña archivo de la barra de menú (ver figura ***II.4***). El usuario debe navegar por su ordenador hasta clicar en el archivo *.mat* que desee cargar. Este archivo tiene que haber sido creado por el programa *Placalculator 1.0*, ya sea un archivo creado por el mismo usuario o un archivo de ejemplo de los adjuntados.

Si se intenta cargar un archivo que no tenga una extensión *.mat*, el programa mostrará un mensaje de error como el de la figura ***II.25***.

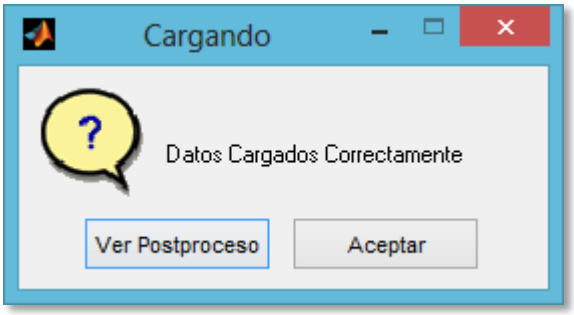


II.25: *Error al cargar un archivo con una extensión diferente a la .mat*

De cargar un archivo *.mat* correcto, el programa mostrará una barra de espera (***II.26***) y también la siguiente ventana (***II.27***):



II.26: *Barra de progreso en el cargado de archivos .mat*



II.27: *Pantalla de éxito en el cargado*

La figura ***II.27*** se corresponde con la figura ***II.20***. Todo lo que se ha dicho a partir de la figura ***II.20*** es aplicable también a partir de éste punto.

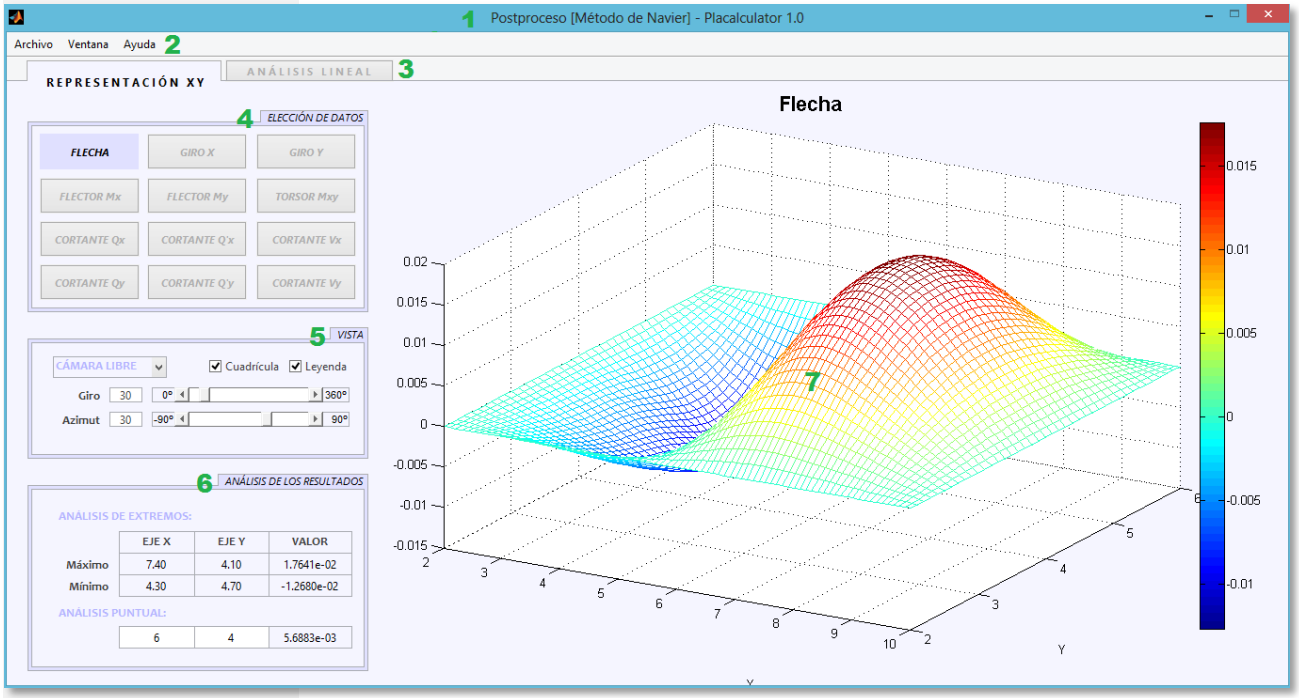
En cualquier caso, el usuario acabará por abrir de una forma u otra la *ventana de postproceso*.

NOTA: *Los datos sólo se pueden cargar en la ventana de preproceso. Si el usuario se encuentra, sin embargo, en la ventana de postproceso, deberá cambiar de ventana para poder cargar datos nuevos.*

3. POSTPROCESO

3.1. VISIÓN GLOBAL – REPRESENTACIÓN TRIDIMENSIONAL

El postproceso es la pantalla que muestra los resultados tras realizar el cálculo de la placa o tras cargar datos existentes:



II.28: *Pantalla de postproceso. Partes que lo componen numeradas de verde.*

Se puede ver que aparece la palabra "*postproceso*" en el **título** **1** de la ventana, seguida del método de resolución usado.

Arriba de la ventana tenemos la **barra de menú** **2** con sus diferentes pestañas.

Justo debajo tenemos la **barra de selección de representación gráfica** **3** (*Tridimensional* o *Lineal*). Esta barra es muy importante, puesto que según elijamos una representación u otra vamos a ver dos pantallas de postproceso notablemente diferentes.

Por ahora vamos a explicar la primera.

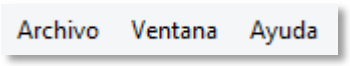
La parte **7** *(ejes x-y-z)* forma la **representación gráfica** del problema. Representan los resultados a partir de los parámetros que marcan las partes **4** y **5**.

La sección **4** se llama "*elección de datos*". Contiene 12 botones, uno para cada **función solución** encontrada (flecha, giros, momentos y esfuerzos).

La sección **5** permite configurar la **vista** de la representación gráfica.

La sección **6** permite analizar los resultados.

3.2. BARRA DE MENÚ



II.29: *Barra de menú*

Nos muestra 3 pestañas: "*archivo", "ventana"* y *"ayuda"*.

Estas tres pestañas se comportan de la misma manera que las mismas descritas en el apartado 2.2. de este capítulo.

Sólo hay ciertas diferencias pequeñas que son:

- archivo > cargar... aparece desactivado. Sólo se pueden cargar datos en la ventana de preproceso.

- archivo > guardar como... aparece activado, puesto que el cálculo ya ha sido realizado.

- ventana > preproceso aparece activado. Ahora podemos pasar de una ventana a otra siempre se desee.

Por lo demás, todas las otras opciones se mantienen idénticas.

3.3. BARRA DE SELECCIÓN DE REPRESENTACIÓN GRÁFICA

II.30: *Barra de elección de representación gráfica*

Esta barra nos permite cambiar el tipo de representación gráfica, y por consiguiente nos cambia todo el *layout* en la *ventana de postproceso*.

Si elegimos REPRESENTACIÓN XY podremos ver en **7** los resultados de una manera tridimensional. La coordenada *z* representa la función solución en función de *x* y de *y*. Irá cambiando, por tanto, según cual sea la solución que estemos representando.

La pestaña ANÁLISIS LINEAL permite hacer un corte de la representación anterior por una coordenada concreta, de manera que lo que vemos es un gráfico lineal de los resultados evaluados en esa misma coordenada.

3.4. ELECCIÓN DE DATOS

II.31: *Elección de datos*

Éste apartado es clave en el postproceso. Aparece en ambas representaciones, la tridimensional y la lineal. Permite elegir datos que se van a representar en **7**.

Se compone de 12 botones:

FLECHA le confiere al eje *z* unidades de longitud [*L*] y representa la función *w(x,y)* en los ejes *x-y-z*.

GIRO X y GIRO Y le dan al eje *z* unidades de longitud lineal [*L/L*] y representan las funciones *gx(x,y)* o *gy(x,y)* en los ejes *x-y-z*.

FLECTOR MX, FLECTOR MY y TORSOR MXY le dan al eje *z* unidades de momento lineal [*FL/L*] y representan las funciones *Mx(x,y)*, *My(x,y)* o *Mxy(x,y)* en los ejes *x-y-z*.

Los 6 botones restantes están asociados a las funciones solución de esfuerzo cortante. Le dan al eje *z* unidades de fuerza lineal [*F/L*] y representan las funciones *Qx(x,y)*, *Q'x(x,y)*, *Vx(x,y)*, *Qy(x,y)*, *Q'y(x,y)* o *Vy(x,y)* en los ejes *x-y-z*.

Se puede acudir a esta sección siempre que se quiera para visualizar unos resultados u otros.

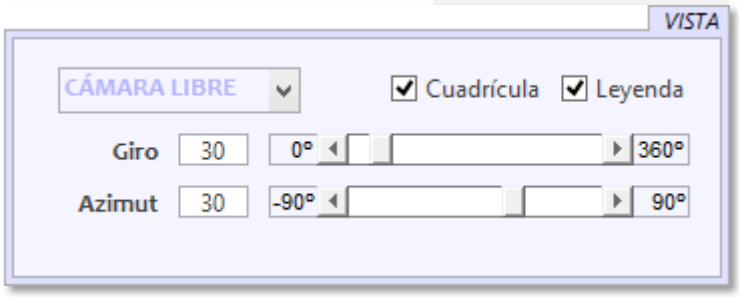
Además de modificar la representación gráfica, también elige la función solución que se va a analizar en **6**.

3.5. VISTA

Esta sección permite configurar la manera en que se presentan los resultados seleccionados en el punto 3.4.

a) *Selector de cámara*

En la esquina superior izquierda encontramos un desplegable con cuatro opciones: *"frontal", "superior", "lateral" y "cámara libre".*



II.32: *Vista*

Cada una de estas opciones modifica el *giro* y el *azimut* de la cámara. De tal manera que la opción *frontal* muestra los ejes *x-z*, *superior* los ejes *x-y* y *lateral* lo hace con los ejes *y-z*. *Cámara libre* es la opción marcada por defecto, que deja la cámara en una posición de 30º para el azimut y para el giro.

b) *Deslizables y casillas de entrada*

Tanto el giro como el azimut de la cámara pueden modificarse, sin embargo, utilizando los deslizables presentes en la figura ***II.32***. O incluso escribiendo el valor numérico deseado en la casilla correspondiente.

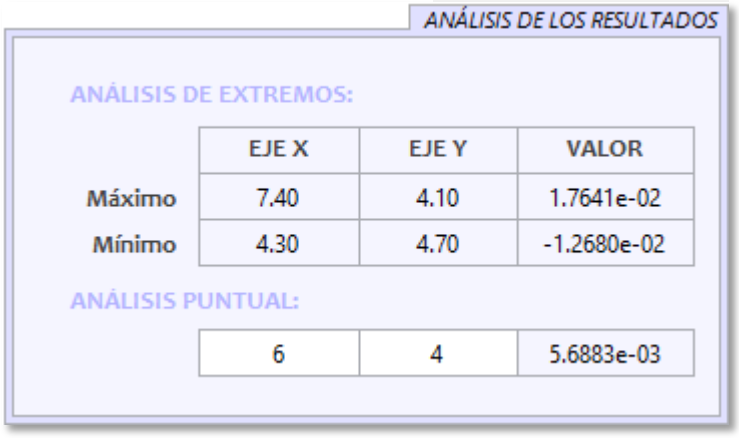
c) *Cuadrícula y Leyenda*

Los dos últimos seleccionables que vemos en la figura corresponden a la vista de la cuadrícula y de la leyenda. Se pueden deseleccionar si no queremos representar dichos elementos, y volver a marcar si, por el contrario, queremos volver a representarlos.

3.6. ANÁLISIS DE RESULTADOS (a)

Esta sección es muy interesante desde un punto de vista ingenieril.

Contiene dos tipos de análisis de la función solución que hayamos elegido en **4**: el **análisis de extremos** y el **análisis puntual**.



II.33: *Análisis de los resultados en representación tridimensional*

a) Análisis de extremos

Esta parte en concreto no es interactiva con el usuario. El programa sitúa el máximo y el mínimo de la función analizada dentro de nuestra **malla** y nos da sus coordenadas (*x,y*) y su valor en las mismas unidades con las que el usuario entró los datos en el preproceso. La ubicación de valores singulares como el máximo o el mínimo pueden ser de gran interés ingenieril, y por supuesto su valor también.

A veces puede que se dé el mismo valor extremal en diversos puntos de la placa, como por ejemplo en placas apoyadas y cargadas uniformemente, donde el máximo de la flecha se va a encontrar en todos los puntos de la malla del contorno con valor *0* (puesto que todos los demás valores serán negativos). En estos casos, el programa tan sólo da el valor del máximo. En ambas coordenadas va a aparecer la palabra multi, que se refiere a que no hay un único punto que presente ese valor de extremo.

Otras veces también se puede dar el caso, en el método de Levy, que en los contornos *y1* e *y2* aparezcan **falsos máximos** o **falsos mínimos**; esto es, valores del orden de 10^{-30}, muy próximos a cero, que deberían ser cero pero no lo son a causa de la limitación física que tiene *Matlab* cuando convierte un conjunto no numerable de números (los reales) en un conjunto numerable. Se pueden dar en los contornos *y1* e *y2* puesto que en el método de Levy la solución homogénea no se desarrolla en la dirección *y*; la función en esta dirección se encuentra calculando los coeficientes A_m, B_m, C_m y D_m. Estos coeficientes son calculados para cada interación *m* con las limitaciones de *Matlab* que acabamos de mencionar. Sin embargo, hay que saber interpretar los resultados y tener en cuenta la posibilidad de la existencia de estos falsos extremos.

NOTA: *Si queremos encontrar un valor más o menos preciso de la ubicación y (por tanto) del valor de los extremos, necesitaremos una malla densa.*

b) Análisis puntual

Esta parte sí que es interactiva con el usuario. Éste último puede escribir las coordenadas *(x,y)* en las que quiere evaluar la función seleccionada, y el *software* la evalúa por él.

En este apartado concreto no se usa la malla para nada. El software usa directamente la expresión analítica de la solución para evaluar un punto en ella.

Es por eso que no necesitamos una buena malla para conocer el valor preciso de un punto concreto.

Como ya se ha dicho anteriormente, **la malla no tiene ningún tipo de influencia en la precisión** del valor de la solución en un punto concreto.

A veces, si se evalúa en este apartado la función solución en las coordenadas de los extremos, su valor no coincide perfectamente con el que nos proporciona el análisis extremal. Esto se debe a que en el análisis puntual se utiliza la **expresión analítica**, que se puede evualuar en cualquier punto *(x,y)*, mientras que el análisis extremal usa únicamente los puntos de la **malla**.

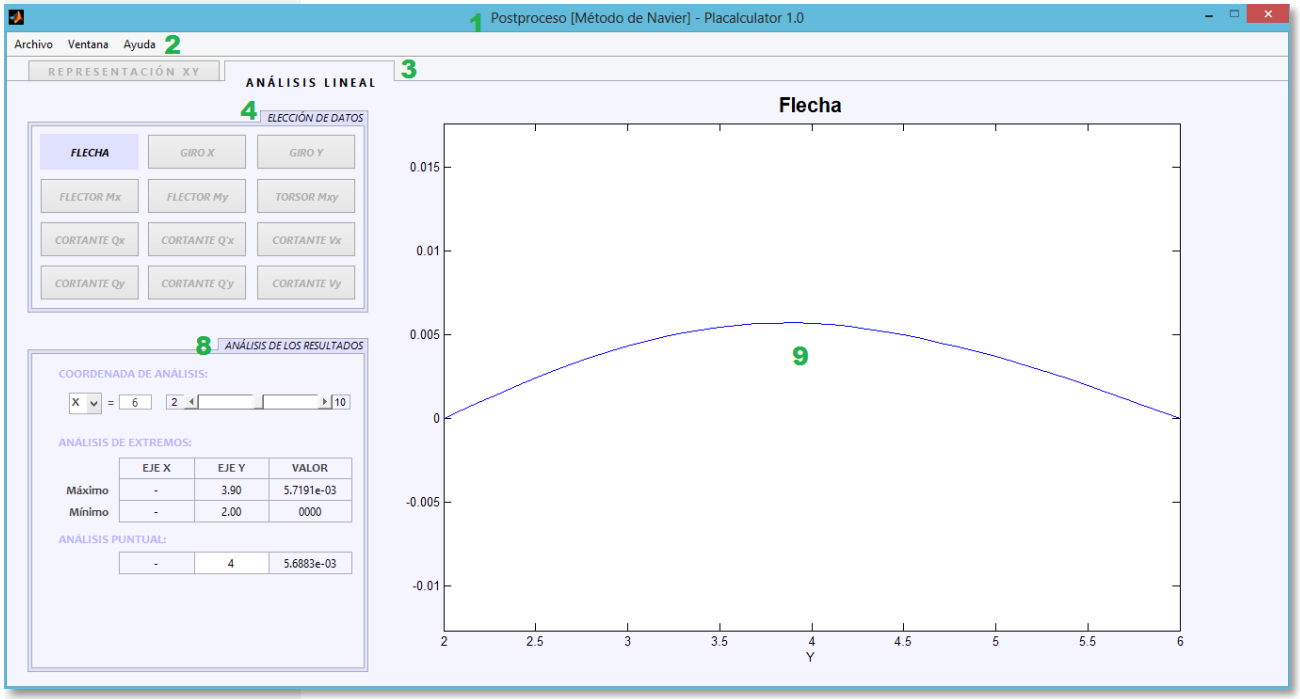
Obviamente, la malla también proviene de la misma expresión analítica. La diferencia está en las coordenadas. Las coordenadas de la malla puede que no sean exactamente las que nos proporciona el software, puesto que sólo nos proporciona las tres primeras cifras significativas. Y puede que al evaluar la expresión analítica en estas coordenadas, éstas no sean exactamente las verdaderas coordenadas de la malla, por lo que el valor que obtenemos difiere ligeramente del anterior.

Por lo tanto, no hay que alarmarse si vemos esta incongruencia. Es un hecho que el usuario debe conocer y tener presente.

NOTA: *Si sólo queremos precisión de resultados en un punto concreto, podemos rebajar la calidad de la malla para aminorar el coste computacional.*

3.7. VISIÓN GLOBAL – REPRESENTACIÓN LINEAL

La segunda parte del postproceso se abre clicando la pestaña ANÁLISIS LINEAL en la barra de selección de representación gráfica **3**:



II.34: *Pantalla de postproceso. Partes que lo componen numeradas de verde.*

Las partes **1**, **2**, **3** y **4** no tienen ningún cambio con respecto a las anteriores. Podemos ver que la sección de vista **5** ha desaparecido, y que las secciones **6** y **7** se han modificado ligeramente, por lo que se las ha renumerado con **8** y **9**.

Así pues se van a explicar las partes **8** y **9**, que son las que presentan cambios con respecto a la pantalla anterior.

La parte **9** *(ejes y-z o los x-z)* es la **representación lineal** del problema. Representa los resultados a partir de los parámetros que marcan las partes **4** y **8**.

La sección **8** equivale al análisis de resultados del punto 3.6, aunque ligeramente diferente.

3.8. ANÁLISIS DE LOS RESULTADOS (b)

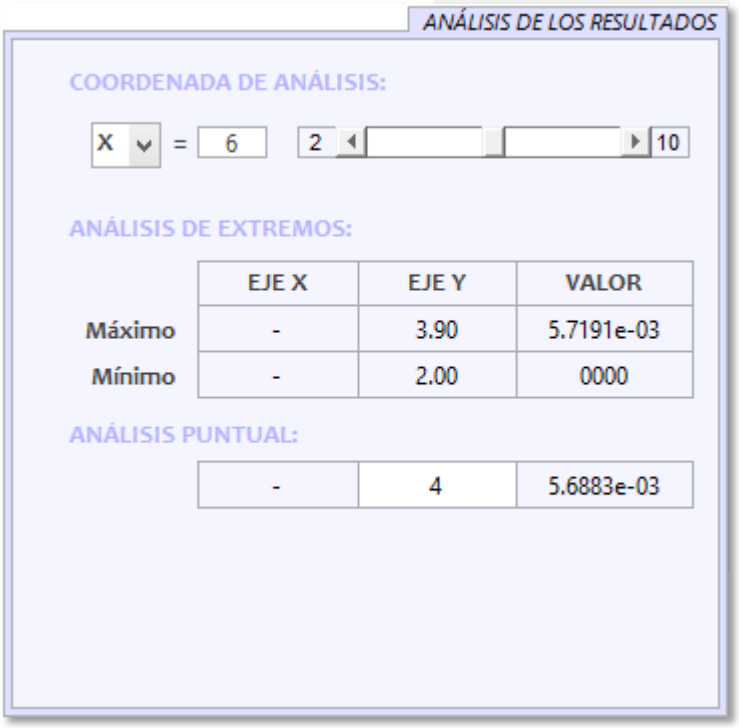
La mayor diferencia con respecto al punto 3.6. es el **dominio de análisis**; mientras antes era una superficie *XY* (toda la placa), ahora va a ser únicamente un **dominio lineal** $x=x^*$ o $y=y^*$.

a) *Coordenada de Análisis*

Lo primero que habrá que hacer será, pues, elegir el dominio de análisis.

La figura ***II.35*** ilustra la sección de análisis de resultados en la representación lineal del problema.

En el margen superior izquierdo encontramos un desplegable que nos da la opción de realizar el corte según la coordenada *x* o según la coordenada *y*.



II.35: *Análisis de los resultados en representación lineal*

Seleccionamos la deseada. Tras hacerlo, podremos elegir mediante el deslizable (o directamente escribiendo en la casilla correspondiente) el valor que corresponde a nuestra coordenada de análisis.

Ya tendremos nuestro dominio $x=x^*$ o $y=y^*$ definido.

b) *Análisis de extremos*

Esta sección funciona de manera igual a la explicada en el punto 3.6. , a diferencia de que en este apartado el dominio de análisis es distinto.

Supongamos que la coordenada elegida es $x=x^*$ (se haría el mismo razonamiento para $y=y^*$). En el primer análisis, por tanto, el software calculará la expresión analítica de la solución restringida a nuestro dominio (esto es, evaluada en $x=x^*$). Más tarde usará el mallado en *y* para generar los *p* puntos (x^*,y_p) en los que va a evaluar esta expresión. De ahí va a sacar cual es la coordenada *y* que se corresponde con el máximo y el mínimo, y nos ofrecerá también el valor que toman estos extremos.

Por lo demás, las explicaciones del punto 3.6. son perfectamente aplicables aquí.

c) Análisis puntual

En esta parte, el usuario debe escribir la coordenada *y* (o *x*) en la que quiere conocer el valor exacto de la solución.

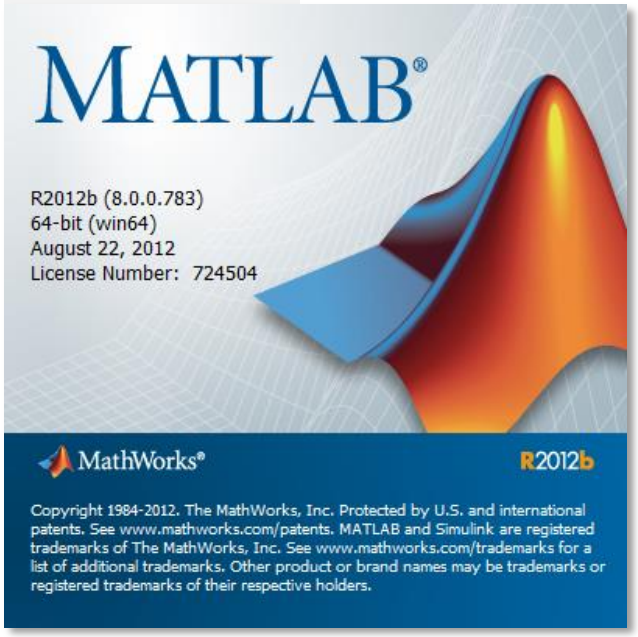
Placalculator 1.0 tomará esta coordenada y evaluará la expresión analítica de la solución en el punto *(x*,y)* (o si se ha definido al inicio el dominio de análisis según *y=y**, en el punto *(x,y*)*).

Por lo demás, todas las conclusiones y notas explicadas en el punto 3.6. son aplicables aquí en este punto también.

CAPÍTULO III

PROGRAMACIÓN EN MATLAB

1. INTRODUCCIÓN A MATLAB



III.01: *Matlab de MathWorks Inc.*

Tal y como se ha dicho en el capítulo II, *Placalculator 1.0* se ha desarrollado mediante el software *Matlab* de *MathWorks, Inc.*

Matlab es un software que trabaja (principalmente) con dos tipos de códigos, los *scripts* y las *functions*.

Para comprender la diferencia entre estos dos códigos, primero hay que hablar de *variables* y de *workspaces* (o áreas de trabajo).

Para este capítulo se ha usado información extraída de las ***fuentes [15-18]***

1.1. VARIABLES

Las variables son los objetos que se pueden almacenar, manipular, extraer o guardar en un *workspace*. Existen muchos tipos de variables.

Pueden ser *char* (cadenas de texto), *numeric* (como *single*, *double*, *integer*, *etc.* diferentes tipos de variables numéricas), *logical* (valores lógicos *1* o *0*), *symbolic* (expresiones simbólicas) o *function handle* (funciones matemáticas), entre otras.

Hay que entender que las variables viven en un cierto *workspace*, por lo que cada una se puede comportar de manera distinta frente a una situación u otra. Es por eso que *workspace* y *variable* son dos términos muy relacionados entre sí.

Como apunte final, remarcamos un tipo de variable importante que se llama **variable global**, que tiene como característica que vive en el *workspace* global (véase punto *1.2*). Las variables globales se deben designar como tales antes de definirlas o manipularlas. Este tipo de variables siempre se puede llamar desde cualquier *workspace*.

1.2. WORKSPACES

Un *workspace* es el lugar que utiliza *Matlab* para almacenar sus variables y trabajar con ellas. Existen diferentes *workspaces:*

a) *Principal*

Es el espacio donde *Matlab* guarda, almacena, busca y manipula la mayoría de variables por defecto.

También es el lugar donde *Matlab* ejecuta por defecto los *scripts* (véase punto 1.3) y los comandos escritos directamente en la *ventana de comandos*.

b) *Global*

Es el espacio donde se guardan, almacenan, buscan y manipulan las **variables globales**.

Las variables en este espacio pueden ser llamadas, manipuladas o grabadas desde cualquier otro *workspace.*

c) *De cada función*

Es el espacio donde trabajan las *funciones*. Cada función tiene su propio *workspace*.

Las variables de entrada y de salida están muy bien especificadas:

Cualquier variable del *workspace* principal que no se especifique como variable de entrada no va a estar presente en el *workspace* propio de la función, y por tanto no se va a poder trabajar con ella. De la misma manera, cualquier variable dentro de la función que no esté especificada como variable de salida no va a ser grabada en el *workspace* principal.

Las variables del *workspace* global, sin embargo, pueden ser manipuladas, definidas, cargadas… desde comandos lanzados desde el *workspace* función, sin necesidad de definirlas como variables de entrada o salida.

1.3. CÓDIGOS

Los códigos son los documentos que contienen escritos los comandos que *Matlab* debe ejecutar. Forman realmente la parte básica y fundamental de la programación. Existen, principalmente, dos tipos de códigos: los *scripts* y las *functions*.

Según el tipo de código, el *workspace* utilizado va a ser distinto, y por tanto las variables en él también lo serán.

a) Scripts

Son códigos que trabajan en el *workspace* principal. Pueden usar, por tanto, cualquier variable contenida en él, o cualquier variable global.

Las variables definidas en un script, por tanto, se van a guardar en el *workspace* principal, a no ser que se definan específicamente como variables globales.

Los scripts pueden lanzar funciones propias de *Matlab*, *functions* definidas por el usuario u otros *scripts* definidos también por el usuario. Es conveniente que estos dos últimos estén guardados físicamente en el pc en el mismo directorio que el script que les llama.

b) Functions

Las *functions* son códigos, igual que los scripts. Trabajan con su propio *workspace*. Inicialmente este *workspace* está vacío cada vez que lanzamos la *function*. Las variables de entrada que tiene definidas son variables presentes en el *workspace* principal. Una *function*, al ser lanzada, copia estas variables del *workspace* principal al suyo propio. También hace lo propio con las variables globales.

Las *functions* pueden lanzar funciones propias de *Matlab*, *scripts* definidos por el usuario u otras *functions* definidas también por el usuario. Sin embargo, hay que tener en cuenta que siempre se van a lanzar dentro del *workspace* propio de la *function*, por lo que muchas veces no van a ejecutarse de manera correcta.

evalin('base', ...)

NOTA: *A veces es necesario que sea una function quien lanze un script u otra function, pero que lo haga en el workspace principal. Para ello se usa la función de Matlab* ***evalin('base',…)****. De no ser así, estos scripts o functions se lanzarían en el workspace propio de la function.*

2. PROGRAMACIÓN INTERNA

2.1. VARIABLES

El programa trabaja con una cantidad considerable de variables. Aquí no se van a enumerar todas. Sin embargo, se van a citar los grandes grupos y se van a explicar las que son verdaderamente relevantes.

a) Layout

La mayoría de variables que se van a generar son para la creación de una interfaz por donde el usuario pueda navegar. Necesitan un nombre para que los diferentes códigos las identifiquen, y hagan operaciones en ellas (mostrarlos, ocultarlos, cambiarles el tamaño, activarlos o desactivarlos…).

Estan definidas en el *workspace* principal, gracias a los comandos que se lanzan desde el preproceso y desde el postproceso.

b) Internas

Son de tipo *numeric*, *char, sym* o *function handle*. Son variables definidas en las *functions*. Son variables que las funciones necesitan generar para funcionar. Viven, por tanto, en el propio *workspace* de la *function* que les llama. Una vez que la *function* acaba de ejecutar todos los comandos que debe realizar, estas variables desaparecen. Estas variables nunca son visibles en el *workspace* principal.

Cada *function* tiene sus propias variables internas.

c) Globales

Son las variables almacenadas en el *workspace* global. Constituyen tanto los datos de entrada del preproceso como los resultados obtenidos en el postproceso tras realizar el cálculo. Pueden ser llamadas, modificadas o utilizadas desde cualquier *workspace* (esto es, desde cualquier *function* o *script*).

Son del tipo *numeric*, *sym* o *function handle*. Se van a listar y explicar cada una de ellas en la siguiente página:

```
>> who global

Your variables are:

D          My         Qx         X2         func_gy    func_qx    id2        x_eval
Gx         N          Qy         Y1         func_mx    func_qy    method_id  y1
Gy         O          Vx         Y2         func_mxy   func_vx    nch        y2
M          P          Vy         calctime   func_my    func_vy    nu         y_eval
Mx         QQx        W          f          func_qqx   func_w     x1
Mxy        QQy        X1         func gx    func qqy   id1        x2
```

III.02: *Variables globales*

P R E P R O C E S O	**D:** Rigidez de la placa	*[num 1x1]*
	nu: Parámetro un de la placa	*[num 1x1]*
	M: Número de sumas de Fourier en *x*	*[num 1x1]*
	N: Número de sumas de Fourier en *y*	*[num 1x1]*
	O: Número de divisiones del mallado en *x*	*[num 1x1]*
	P: Número de divisiones del mallado en *y*	*[num 1x1]*
	X1, X2, X1, Y2: Coordenadas *x1, x2, y1,y2* de la placa	*[num 1x1]*
	id1,id2: Condiciones de contorno (0=Apoyada, 1=Empotrada, 2=Libre)	*[num 1x1]*
	nch: Número de cargas	*[num 1x1]*
	x1, x2, y1, y2: Coordenadas *x1, x2, y1,y2* de cada carga	*[num* **nch***x1]*
	f: Expresión simbólica del valor de cada carga *q(x,y)*	*[sym* **nch***x1]*
	x_eval: Coordenadas *x* de todos los nodos de la malla	*[num* **O***x1]*
	y_eval: Coordenadas *x* de todos los nodos de la malla	*[num* **P***x1]*

[num 1x1]	POSTPROCESO	**calctime:** Tiempo de cálculo [seg]
[num 1x1]		**method_id:** Identificador del método de resolución (1=Navier, 2=Levy)
[function_handle]		**func_w:** Flecha *w(x,y)*
[function_handle]		**func_gx, func_gy:** Giros *gx(x,y), gy(x,y)*
[function_handle]		**func_mx, func_my, func_mxy:** Momentos flectores *Mx(x,y), My(x,y)* y torsor *Mx(x,y),*
[function_handle]		**func_qx, func_qy:** Esfuerzos cortantes *Qx(x,y)* y *Qy(x,y)*
[function_handle]		**func_qqx, func_qqy:** Esfuerzos cortantes *Q'x(x,y)* y *Q'y(x,y)*
[function_handle]		**func_vx, func_vy:** Esfuerzos cortantes *Vx(x,y)* y *Vy(x,y)*
[num **O***x***P***]*		**W,Gx,Gy:** *w(x,y),gx(x,y)* y *gy(x,y)* evaluados en la malla
[num **O***x***P***]*		**Mx,My,Mxy:** *Mx(x,y), My(x,y)* y *Mxy(x,y)* evaluados en la malla
[num **O***x***P***]*		**Qx,Qy,QQx,QQy,Vx,Vy:** *Qx(x,y), Qy(x,y), Q'x(x,y), Q'y(x,y), Vx(x,y)* y *Vy(x,y)* evaluados en la malla.

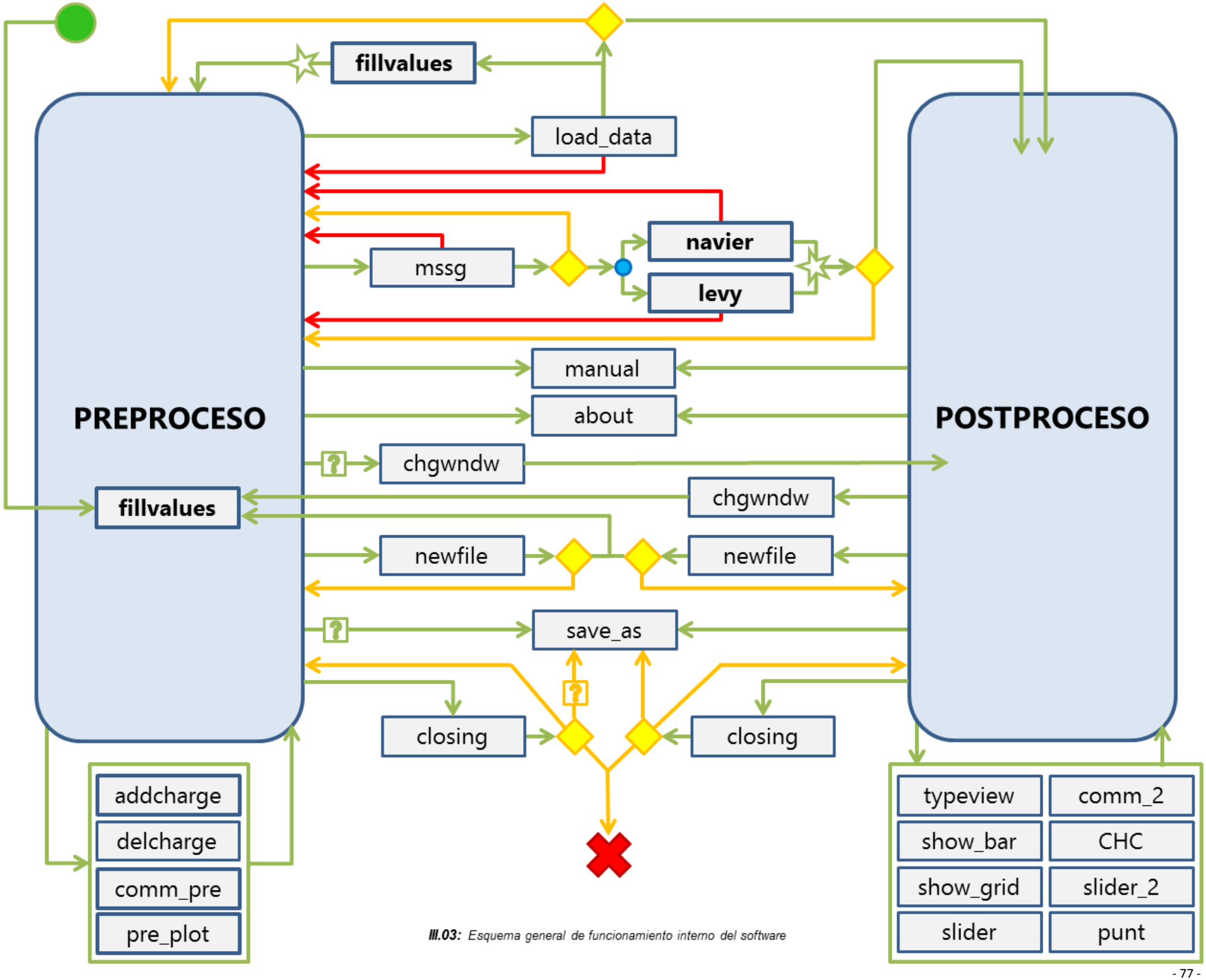
Todas estas variables globales pueden ser definidas o modificadas desde cualquier *script* o *function*. Por ello mismo se han definido como tales.

2.2. ESQUEMA GENERAL DE FUNCIONAMIENTO

La estructura interna de un software de este tipo puede llegar a ser muy compleja, sobretodo de entender. No es posible saber de antemano qué comandos va a ejecutar el usuario, por lo que hay muchos caminos posibles a seguir.

La siguiente página nos muestra de manera gráfica el esquema general de funcionamiento del *software*, que está compuesto por 2 *scripts* y 23 *functions*. En él aparecen todos los caminos posibles que el usuario puede tomar y los comandos a ejecutar que se derivan.

En el punto 2.4 se muestra la leyenda de este gráfico.



III.03: *Esquema general de funcionamiento interno del software*

2.3. LEYENDA

INICIO	Enciende *Placalculator 1.0*
FINAL	Apaga *Placalculator 1.0*
SCRIPT	Ejecuta/vuelve al *script* indicado
FUNCTION	Ejecuta la *function* indicada
FUNCTION *	Ejecuta la *function* indicada (y quizás otras). Funcionamiento detallado más adelante.
PREGUNTA	Mensaje de pregunta. El usuario debe responder afirmativa o negativamente.
FLECHA VERDE	Funcionamiento normal del programa / Respuesta afirmativa del usuario.
FLECHA NARANJA	Respuesta negativa del usuario.
FLECHA ROJA	Mensaje de error. No ejecución de la *function*. Vuelta al script original.
BIFURCACIÓN	El *software* elige qué camino seguir en función de la información obtenida.
INTERRUPTOR	Marca la posibilidad de un camino de estar interrumpido. Inicialmente interrumpido.
LLAVE DE PASO	Activa los caminos antes desactivados por el interruptor.
BLOQUE DE FUNCTIONS	Conjunto de *functions* que pueden ser llamadas **individualmente** desde el *script* de entrada, y posteriormente vuelven a él.

III.04: *Leyenda*

2.4. EXPLICACIÓN DETALLADA DE CADA CÓDIGO – GENERAL

En este apartado se van a explicar todas las *functions* que pueden ser llamadas tanto en preproceso como en postproceso.

a) newfile

Lanza un mensaje de pregunta al usuario sobre si **quiere abrir** o **no** un nuevo documento en blanco. En caso afirmativo se borran todos los datos existentes y se abre la ventana de preproceso de nuevo. En caso negativo simplemente vuelve a la ventana que teníamos activa.

b) save_as

Abre una ventana para que el usuario seleccione un nombre y una ubicación para el archivo de datos *.mat* que va a grabar en su ordenador. Una vez elegidos, graba las variables globales en este nuevo archivo.

NOTA: *Si se ha calculado mediante el método de Navier no se van a guardar las variables id1 e id2, puesto que éstas están vacías.*

c) closing

Abre una ventana de confirmación de salida del programa.

Si se han realizado los cálculos se pueden elegir tres opciones: **guardar y salir**, **salir sin guardar** o **no salir**. Si no se han realizado los cálculos se puede elegir entre **salir sin guardar** o **no salir**.

d) chgwndw

Cierra la ventana actual y abre la que no se estaba mostrando antes (preproceso o postproceso).

e) *manual*

Muestra el pdf que contiene el manual de usuario del capítulo II.

f) *about*

Abre una ventana con la información básica del programa.

2.5. EXPLICACIÓN DETALLADA DE CADA CÓDIGO – PREPROCESO

En este apartado se va a explicar el *script* **pre** y todas las *functions* que pueden ser llamadas en él.

a) *pre*

script que muestra la ventana de preproceso. Cada vez que es llamado ejecuta automáticamente la *function* **fillvalues** (ver punto 2.5g) para rellenar las casillas de sus datos correspondientes y dibujar el preproceso (en caso de que se haya realizado el cálculo).

b) *load_data*

Se abre una ventana para que el usuario seleccione un archivo de su ordenador. Si el archivo no tiene la extensión correspondiente (*.mat*) se recibirá un mensaje de error.

Si, por el contrario, el archivo es correcto, se procederá borrar los datos antiguos y cargar los nuevos en el *workspace* principal. Una vez hecho, se ejecuta en éste mismo la *function* **fillvalues** (ver punto 2.5g).

Acto seguido, se le pregunta al usuario si quiere ver el postproceso. En caso afirmativo, éste se carga. En caso negativo se muestra el preproceso.

c) *pre_plot*

Se encarga de dibujar el preproceso. Es llamado por cualquier elemento (casilla, desplegable…) que deba afectar a la representación de placa, cargas o malla. También puede ser llamado por las *functions* **addcharge** o **comm_pre** (ver puntos 2.5d y 2.5f).

Lo primero que hace siempre es borrar todos los dibujos realizados y desactivar todas las pestañas en menú>representar, a excepción de la pestaña ejes.

Si encuentra algún error en los datos de entrada de la placa no dibuja **nada**.

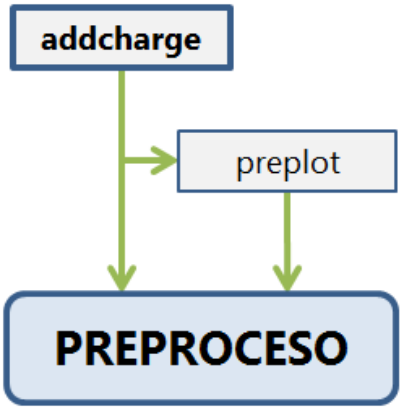
Si no hay ningún problema con la placa, la dibuja junto con la malla y las cargas.

Sin embargo, si hay algún error en los datos de la malla no la dibuja. De manera paralela, no dibujará aquella carga que presente anomalías.

A medida que va verificando que los datos se han introducido correctamente y los va dibujando, va activando las pestañas correspondientes en menú>representar.

NOTA: *Para saber más sobre introducción correcta de los datos, véase capítulo II – Manual de usuario.*

d) *addcharge*



III.05: *Esquema de la function addcharge*

Añade una carga al problema.

Primero observa cuántas cargas existen y luego modifica el layout: añade una pestaña debajo de las ya existentes y desplaza las pestañas + y parámetros de cálculo hacia abajo. También añade esta carga en la pestaña representar de la barra de menú. Acto seguido llama a la *function* **preplot** (véase punto 2.5c) para que (si es posible) la represente en los ejes.

Puede ser llamada por la *function* **fillvalues** (ver punto 2.5g)

NOTA: *addcharge no se puede ejecutar si ya hay cinco cargas en la ventana de preproceso.*

e) *delcharge*

Elimina una carga del problema.

Primero observa cuántas cargas existen y luego modifica el layout: elimina la pestaña de la última carga y desplaza las pestañas + y parámetros de cálculo hacia arriba.

También elimina esta carga en la pestaña representar de la barra de menú.

Acto seguido invisibiliza la carga en la representación gráfica.

Puede ser llamada por la *function* **fillvalues** (ver punto 2.5g).

NOTA: *delcharge sólo se puede ejecutar en la última carga disponible, y siempre que ésta no sea la única.*

NOTA: *delcharge no elimina los datos. Tan sólo elimina la pestaña de la ventana. Esta carga eliminada se puede recuperar usando addcharge (ver punto 2.5d)*

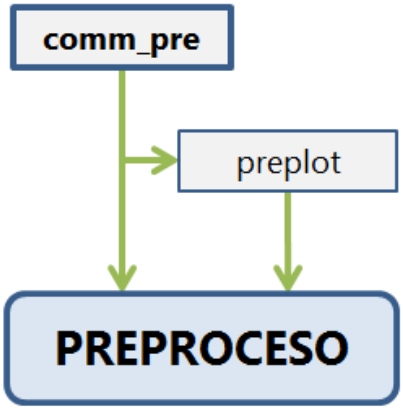
f) *comm_pre*

Es el comando que se ejecuta cuando se pulsa cualquier pestaña (excepto añadir carga y eliminar carga) en el preproceso.

Es el encargado de modificar cada vez el *layout* para que se corresponda con la pestaña seleccionada (ya sea en el selector de método de cálculo o en la entrada de datos).

Lo primero que hace es buscar todos los elementos que se pueden representar dentro del apartado introducción de datos y los invisibiliza. Luego muestra los elementos que corresponden con la pestaña clicada. De esta manera, en un solo código se pueden agrupar acciones diversas de cada pestaña.

Una vez ha hecho esto, llama a la *function* **preplot** (véase punto 2.5c) para que muestre una representación gráfica acorde con las pestañas activas.



III.06: *Esquema de la function comm_pre*

g) *fillvalues*

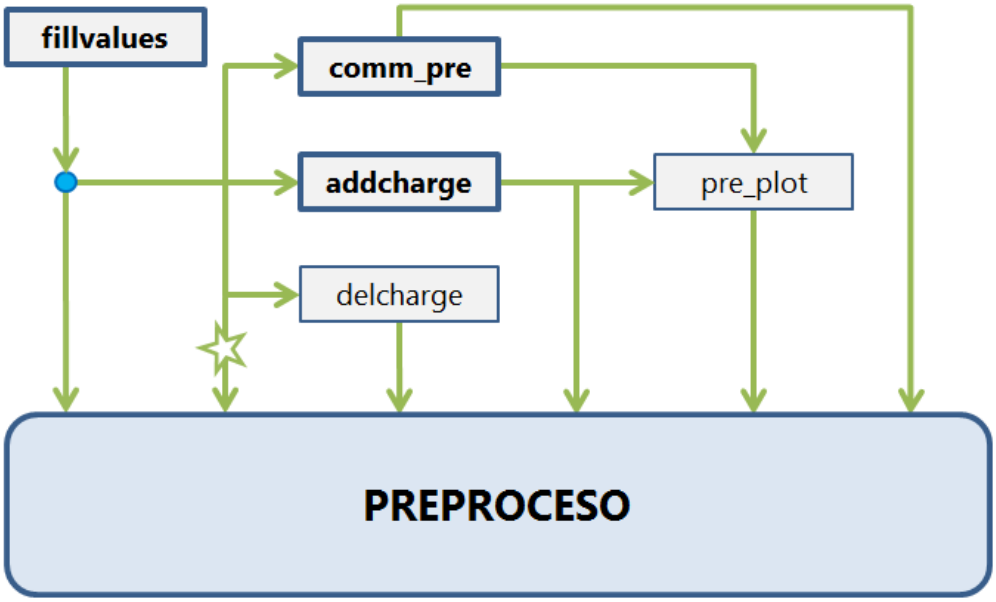
Se ejecuta tras lanzar el *script* **pre,** las *functions* **newfile** y **chgwndw** (ya que lanzan el *script* **pre**) y tras la *function* **load_data**. Sirve para rellenar las casillas en introducción de datos de los datos correspondientes.

Estos datos son las **variables globales de preproceso**. Si estas variables están vacías vuelve directamente al preproceso. Sino, rellena las casillas correspondientes.

Además, también escribe en el título de la ventana el método de resolución usado y activa las *functions* chgwndw y save_as en el preproceso.

Si las variables globales de preproceso no están definidas, lógicamente no se ejecuta.

Una vez ha terminado, llama a las *functions* **comm_pre**, **addcharge** y **delcharge** para que muestren un layout y una representación gráfica acorde con los datos rellenados. Cada vez que se lanza **fillvalues** se llaman a otras múltiples *functions* que a su vez llaman a otras *functions*.



III.07: *Esquema de la function fillvalues*

h) *mssg*

Lanza un mensaje de confirmación para el lanzamiento del cálculo si todos los datos se han introducido correctamente. En caso afirmativo lanza la *function* **navier** o **levy** en función de los datos que haya rellenado el usuario. En caso negativo vuelve al preproceso.

Si, por el contrario, el programa detecta algún error en la entrada de los datos, lanza un mensaje de error como los mostrados en el capítulo II y fuerza al uusuario a volver al preproceso para corregir los datos.

2.6. EXPLICACIÓN DETALLADA DE CADA CÓDIGO – CÁLCULO

En este apartado se van a explicar las *functions* **navier** y **levy**, que son los dos códigos que nos calculan nuestra placa. Los dos son lanzados siempre a través de la *function* **mssg** (véase punto 2.5h). Ambos disponen de un control de posible error durante el cálculo. Es decir, si se detecta un error no esperado, ambos códigos dejan de calcular y lanzan un mensaje de error, de manera que el programa no se quede trabado y se pueda volver al preproceso para corregir los datos de entrada.

De todas formas ya se ha hecho un control de posibles errores en la etapa anterior, en la *function* **mssg**. Sin embargo, esta manera de proceder nos da mayor seguridad para darle robustez al *software.*

a) navier

Código que calcula el estado tensional y deformacional de nuestra placa a partir de los datos de entrada que hemos impuesto, mediante el **método de Navier.**

Lo primero que hace es empezar a contar los segundos que tarda en calcular. Esta información le será dada al usuario cuando se finalicen los cálculos.

Acto seguido se dispone a grabar (o mediante pequeñas operaciones a calcular) nuestros datos de entrada como **variables globales de preproceso** (hasta ahora estas variables estaban vacías, no se les había asignado ningún valor). De esta manera, al terminar de calcular, siempre que se quiera volver a cargar el preproceso la *function* **fillvalues** tendrá los datos disponibles en el *workspace* global para usarlos.

NOTA: *las variables globales id1 e id2 no se rellenan en este método, puesto que no se pueden elegir las condiciones de contorno.*

Todas las cargas son tratadas de la misma manera. Para cada una de ellas, el código se encarga de ordenar de mayor a menor las coordenadas en *x* y en *y,* y más tarde le añade a cada coordenada un diferencial de longitud *épsilon* hacia el exterior. De esta manera, si hubiese una carga puntual o lineal, el código las transforma automáticamente en cargas repartidas. Lógicamente, si las cargas son realmente lineales o puntuales, el programa se encargará también de dividir su valor entre *2·épsilon* o $(2 \cdot épsilon)^2$ respectivamente para que la fuerza total aplicada se corresponda con los datos de partida.

Para el valor de las cargas *q(x,y),* el programa trabaja siempre con variables **simbólicas** ya que son apropiadas para realizar operaciones como integrales, límites, derivadas, etc. Si las variables *x* e *y* están escritas en mayúsculas, **navier** las transforma en minúsculas.

El programa se dispone a realizar los cálculos [5] pertinentes con variables simbólicas *x* e *y,* (tal y como se ha explicado en el punto *2.1.* del capítulo I), hasta que realiza el cálculo de los coeficientes w_{mn}. Una vez ha computado el valor de estos coeficientes, calcula el límite de éstos cuando *épsilon* tiende a cero. De esta manera, los coeficientes w_{mn} tanto de las cargas repartidas como de las cargas lineales o puntuales son calculados correctamente, utilizando un mismo procedimiento para las tres tipologías de carga definidas.

Mediante la formulación *(I.43)* y *(I.44)* del capítulo I el programa consigue la expresión *w(x,y)* analítica de la flecha en lenguaje simbólico. Una vez llegado a este punto, calcula de una manera simple, también con lenguaje simbólico, las derivadas en *x* y en *y* de la expresión *w(x,y)* y, en definitiva, es capaz de calcular las expresiones para los giros, momentos y esfuerzos mediante la formulación también explicada en el capítulo I.

En todas las expresiones anteriores los ejes de coordenadas estaban situados en una esquina de la placa. Por tanto, una vez tiene todas las expresiones escritas, el programa realiza un cambio de variable en *x* y en *y* para desplazar los ejes de coordenadas al punto donde le corresponden, con tal de que la placa se sitúe en las

[5] Ver *anejo 2: Cálculo en Matlab detallado*

coordenadas designadas por el usuario.

El siguiente paso es **transformar estas variables de tipo *simbólico* a tipo *function handle*.** Este paso es muy importante antes de evaluar las funciones en los nodos de la malla. Las variables tipo *sym* son muy apropiadas para realizar cálculos del tipo integración, derivación, límites… con ellas, pero no son muy eficientes a la hora de ser evaluadas en *x* e *y*. Aunque se puede hacer todo el proceso de evaluación mediante variables tipo *sym*, las variables tipo *function handle* son mucho más eficientes; evalúan las funciones en *x* e *y* con una velocidad del orden de 10^4 veces superior a las citadas anteriormente. Esto permite reducir de manera drástica el tiempo de computación cuando se implementan mallas extensas.

Una vez tiene todas las funciones solución calculadas, con los ejes de coordenadas desplazados y transformadas a variables tipo *function handle* en función de *x* e *y*, las **guarda** como **variables globales de postproceso** en el *workspace* global. De esta manera, se podrán recuperar siempre que se llame al *script* **post** o se quieran guardar los datos desde el preproceso o el postproceso.

El siguiente paso es evaluar todas estas funciones en los nodos de la malla. Es una operación sencilla puesto que se han definido las variables como tipo *function handle*.

Una vez ha finalizado el cálculo de los valores en los nodos de la malla, el código actúa automáticamente como llave de paso y activa las *functions* **chgwndw** y **save_as** desde la ventana de preproceso. También añade en el título de la ventana el nombre del método utilizado (en este caso método de Navier).

Finalmente, graba la **variable global** *method_id* con valor 1 en el *workspace* global, para que al volver a cargar en un futuro la ventana de preproceso, el código **fillvalues** sepa qué método se ha usado y muestra un *layout* correcto.

En este momento detiene el cronómetro y muestra una ventana de finalización de tiempo de cálculo al usuario. En esa ventana se puede leer el tiempo que ha tardado el *script* en calcular y se puede elegir entre **visualizar el postproceso** o **volver al preproceso**. En función de lo que elija el usuario, se va a borrar la ventana de preproceso y se va a lanzar el *script* **post** (véase punto 2.7a) o de manera contraria se cerrará esta ventana, volviendo al preproceso.

b) levy

Levy es el código que calcula el estado tensional y deformacional de nuestra placa a partir de los datos de entrada que hemos impuesto, ésta vez mediante el **método de Levy.**

Tiene un funcionamiento muy parecido al código que acabamos de explicar, pero sin embargo con ciertas diferencias que vamos a explicar a continuación.

El código empieza de la misma forma que la *function* **navier**. Sin embargo, ésta sí que guarda *id1* e *id2* como **variables globales de preproceso**, puesto que sí que van a influir en este caso sobre los resultados.

Luego avanza de la misma forma que el método de Navier hasta que llega a computar la expresión *w(x,y)* (antes de mover desplazar los ejes y de transformarla en *function handle*). Esta expresión, en el método de Levy, tan sólo va a formar la solución particular. Por tanto, **levy** en este punto se dispone a calcular también la solución homogénea.[6]

Para ello sigue usando el lenguaje simbólico. Escribe la ecuación *(I.46)* del capítulo I con las incógnitas *A B C* y *D* y, en función del valor de las variables *id1* e *id2* (que van asociadas a las condiciones de contorno), y la suma a la expresión anterior para tener la expresión de la flecha total (particular más homogénea).

Con esta expresión y con sus derivadas escribe el sistema de ecuaciones descrito en la figura *I.14* del capítulo I. Este sistema lo puede resolver para cada iteración *m*, por lo que encuentra los coeficientes *A*, *B*, *C* y *D* y, por tanto, la expresión de la solución general de la flecha.

Una vez llegados a este punto sigue los mismos pasos que **navier**, a excepción de la **variable global *method_id***, que la graba en el *workspace* global con un valor de 2 (el que se corresponde con el método de Levy) y a excepción también del título de la ventana de post y preproceso, que marca con el nombre “método de Levy” en este caso.

Todos los demás pasos son exactamente iguales que en **navier.**

[6] Ver *anejo 2: Cálculo en Matlab detallado*

2.7. EXPLICACIÓN DETALLADA DE CADA CÓDIGO – POSTPROCESO

En este apartado se va a explicar el *script* **post** y todas las *functions* que pueden ser llamadas en él.

a) *post*

script que muestra la ventana de postproceso. Por defecto, cuando es llamada, muestra la representación tridimensional de la flecha con una vista de 30º para el azimut y para el giro de la cámara.

b) *typeview*

Vamos a ver primeramente las functions lanzadas desde el análisis tridimensional.

Typeview se lanza cuando se clica sobre el desplegable en la sección vista precisamente del postproceso tridimensional.

Simplemente modifica la posición de la cámara (giro y azimut) en función de la opción seleccionada (cámara frontal, lateral, superior o libre).

También modifica a su vez los deslizables y las casillas correspondientes al giro y al azimut para que marquen los valores que se acaban de seleccionar.

c) *showbar*

Muestra la leyenda que aparece al lado de la representación gráfica tridimensional. Si ésta ya era visible, la hace desaparecer.

d) *showgrid*

Muestra el conjunto de líneas punteadas auxiliares que aparecen en la representación gráfica tridimensional. Si éste ya era visible, lo hace desaparecer.

e) slider

function que es lanzada cuando se clica sobre el deslizable de giro o azimut en la parte tridimensional del postproceso o se escribe directamente su valor en la casilla correspondiente.

De manera análoga a la *función* **typeview**, modifica la posición de la cámara en función del valor que tomen el giro y el azimut en los deslizables correspondientes.

También modifica las casillas correspondientes al giro y al azimut para que marquen los mismos valores que en el deslizable, si es el deslizable quien ha lanzado la *function*. Si, por el contrario, este código ha sido lanzado escribiendo el valor exacto en la casilla, se va a modificar el valor de la variable en el deslizable.

Por último, marca en el desplegable de elección de cámara la opción cámara libre.

f) slider2

Pasamos ahora a ver las *functions* que sólo afectan al postproceso en **análisis lineal**.

La primera que quería comentar es la función **slider2**, que está asociada al deslizable que marca la coordenada de análisis y a su casilla correspondiente.

Una vez el usuario mueve el deslizable para seleccionar el valor de la coordenada ($x=x^*$ o $y=y^*$) a analizar, lo primero que hace este código es rellenar la casilla con el mismo valor. Si por el contrario, este código ha sido lanzado desde la casilla, hace lo propio con el deslizable.

Una vez la coordenada x^* o y^* ha sido seleccionada, el programa se encarga de evaluar la función solución que esté marcada en la sección elección de datos en esta misma coordenada. Obtiene, por tanto, una función que solo depende de y o de x (la variable que no se ha seleccionado como **coordenada de análisis**). Esta función es evaluada en los nodos que surgen de intersecar la malla con la coordenada de análisis.

Por último, actualiza los datos de análisis de datos para mostrar el valor máximo y el valor mínimo de la función representada.

g) *CHC*

Es el código asociado al desplegable de selección de coordenada a analizar, que puede tomar valor de *x* o de *y*.

Cuando el usuario elige una de las dos opciones, esta *function* es ejecutada.

Lo primero que hace es elegir, para el deslizable y su casilla asociada, la coordenada central de la variable elegida. Es decir, si se ha elegido la variable *x* como coordenada de análisis, tanto el deslizable como su casilla asociada van a tomar el valor promedio entre *x1* e *x2*. Esto es así por defecto, cada vez que el usuario cambia de coordenada, para que ni la casilla ni el deslizable tomen valores fuera del rango de análisis.

Más tarde, el procedimiento seguido es exactamente igual que en la *function* **slider2**:

Se evalua la función solución que esté marcada en la sección elección de datos en la coordenada central de la variable elegida. Obtiene, por tanto, una función que solo depende de *y* o de *x* (la variable que no se ha seleccionado como **coordenada de análisis**). Esta función es evaluada en los nodos que surgen de intersecar la malla con la coordenada de análisis.

Por último, actualiza los datos de análisis de datos para mostrar el valor máximo y el valor mínimo de la función representada.

h) *punt*

Por último, describimos las dos *functions* que se lanzan tanto desde el análisis lineal como desde el tridimensional.

La primera se llama **punt**. Es el código que va asociado al **análisis puntual**, en el cual el usuario puede elegir la(s) coordenada(s) donde quiere evaluar la función solución para obtener su valor exacto.

Si el usuario está situado en el **análisis tridimensional**, deberá elegir tanto la coordenada de análisis *x* como la *y*.

En ese caso, **punt** evalúa la función solución seleccionada, en las coordenadas *x** e *y** que ha escrito el usuario en el apartado de análisis puntual, y muestra el resultado en la casilla correspondiente.

Si, por el contrario, el usuario está situado en **análisis lineal**, sólo podrá elegir en el apartado de análisis puntual una de las coordenadas, puesto que la otra ya la ha elegido con anterioridad en el desplegable de elección de coordenada de análisis.

En este caso, el programa toma el par de coordenadas definidas por el usuario (una en el desplegable, y otra en la casilla de análisis puntual) y las evalúa en la función solución seleccionada. Acto seguido muestra el resultado en la casilla correspondiente, igual que en el caso de análisis tridimensional

NOTA: *Este código no va a cambiar nunca la representación gráfica de la solución, ya sea en análisis tridimensional o análisis lineal. Todos los códigos explicados con anterioridad sí podían hacerlo de una forma u otra.*

i) comm_2

Éste es el último código del programa.

Es lanzado cuando se pulsan las pestañas de selección de representación gráfica o los botones en la sección de elección de datos. Se puede lanzar tanto desde el postproceso en análisis tridimensional como en análisis lineal.

Es por eso que es el código más completo del postproceso. Es capaz de controlar la **interfaz** (en función de quién lanza el código sabe cuáles son los elementos que debe mostrar y los que no), controlar la **representación gráfica** de la solución (tanto en análisis tridimensional como en análisis lineal) y controlar la **actualización de** los valores de los **máximos y mínimos** en todo momento de la función analizada.

Los procedimientos seguidos en este último caso ya se han explicado detalladamente en el punto 2.7*h*.

CAPÍTULO IV

ANÁLISIS DE LOS RESULTADOS

1. EJEMPLOS DE VALIDACIÓN

1.1. INTRODUCCIÓN

En este apartado se han recopilado ejercicios resueltos a mano, extraídos de diferentes fuentes citadas en la bibliografía.

Se trata de comparar el resultado obtenido por nuestro software de cálculo con el que proporcionan estas fuentes.

Los ejercicios que se deban resolver mediante el método de Levy van a ser resueltos aplicando este mismo método. Sin embargo, los ejercicios que han sido resueltos por el método de Navier van a ser contrastados con los que obtengamos en nuestro software aplicando tanto éste método como el de Levy. De esta forma, podremos hacer también un análisis comparativo entre ambas maneras de resolver.

Como se ha dicho anteriormente, los ejemplos presentes en la bibliografía están resueltos a mano, por lo que sólo se ha desarrollado la función carga $q(x,y)$ y la función flecha $w(x,y)$ en una sola serie de Fourier.

Esto significa que deberemos desarrollar las cargas en la dirección (o las direcciones) elegidas por el ingeniero calculista, pero sólo en una sola serie de Fourier. Por lo que sabemos que estos resultados no son todo lo precisos que podrían ser. En este apartado vamos a convivir con ello: no buscamos un resultado preciso aplicando una metodología compleja, sino que buscamos el mismo resultado aplicando la misma metodología. Más adelante, en el apartado 2 de este capítulo, se van a realizar análisis de convergencia para encontrar soluciones precisas a los problemas planteados.

El mallado no va a influir en la precisión de nuestros resultados. Únicamente va a influir en la representación gráfica de nuestras funciones solución. Podríamos elegir un mallado muy ancho, con pocos puntos. Pero el tiempo de cálculo no se ve incrementado de manera notable con una malla más refinada, por lo que se han usado mallas relativamente precisas.

NOTA: *Todos los archivos .mat de cálculo están adjuntados y disponibles para ser cargados al software, de manera que se puedan analizar con detalle los ejemplos que mencionaremos seguidamente.*

1.2. EJEMPLO 01

El primer ejemplo seleccionado se ha extraído de la ***fuente [01].***

XVII.B.3.2. EJEMPLO

Se trata de determinar la deformación de la placa apoyada en los cuatro bordes y cargada uniformemente en su mitad según se indica en la fig. XVII.B.3.

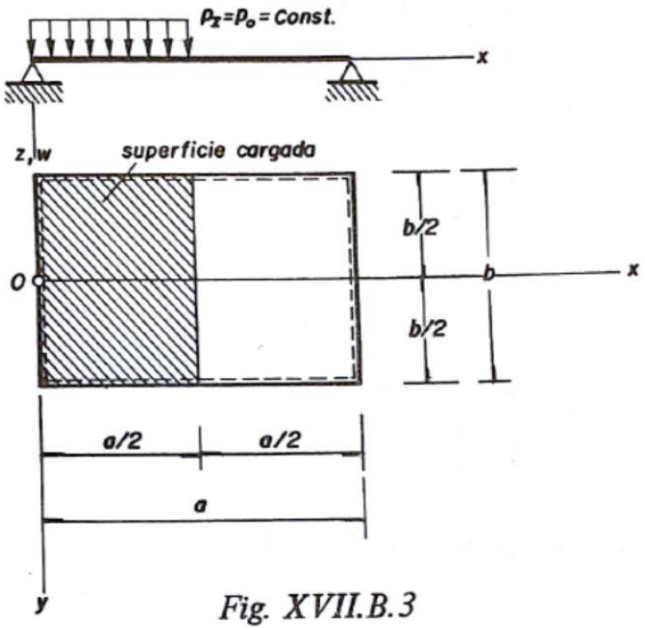


Fig. XVII.B.3

IV.01: *Enunciado del ejemplo XVII.B.3.2 de la* ***fuente [01]***

Se trata de una placa apoyada en sus cuatro bordes, de dimensiones *a* x *b* y de parámetros elásticos nu $\gamma=0$ y rigidez *K*, cargada en su mitad izquierda por una carga de valor *p* constante.

El resultado que se da en el mismo libro usando la metodología de Navier es el siguiente:

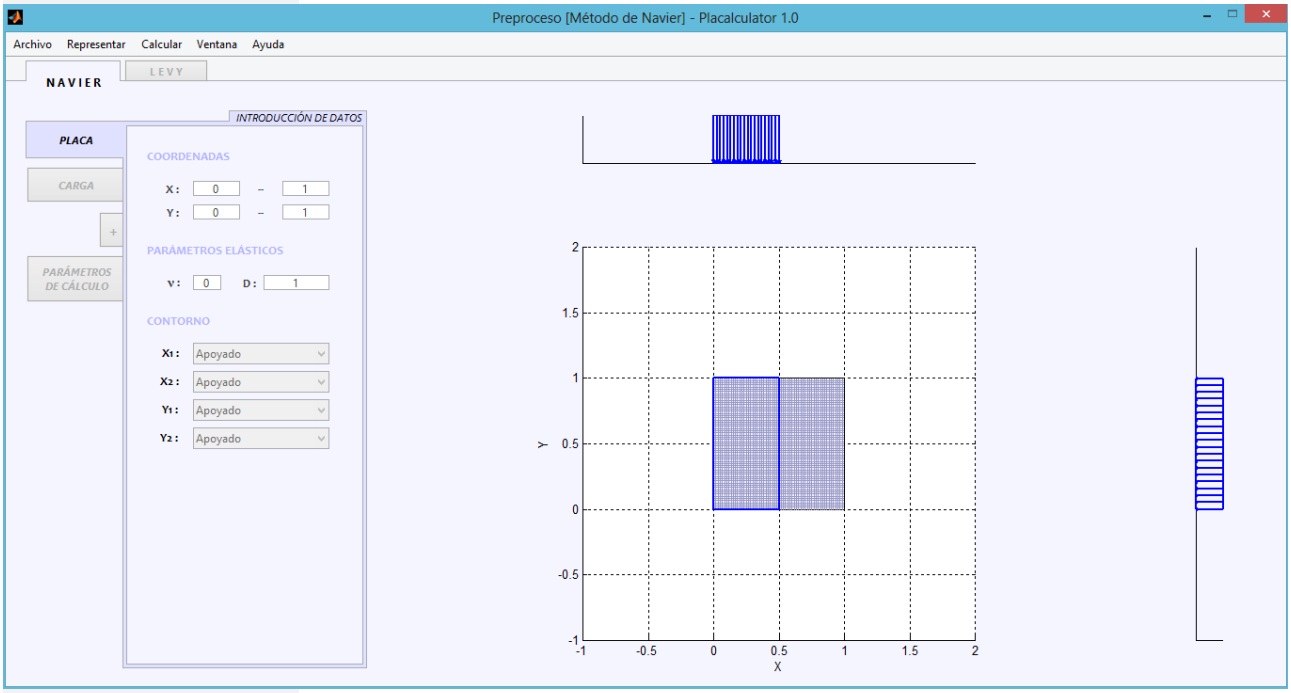
$$(w)_{x=\frac{a}{2},\ y=0} = 0{,}00208\ \frac{p\,a^4}{K}$$

IV.02: *Solución del ejemplo XVII.B.3.2 de la* ***fuente [01]***

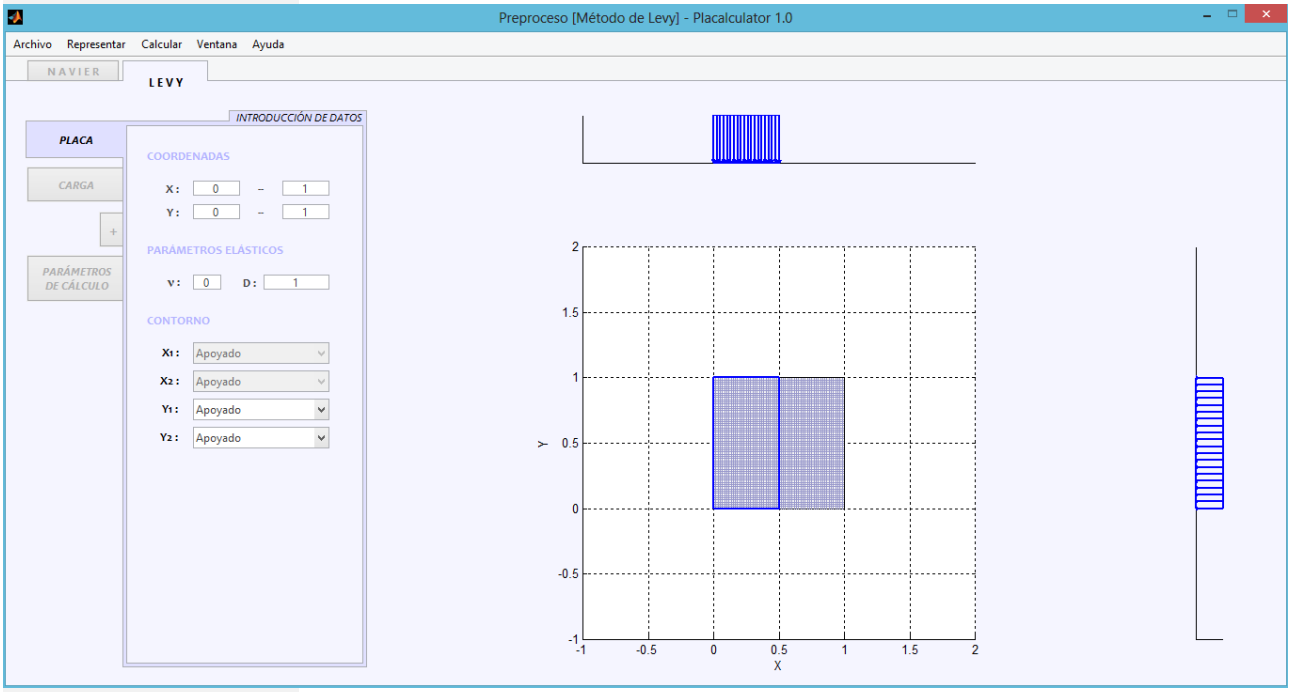
Podemos ver que el resultado se da para *a=b*, y en función de *a*, p y *K*. Nuestro software sólo trabaja con parámetros determinados. Así pues, debemos elegirlos antes de empezar a calcular.

Para simplificar la comparación, se han elegido valores unitarios para dichos parámetros:

$$a=b=K=p=1$$

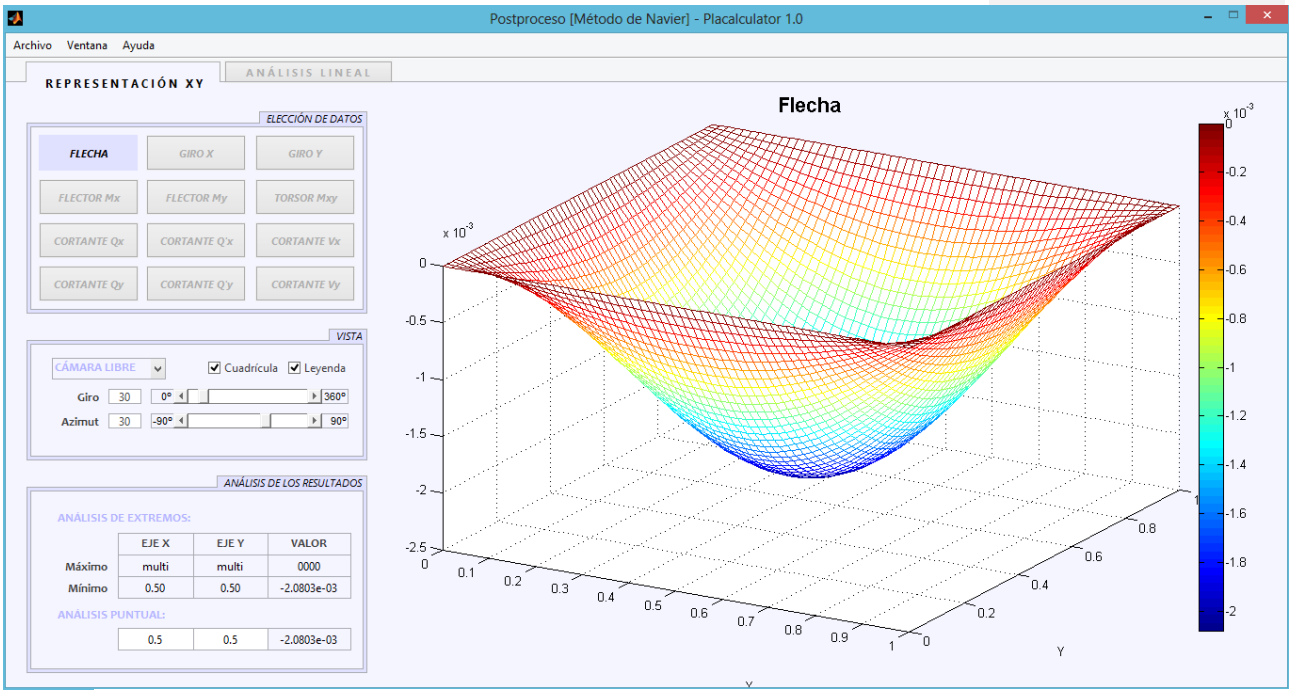


IV.03: *Preproceso del ejemplo 01. Método de Navier*

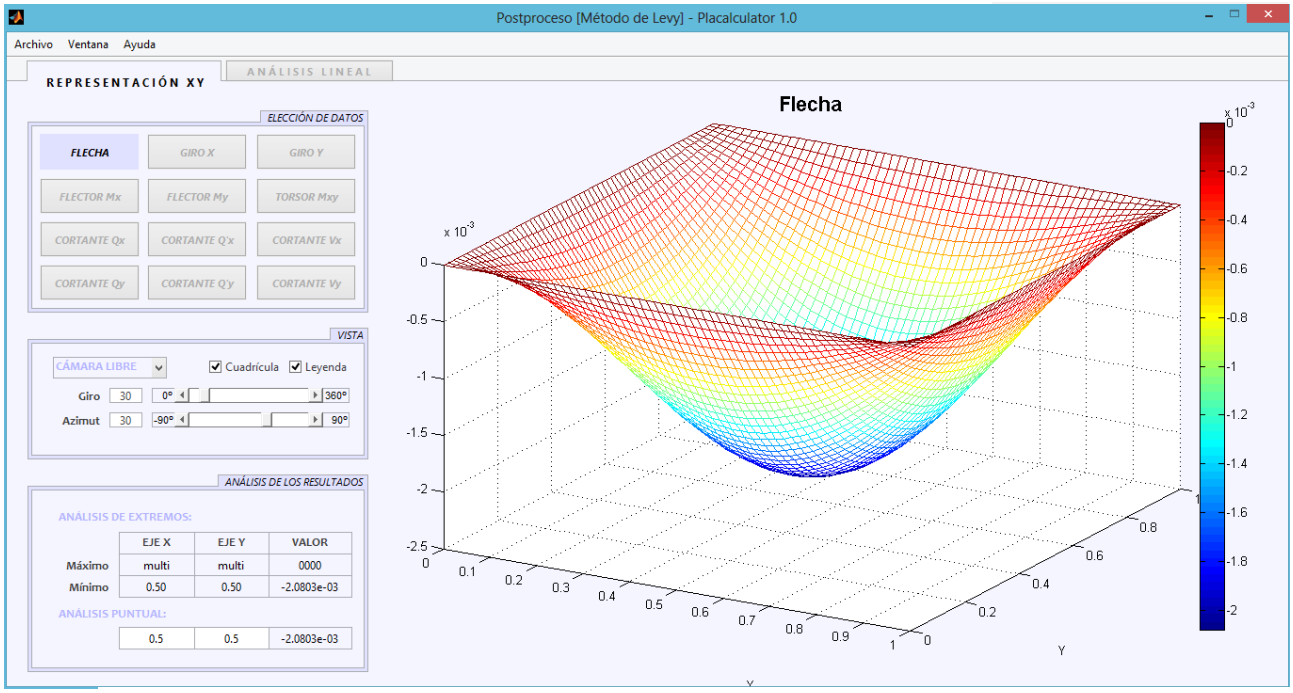


IV.04: *Preproceso del ejemplo 01. Método de Levy*

El problema ha sido resuelto mediante Navier y mediante Levy. Ambos han desarrollado la carga y la flecha en las dos direcciones, *x* e *y*, en una sola serie de Fourier. Los resultados obtenidos son los siguientes:



IV.05: *Postproceso del ejemplo 01. Método de Navier*



IV.06: *Postproceso del ejemplo 01. Método de Levy*

Seguidamente, podemos proceder a analizar la flecha en el centro de la placa. Como es lógico, al sólo disponer de una serie de Fourier en cada dirección, la solución es simétrica en *x* y en *y*.

Por lo tanto la flecha máxima se va a dar en el centro de la placa justamente. Podemos ver que Navier y Levy nos dan exactamente el mismo resultado para la flecha en el centro de la placa: *0.0020803*. Éste resultado es muy parecido al resultado que se da en la bibliografía, que resulta ser de *0.00208*.

Vemos, por tanto, que los cálculos se validan para este ejercicio.

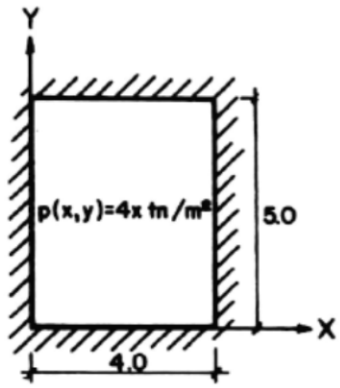
Éste ejemplo nos servirá en el punto 2 de este capítulo para realizar un análisis de convergencia de ambos métodos.

1.3. EJEMPLO 02

El segundo ejemplo se ha extraído del libro *citado en la* ***fuente [04].***

PROBLEMA Nº 7

Se considera la placa rectangular de 5 x 4 m. con el estado de sustentación y cargas que se indica (fig. 3.23)



(fig 3.23)

$D = 400$ tn.m

$\nu = 0$

$p(x,y) = 4 \times$ tn/m^2

CALCULAR :

1º) Flecha máxima y punto donde se produce

2º) Momentos flectores en el punto determinado en el apartado - anterior.

IV.07: *Enunciado del ejercicio 7 de la* ***fuente [04]***

En este caso la placa tiene unas dimensiones de *4*x*5* metros y está apoyada en sus 4 bordes. Tiene un parámetro nu de *ν=0* y una rigidez de *D=400 tn·m* y está sometida a una carga lineal en *x: p(x,y)=4·x tn/m²*.

La solución que se obtiene es la siguiente:

en $\begin{cases} x = 2 \\ y = 2{,}5 \end{cases}$ $W_{max}\,(x,y) = 31{,}681$ m.m.

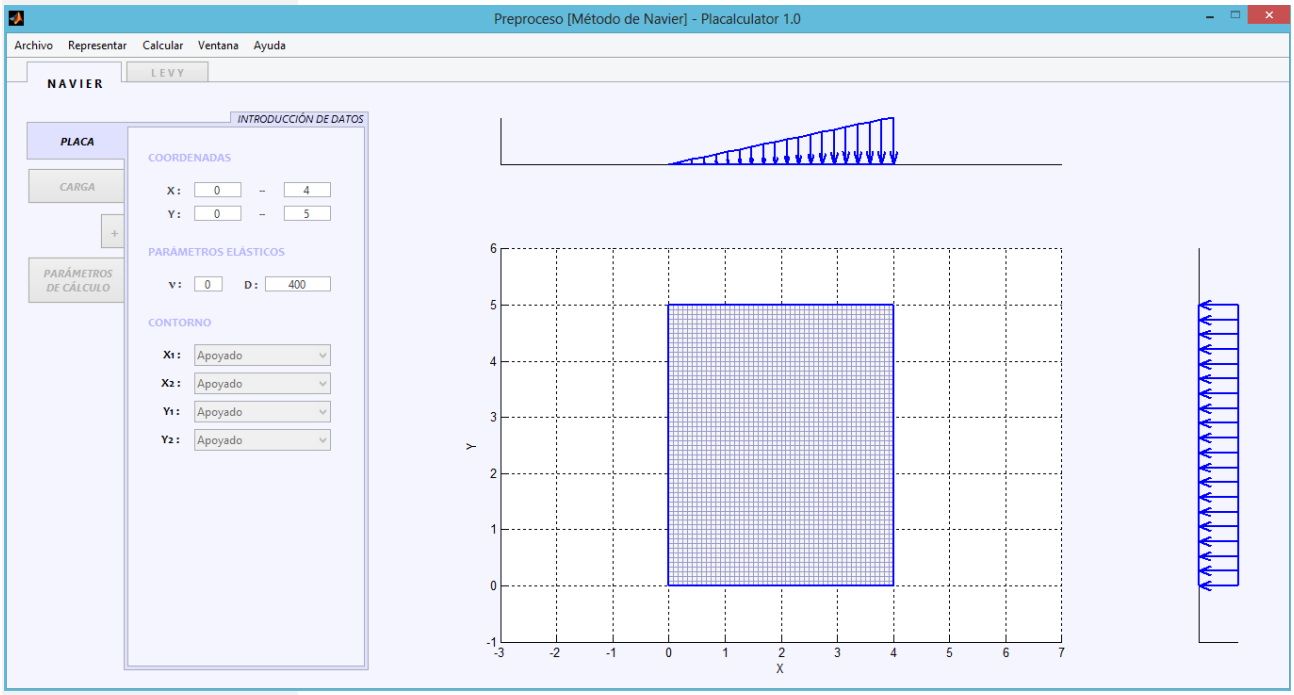
$M_x = 7{,}817$ m.tn/m

$M_y = 5{,}003$ m.tn/m

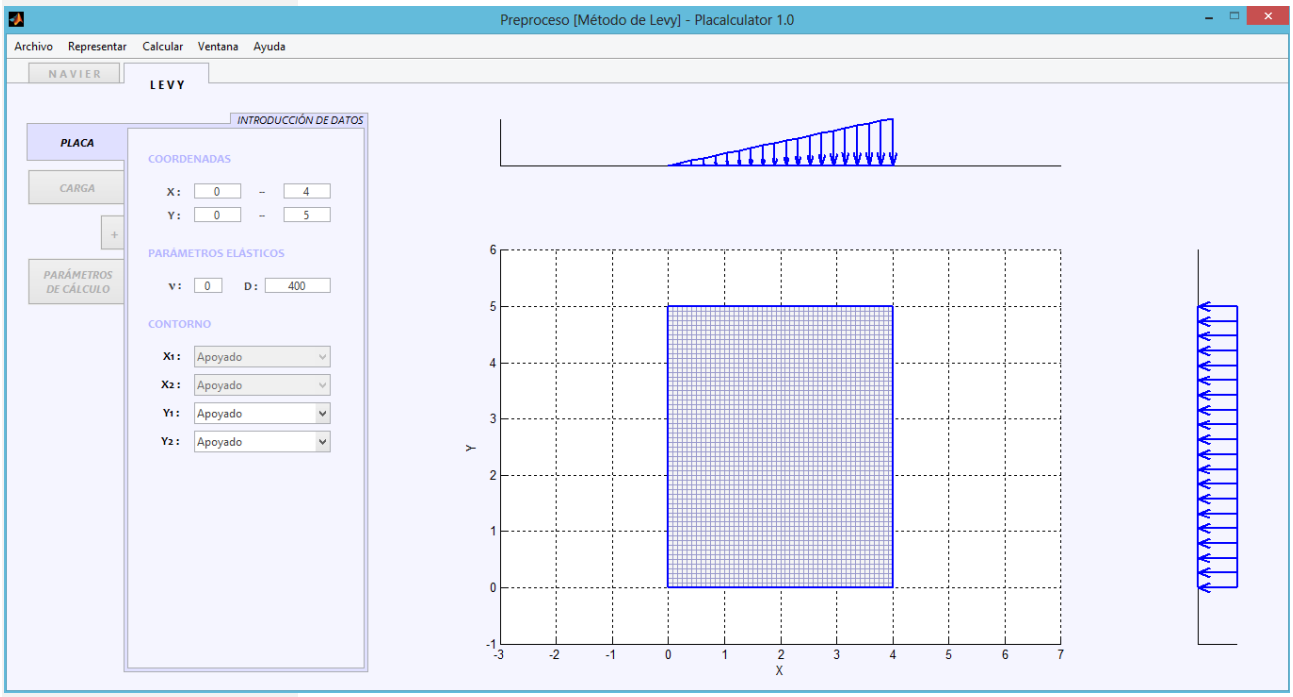
IV.08: *Solución del ejercicio 7 de la* ***fuente [04]***

Nuevamente, vamos a usar tanto el método de Navier como el de Levy para proceder al cálculo de este problema.

Insertamos los datos el nuestro software y obtenemos lo siguiente:

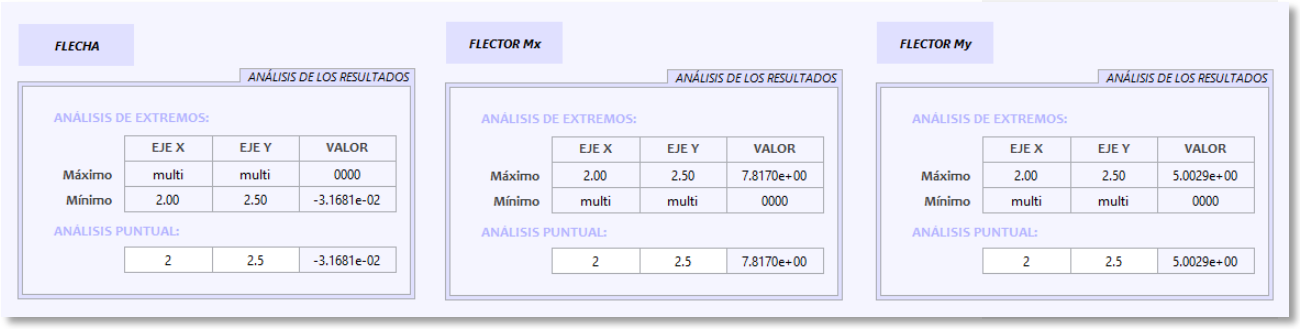


IV.09: *Preproceso del ejemplo 02. Método de Navier*

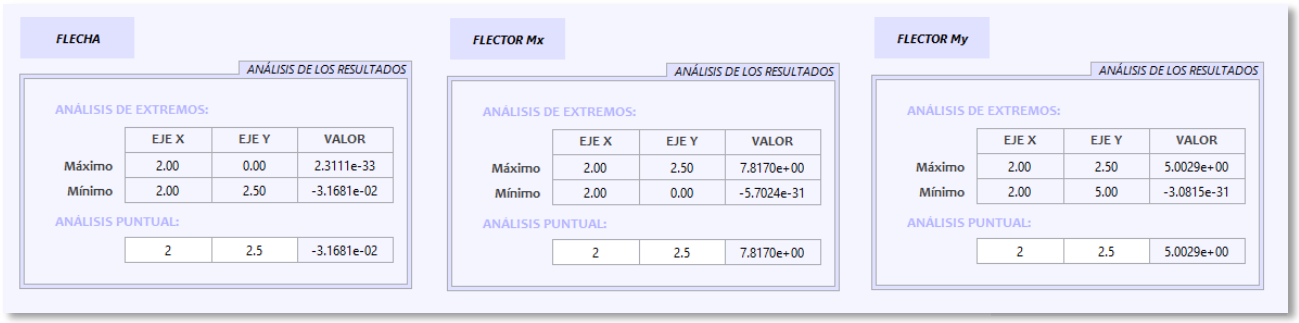


IV.10: *Preproceso del ejemplo 2. Método de Levy*

Ambos han desarrollado la carga y la flecha en la dos direcciones, *x* e *y*, en una sola serie de Fourier. Los resultados obtenidos se encuentran en la siguiente página:



IV.11: *Postproceso del ejemplo 02. Método de Navier*



IV.12: *Postproceso del ejemplo 02. Método de Levy*

Tanto el método de Navier como el de Levy nos dan exactamente los mismos resultados; la flecha máxima se encuentra en el centro de la placa, puesto que el desarrollo en series de Fourier sólo ha sumado una serie para cada dirección.

El valor de la flecha en este punto toma un valor de *31.681 mm* para ambos métodos. La coincidencia se mantiene al analizar los momentos flectores; *Mx* alcanza un valor de *7.8170 m·tn/m,* mientras que *My* se queda en *5.0029 m·tn/m*.

Podemos comparar los resultados de nuestro software con los datos aportados por la fuente. Veremos que todos y cada uno de ellos coinciden a la perfección.

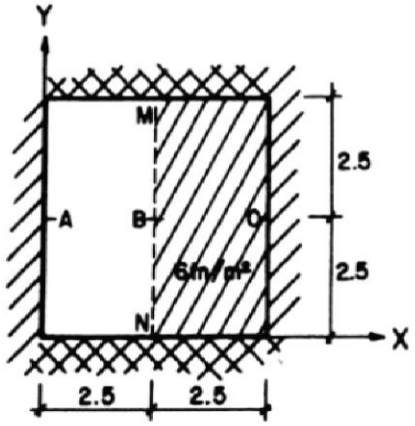
NOTA: *Es posible que, en ciertos casos, el Método de Levy dé falsos máximos o falsos mínimos en puntos concretos del contorno y1 o y2. Este hecho es debido a la limitación física que tiene Matlab con la precisión de números reales. Se puede observar, sin embargo, que el valor de estos falsos positivos se encuentra extremadamente cerca del 0. Hay que tener en cuenta que estos valores no son más de lo que son; falsos positivos que debemos despreciar.*

1.4. EJEMPLO 03

Éste ejemplo solo se puede resolver con el método de Levy, puesto que se trata de una placa apoyada por dos extremos pero empotrada en los otros dos.

PROBLEMA Nº 9

Se considera la placa rectangular 5 x 5 m., con el estado de cargas y sustentación según se indica (fig. 3.26).



(fig 3.26)

$E = 2.10^6\ Tn/m^2$

$\nu = 0$

$\delta = 0{,}10$ m.

$D = \frac{500}{3}$ Tn.m.

La carga de 6 Tn/m^2 solo actúa en la zona sombreada.

CALCULAR : La flecha en el centro de la placa

IV.13: *Enunciado del ejercicio 9 de la* ***fuente [04]***

Como podemos leer en la imagen de aquí arriba, se trata de una placa de rigidez *D=500/3 Tn·m* y de parámetro nu *γ=0*. Ésta se encuentra bajo una carga de valor *6 Tn/m²* repartida sobre su mitad derecha. Las dimensiones son *5*x*5m*.

El resultado es el siguiente:

La flecha TOTAL en B será :

$$(W_B)_{Total} = W_{(S)} + W_{(A)} = 2{,}207 + 0 = 2{,}207 \text{ cm.}$$

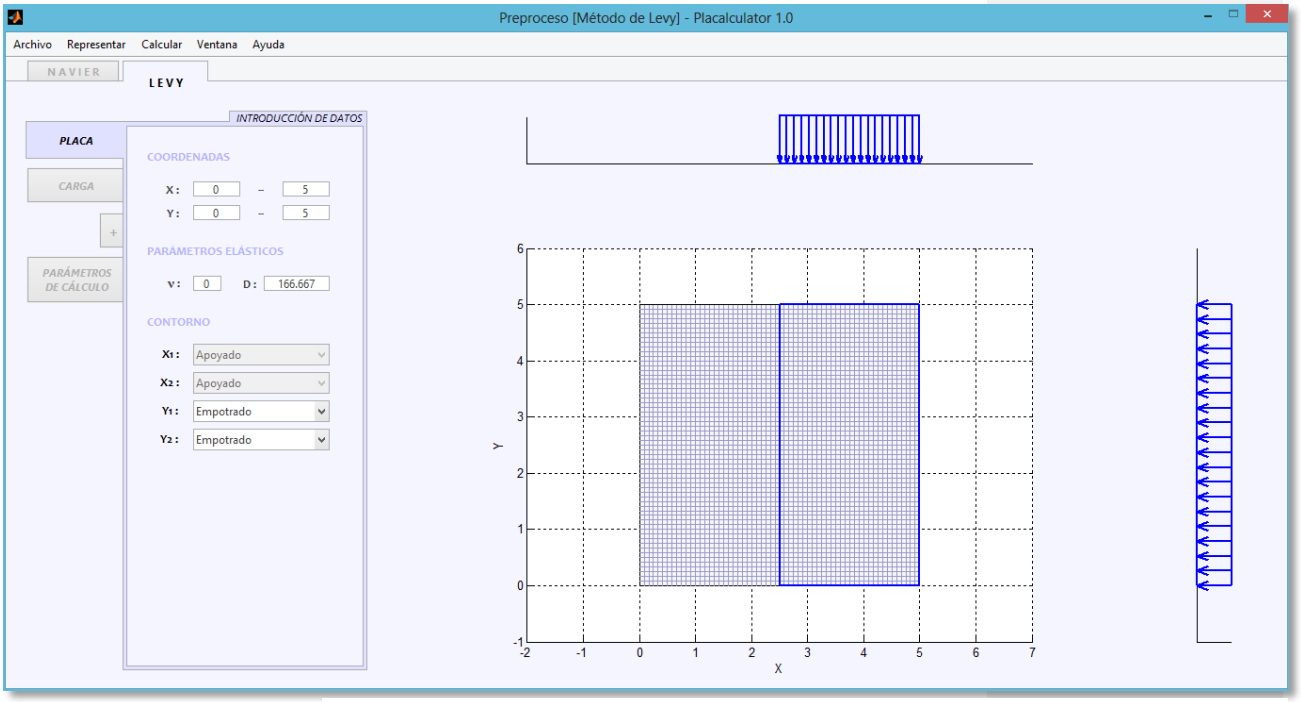
IV.14: *Solución del ejercicio 9 de la* ***fuente [04]***

Éste resultado se ha encontrado aplicando el método de Levy con desarrollo en una serie de Fourier de la carga y la flecha en la dirección *x*. Sin embargo, nuestro software desarrolla siempre en las dos direcciones. Por tanto, deberemos imponer un desarrollo

en dirección *x* de una sola suma de Fourier, mientras que en la dirección *y* necesitaremos cuantas más mejor (en teoría infinitas).

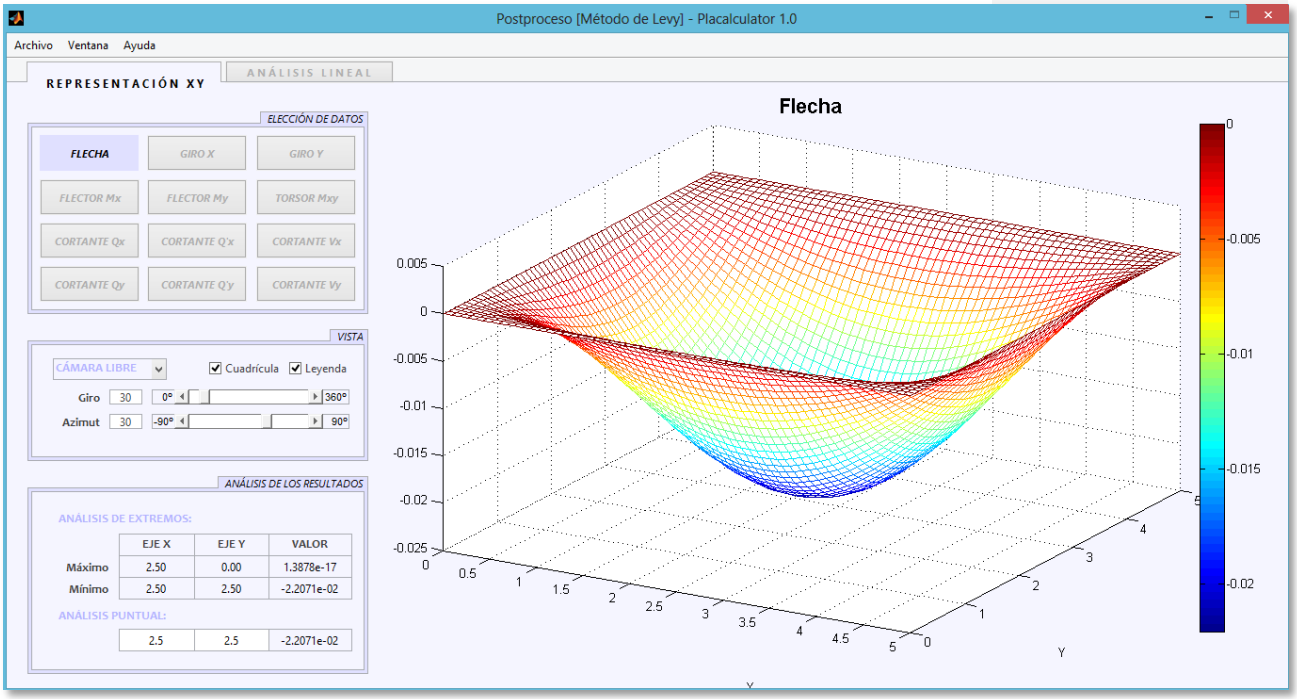
En este caso hemos creído suficiente tomar 100 sumas para este eje.

El preproceso, pues, queda de la siguiente forma:



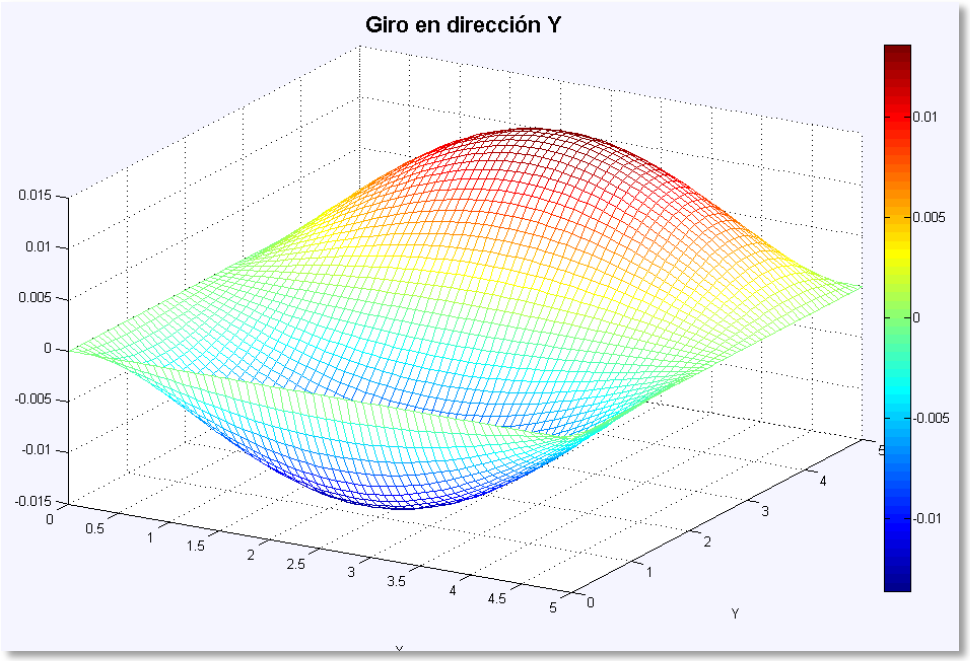
IV.15: *Preproceso del ejemplo 3. Método de Levy*

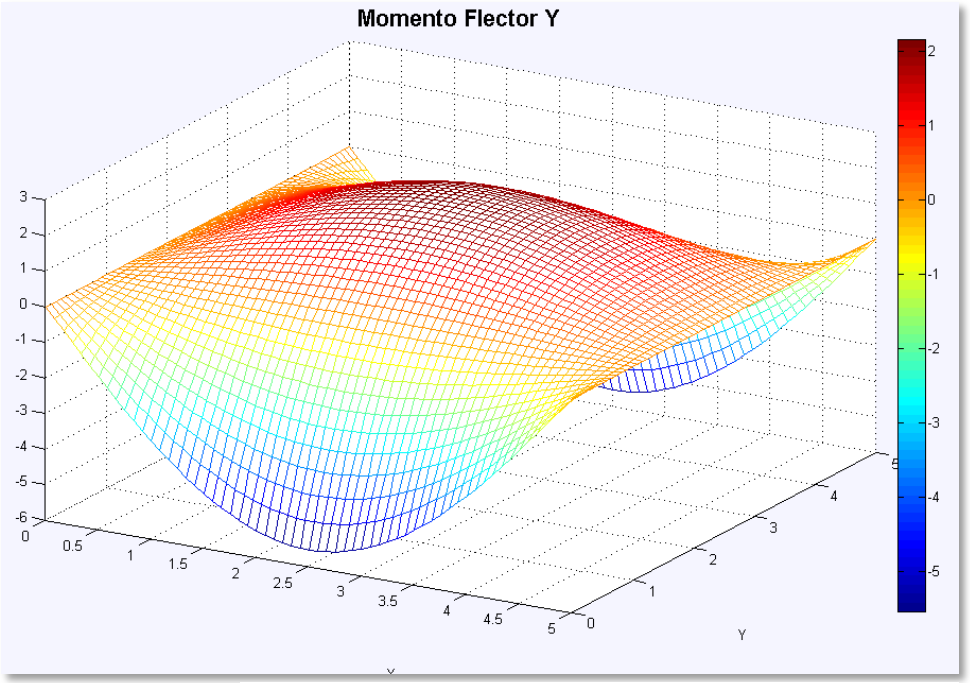
Con él se obtienen los siguientes resultados:



IV.16: *Postproceso del ejemplo 3. Método de Levy*

Éste es el primer ejemplo de placa empotrada. En él podemos observar que la flecha y el giro Y de los empotramientos valen cero, mientras que el momento Y no se anula; y que por tanto se cumplen las condiciones de contorno.





IV.17: *Giro Y y flector Y en el ejemplo 3. Método de Levy*

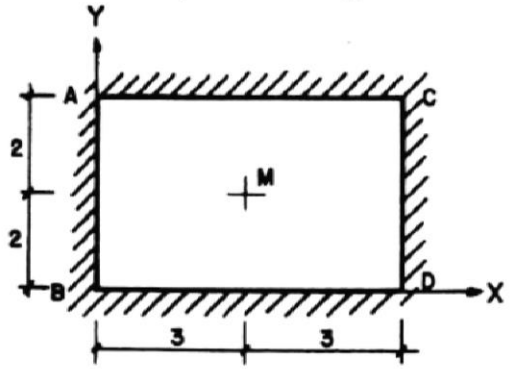
Se observa en el centro de la placa una flecha de *2.2071 cm.* El la bibliografía original la placa sufría en su centro una flecha de valor *2.207 cm.* Por lo tanto podemos afirmar que los dos resultados, una vez más, vuelven a ser idénticos.

1.5. EJEMPLO 04

El último ejemplo se ha extraído también del libro citado en la ***fuente [04]***.

PROBLEMA Nº 8

Se considera la placa de 6 x 4 m., apoyada simplemente en sus cuatro bordes, sobre la que actua una carga de 48 tn (fig. 3.24)



$E = 2 \times 10^6 \ Tn/m^2$

$\nu = 0$

$\delta = 10 \ cm$

$D = \frac{500}{3} \ Tn \times m$

(fig 3.24)

CALCULAR :

1º) Flecha máxima, suponiendo que la carga de 48 Tn actúa uniformemente distribuida en toda la placa:

2º) Flecha máxima, suponiendo que la carga de 48 Tn actúa puntual en M.

IV.18: *Enunciado del problema 8 de la* ***fuente [04]***

En este caso la placa tiene unas dimensiones de *6x4 metros* y está apoyada en sus 4 bordes. Tiene un parámetro nu de $\gamma=0$ y una rigidez de $D=500/3$ *tn·m* y está sometida a una carga de 48 toneladas.

Es un problema muy interesante, puesto que invita a comparar el efecto que tiene una placa sometida bajo acciones repartidas o acciones puntuales.

Separaremos el ejercicio en dos apartados:

- El apartado *a)* donde calcularemos el efecto de una **carga repartida** mediante Navier y mediante Levy

- El apartado *b)* donde calcularemos el efecto de una **carga puntual** en el centro de la placa, también con los dos métodos.

a) *CARGA REPARTIDA*

El libro nos proporciona la siguiente solución:

la flecha máxima, aplicando la expresión (1) para x = 3, y = 2, será :

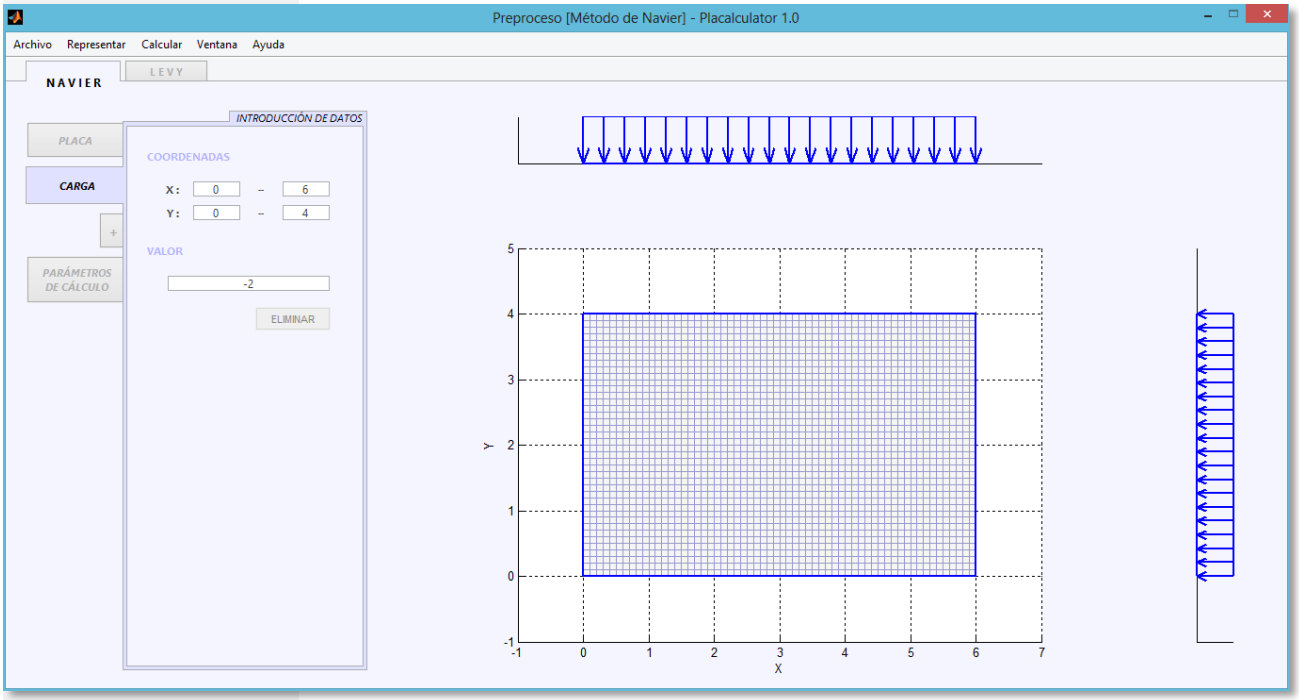
$$W_{max}(x,y) = 2{,}450 \text{ cm.}$$

IV.19: *Solución del ejercicio 8 a) de la* ***fuente [04]***

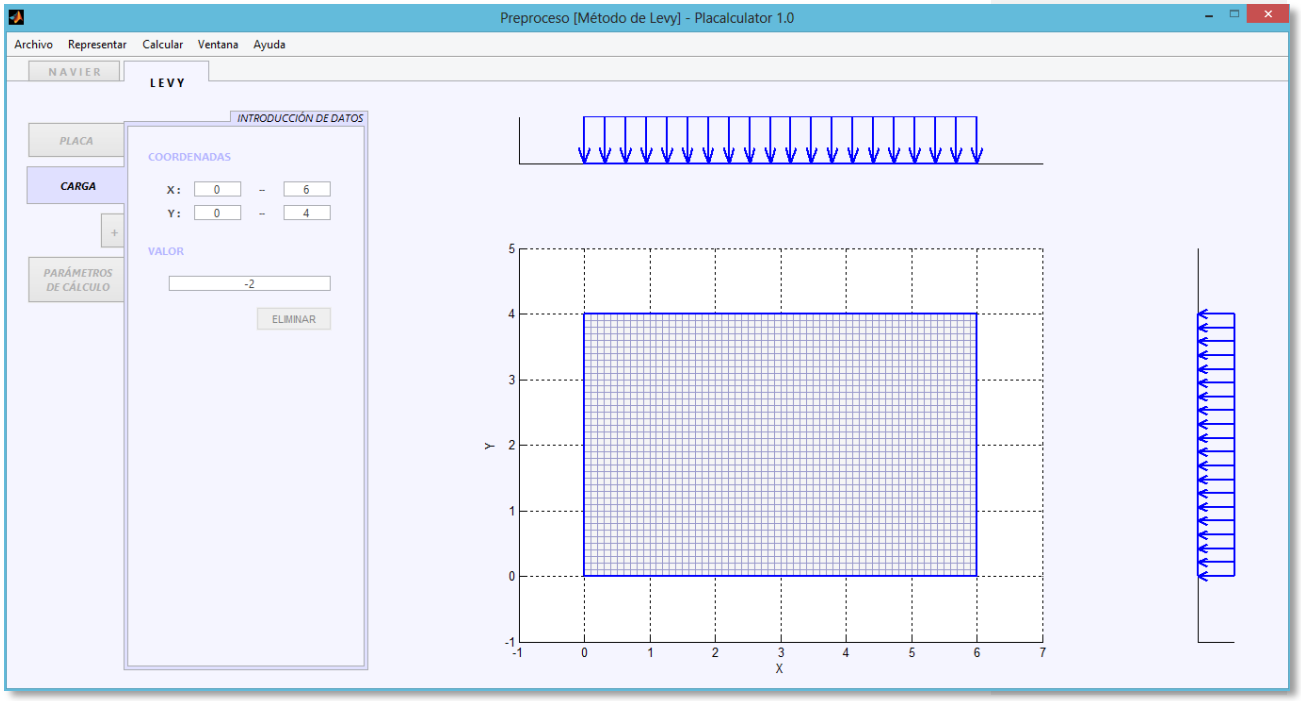
Esta solución se ha encontrado con el método de Navier, es decir, desarrollando la carga y la flecha en una sola suma de Fourier para cada dirección.

Para resolverlo con nuestro software, lo primero que hay que hacer es calcular la carga repartida uniformemente en toda la placa. Simplemente se divide la carga de *48 ton* en *6x4=24m²*. Por tanto se obtienen *2 ton/m²*.

Una vez hecho este pequeño cálculo, nos disponemos a rellenar toda la pantalla de preproceso, en una primera instancia mediante el método de Navier y más tarde con el método de Levy:



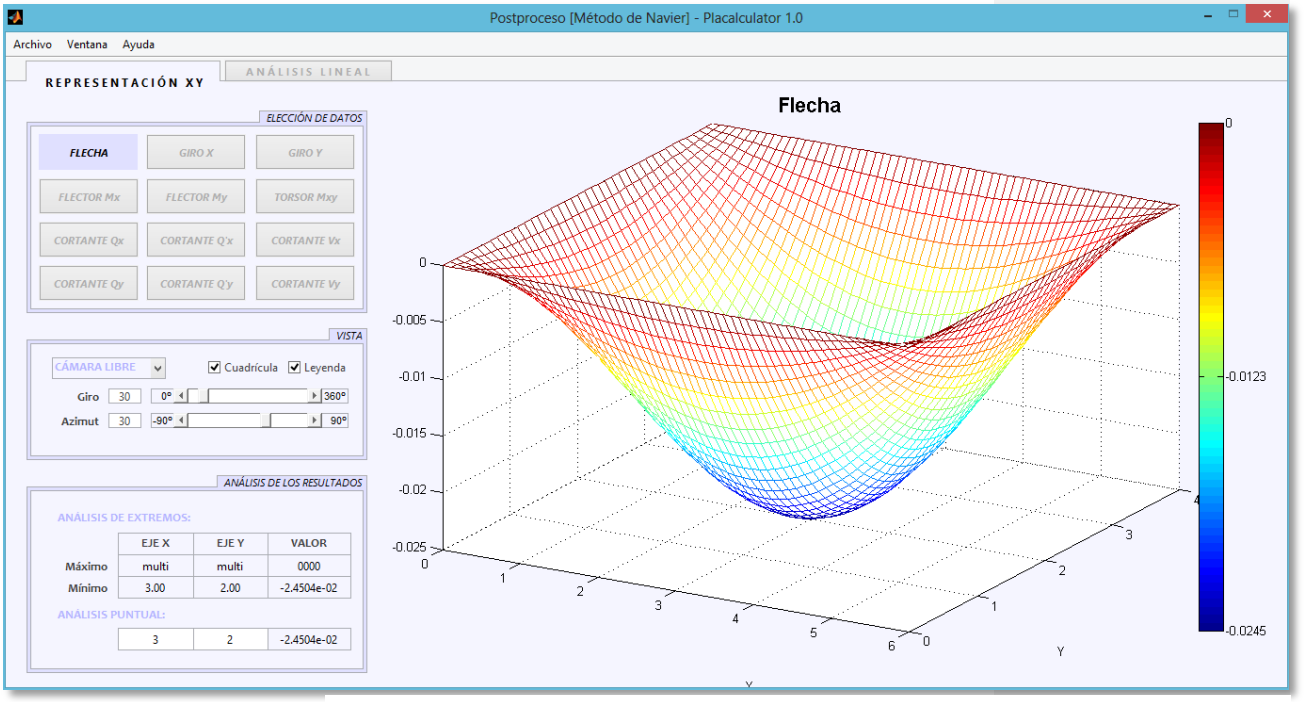
IV.20: *Preproceso del ejemplo 4 a). Método de Navier*



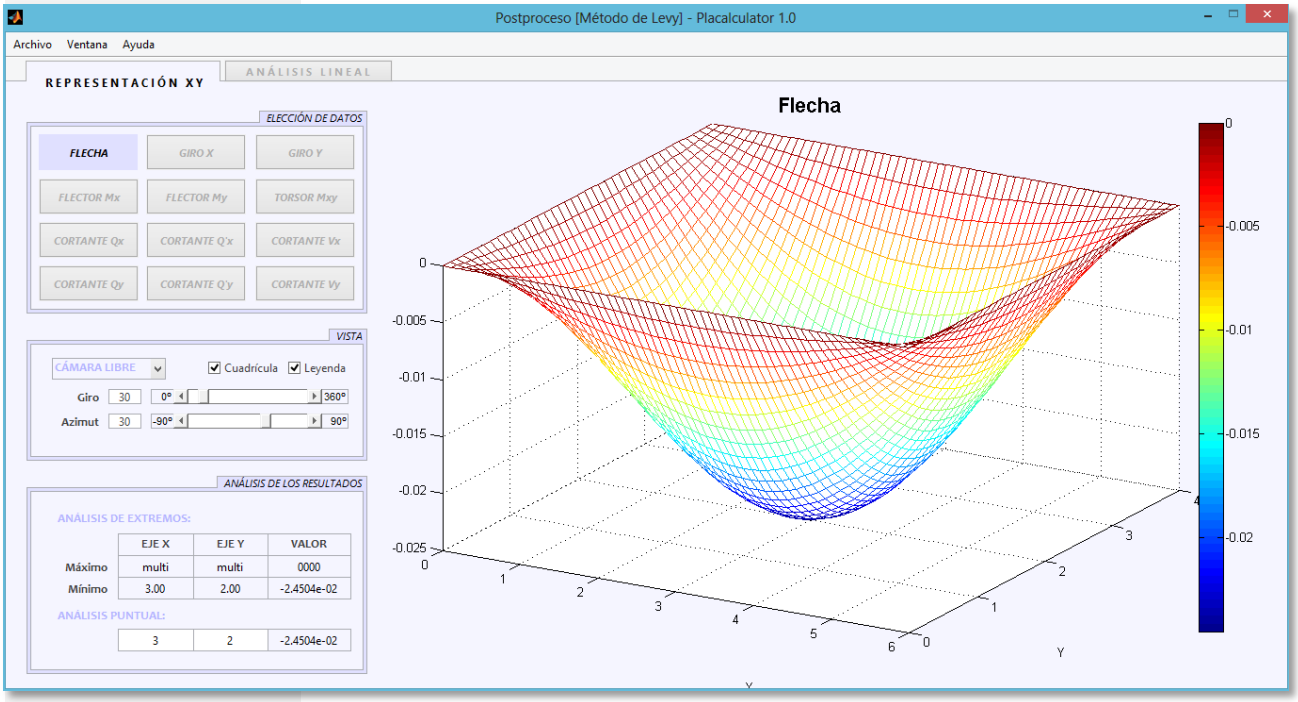
IV.21: *Preproceso del ejemplo 4 a). Método de Levy*

En ambos casos se ha utilizado, para las dos direcciones, el desarrollo en serie de Fourier con un único término en el sumatorio.

El resultado viene a continuación:



22: *Postproceso del ejemplo 4 a). Método de Navier*



IV.22: *Postproceso del ejemplo 4 a). Método de Levy*

En ambos casos se observa que la flecha en el centro de la placa vale *2.4504 cm*. Este valor es muy parecido al que se de en el libro: *2.450 cm*.

Podemos afirmar por tanto que ambos métodos nos vuelven a dar resultados correctos.

b) *CARGA PUNTUAL*

El libro nos ofrece la siguiente solución:

Luego la flecha máxima aplicando **(1)** para x = 3, y = 2 será :

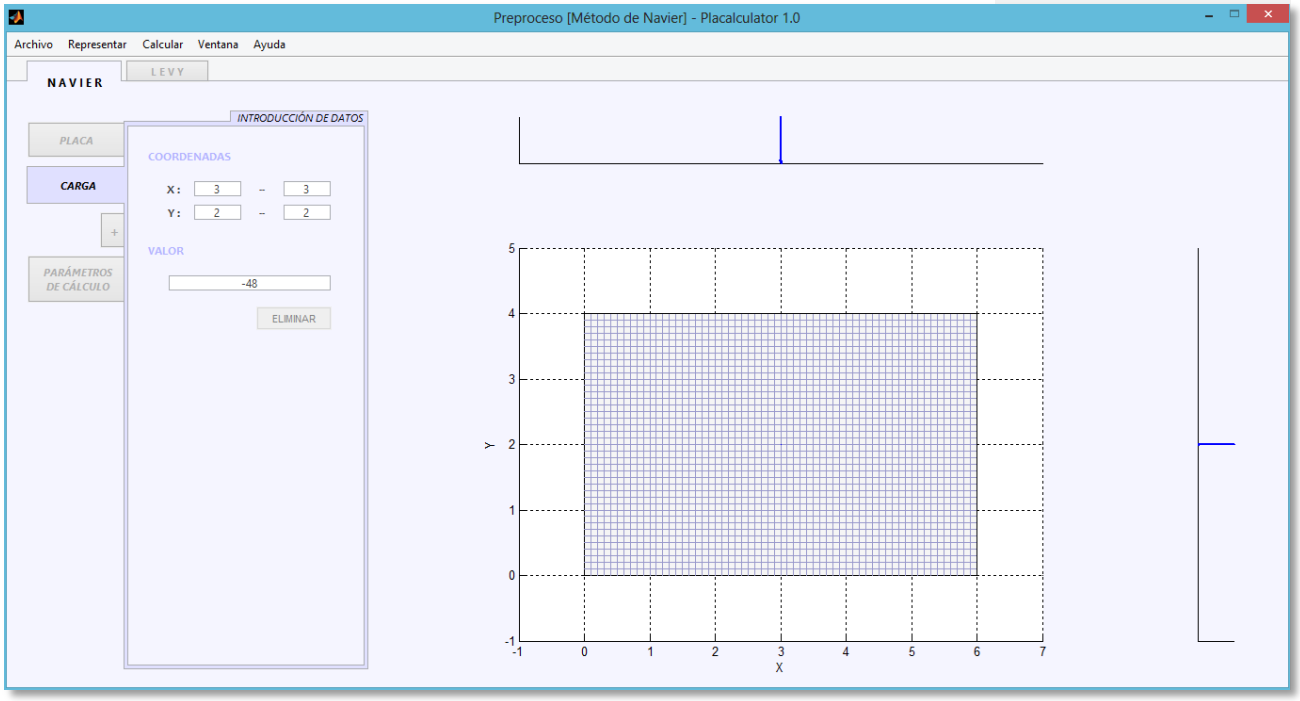
$$W_{max}(x,y) = 6{,}046 \text{ cm.}$$

NOTA.- Evidentemente al concentrar la carga, la flecha es mayor

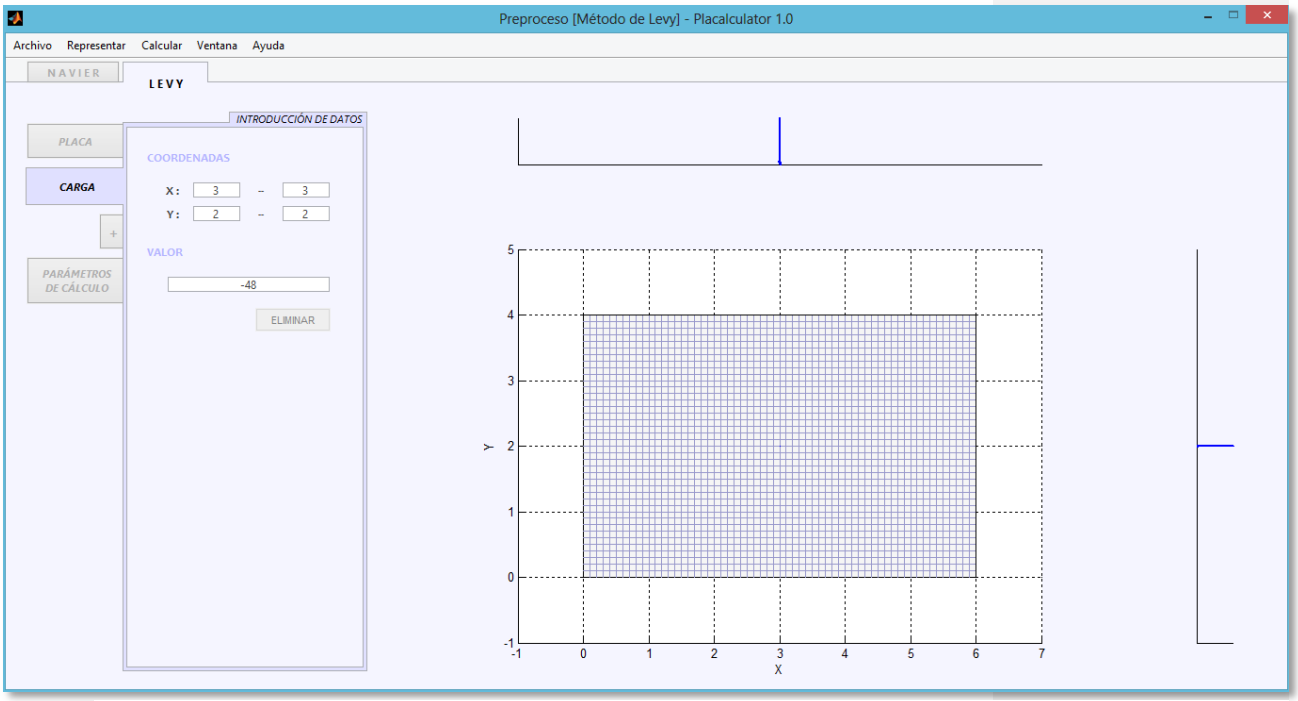
IV.23: *Solución del ejercicio 8 a) de la* ***fuente [04]***

En este caso, al ser la fuerza puntual, debemos entrarla en unidades de fuerza, no de fuerza por unidad de superficie como hemos venido haciendo hasta ahora. Es por eso que entramos en el preproceso directamente las *48 toneladas.*

Las imágenes siguientes lo ilustran:

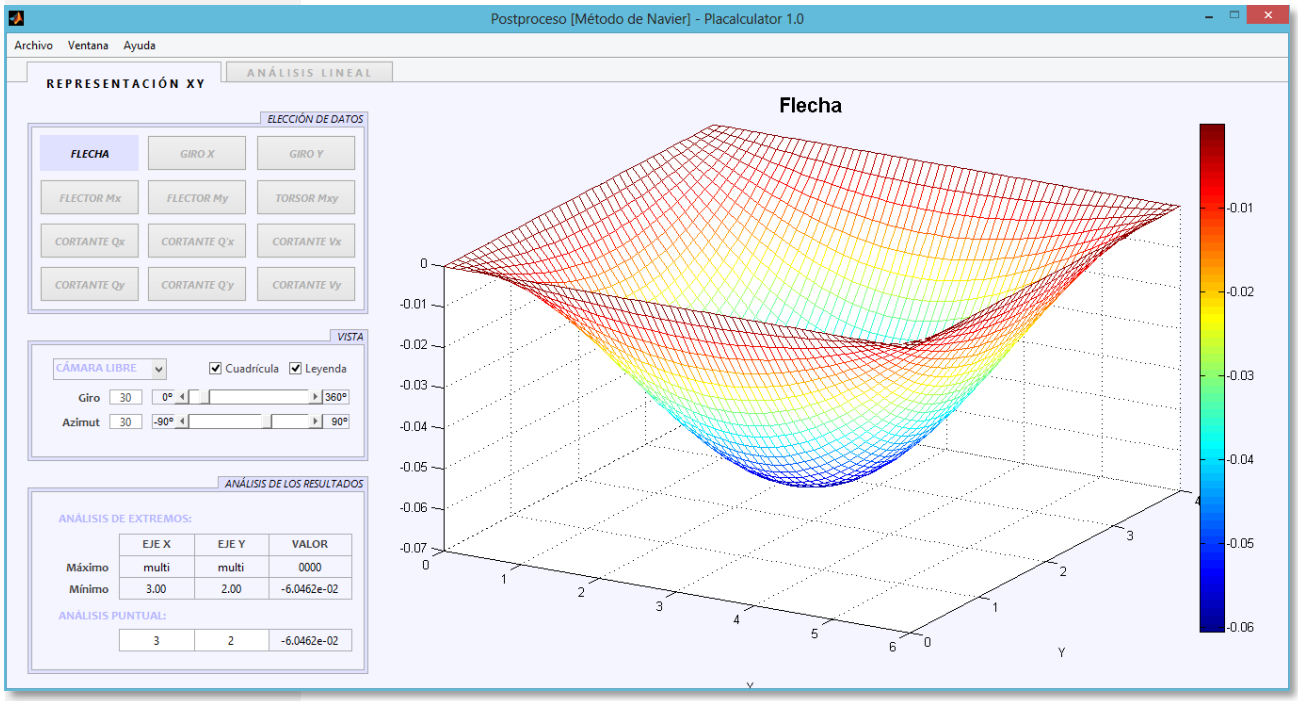


IV.24: *Preproceso del ejemplo 4 b). Método de Navier*

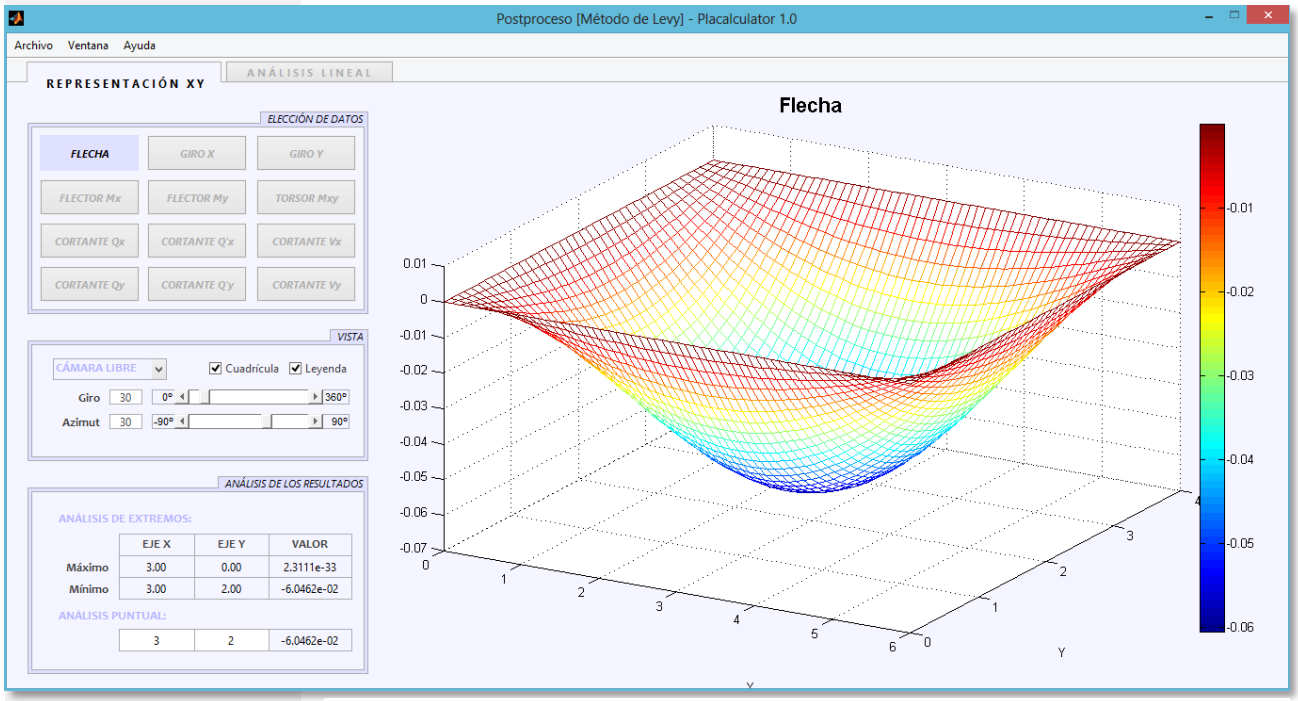


IV.25: *Preproceso del ejemplo 4 b). Método de Levy*

Al igual que en el apartado *a)*, se ha usado una única suma de Fourier en cada dirección. Tras calcular ambas placas, los resultados son los siguientes:



IV.26: *Postproceso del ejemplo 4 b). Método de Navier*



IV.27: *Postproceso del ejemplo 4 b). Método de Levy*

Ambos métodos presentan una flecha en centro de *6.0462cm*, mientras que la calculada a mano por el autor del libro vale *6.046cm*. Evidentemente, y una vez más, nos encontramos con resultados iguales y coherentes.

Finalmente, podemos concluir que la flecha que provoca la carga concentrada es mayor que la que provocaría la carga repartida.

2. ANÁLISIS DE CONVERGENCIA

2.1. INTRODUCCIÓN

Tal y como hemos visto en el punto 1 de este capítulo, se ha comprobado la bondad de los cálculos y se ha validado el software comparando la solución obtenida con cálculos externos realizados por otros autores.

En este punto queremos comprobar, por una parte, que las soluciones pueden ser cada vez más exactas si aumentamos la precisión del cálculo (esto es, el número de sumas de Fourier que forman las series en las que descomponemos las cargas y las flechas). Debemos observar un comportamiento convergente hacia la solución final a medida que añadimos sumas.

Por otra parte, se quiere comprobar que el cálculo con cargas lineales o puntuales se basa en el problema con carga repartida típico, haciendo tender la(s) dimensión(es) correspondiente(s) a cero. Debemos observar un comportamiento convergente hacia la solución que obtenemos aplicando cargas puntuales o lineales a medida que reducimos la superficie de aplicación de la carga repartida.

Es por eso que este apartado se divide en dos bloques:

- 2.2. Convergencia por adición de sumas en las series de Fourier
- 2.3. Convergencia por reducción del área de aplicación de carga

2.2. CONVERGENCIA POR ADICIÓN DE SUMAS EN LAS SERIES DE FOURIER

En este apartado retomamos el ejemplo 01 del apartado 1.2. para hacer en él un análisis de convergencia con el método de Navier (con el método de Levy se habrían obtenido los mismos resultados, al estar programado también con doble desarrollo).

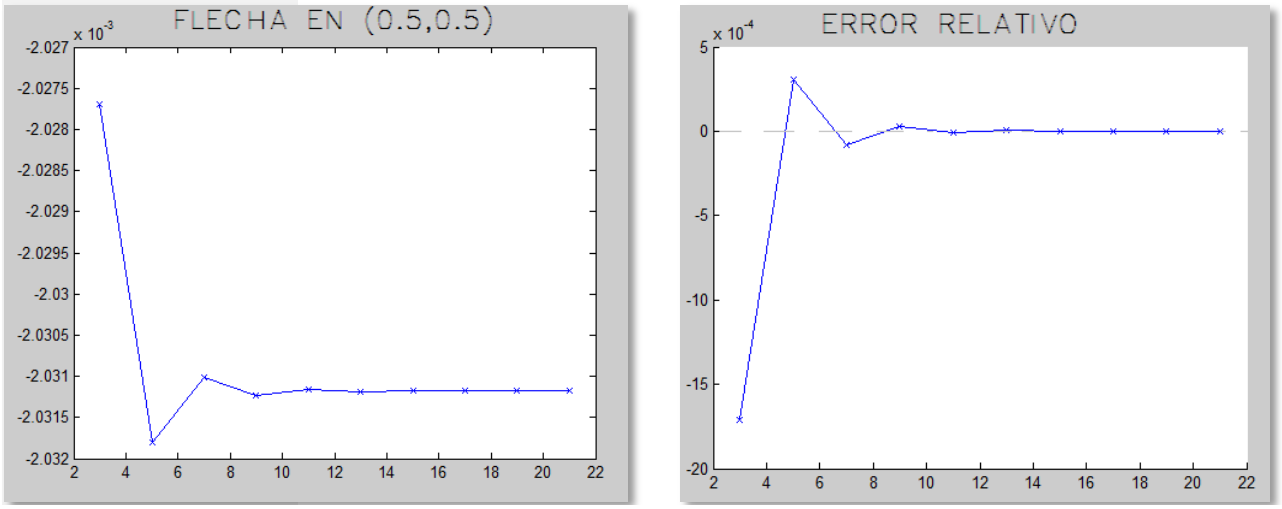
La estrategia que hemos seguido es la siguiente: a cada iteración se ha ido aumentando al mismo ritmo el número de sumas en las series de Fourier, tanto para la dirección *x* como para la dirección *y*. En cada iteración hemos calculado el valor de la flecha en el centro de la placa.

El valor exacto se alcanza con una serie de infinitos términos. Obviamente no se puede alcanzar tal valor. Para comparar los resultados con una solución final, hemos tomando como referencia el valor que se obtiene al hacer 31 sumas para cada dirección. Los resultados son los siguientes:

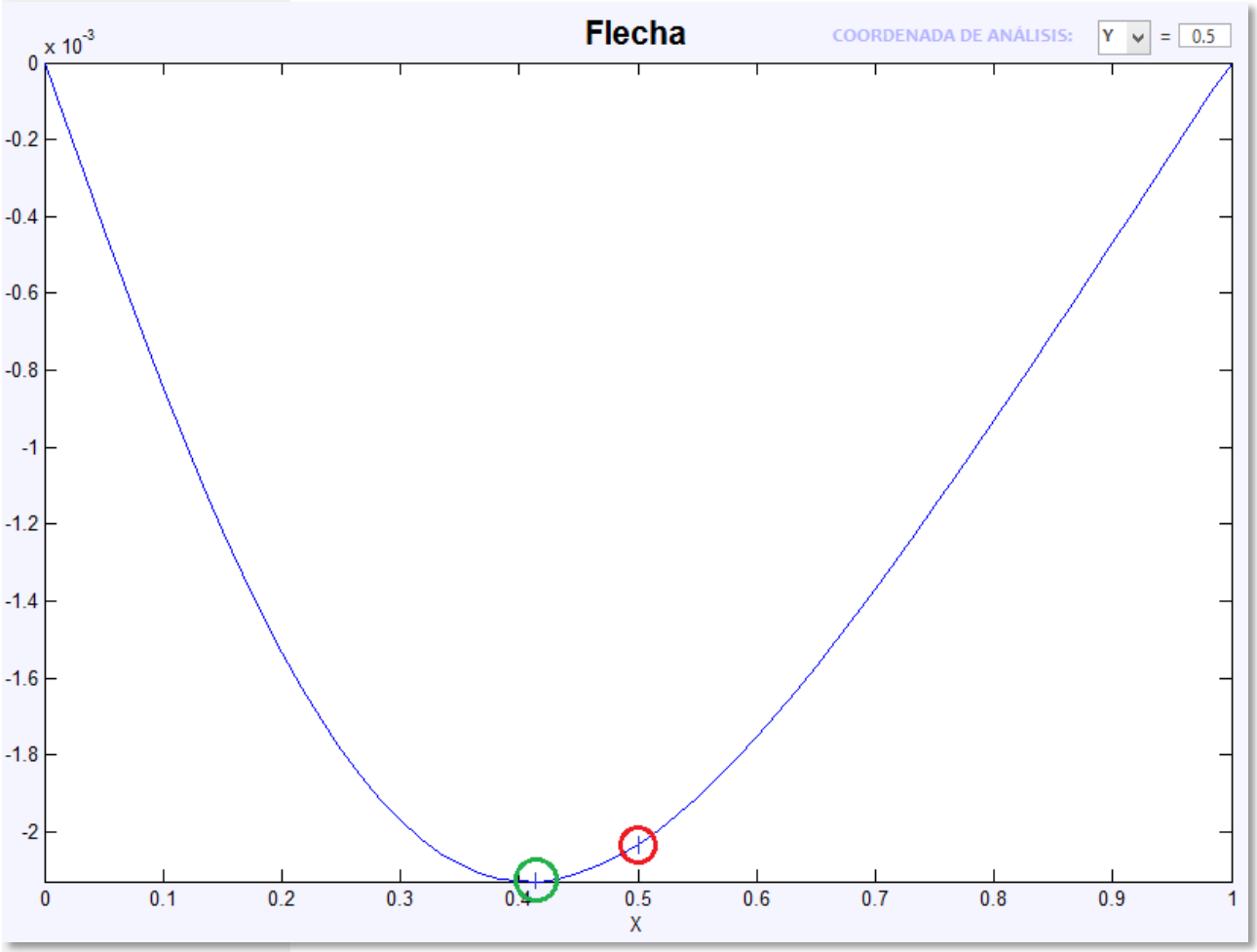
SUMAS EN *X*	**SUMAS EN *Y***	**FLECHA w(0.5,0.5)**	**ERROR RELATIVO [-]**
1	1	-0.002080322946592	0.024196231345338
3	3	-0.002027701334609	-0.001710735055526
5	5	-0.002031798536665	0.000306422382308
7	7	-0.002031013964275	-0.000079842685993
9	9	-0.002031234076247	0.000028524071469
11	11	-0.002031152605147	-0.000011586236270
13	13	-0.002031187712107	0.000005697818682
15	15	-0.002031170463301	-0.000002794209986
17	17	-0.002031179654193	0.000001730701382
19	19	-0.002031174368397	-0.000000871631276
21	21	-0.002031177565435	0.000000702352358
…	…	…	…
31	31	-0.002031176138834	0

IV.28: *Análisis de convergencia de la flecha en el centro de la placa. Tabla recopilatoria*

Se puede apreciar que la flecha en el centro de la placa tiende hacia un cierto valor a media que aumentamos el número de iteraciones. Es por eso que podemos afirmar que **la solución converge** hacia el valor correcto. También observamos que el máximo valor de la flecha ya no se encuentra en el centro de la placa, sino que se encuentra desplazado a la izquierda (como era de esperar), en el punto (0.42, 0.5).



IV.29:*Convergencia de la flecha en el centro de la placa y error relativo. Representación gráfica*



IV.30: *Flecha pico desplazada hacia el punto (0.42 ,0.5) tras añadir términos en las series de Fourier*

2.3. CONVERGENCIA POR REDUCCIÓN DEL ÁREA DE APLICACIÓN DE LA CARGA

En este apartado retomamos el ejemplo 04 del apartado 1.5. para hacer en él un análisis de convergencia con el método de Levy (con el método de Navier se habrían obtenido los mismos resultados, al estar programado también con doble desarrollo).

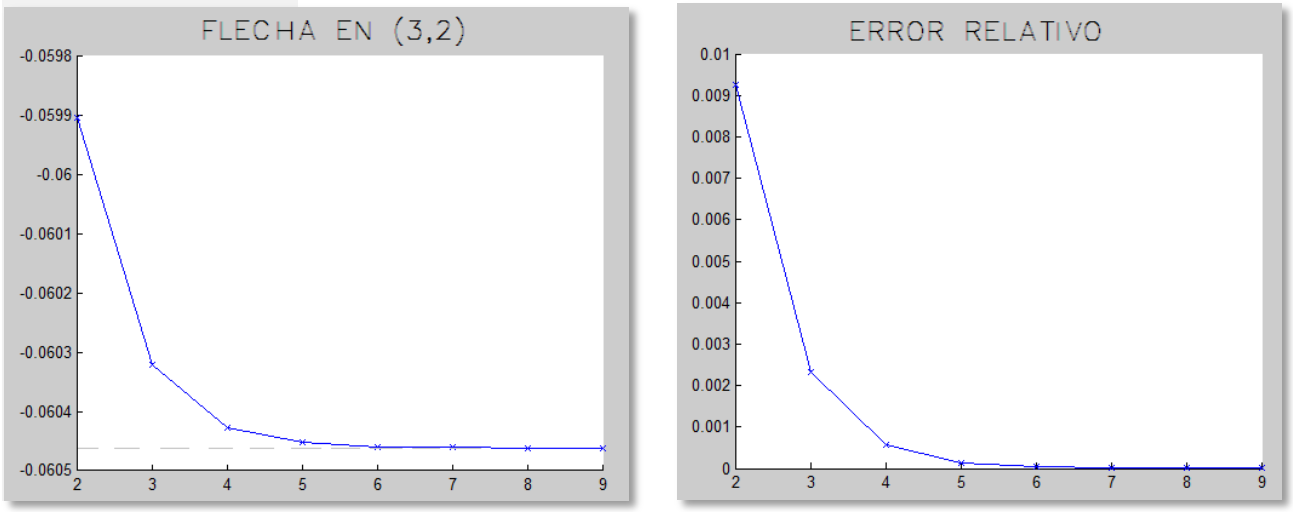
La estrategia que hemos seguido es la siguiente: se ha empezado el cálculo con un área de repartición de la carga de *1x1m*. Esta área se ha ido aminorando en cada iteración en un cuarto, de manera que el lado del cuadrado cargado se ha ido reduciendo a la mitad. La carga en cada iteración debe ser de *48toneladas*, por lo que se empieza aplicando una carga de *48ton/m²*; a cada iteración se debe aumentar la carga repartida a la misma razón que se aminora el área. Es decir, se ha ido multiplicando por 4 el valor de la carga repartida.

Lo que esperamos encontrar es una convergencia de la flecha en el centro de la placa hacia el valor obtenido en el apartado 1.5. de *6.046cm*. Ése apartado se realizó mediante una suma de Fourier en cada dirección, de manera que en éste vamos a proceder de la misma manera. Los resultados se pueden observar en la siguiente tabla:

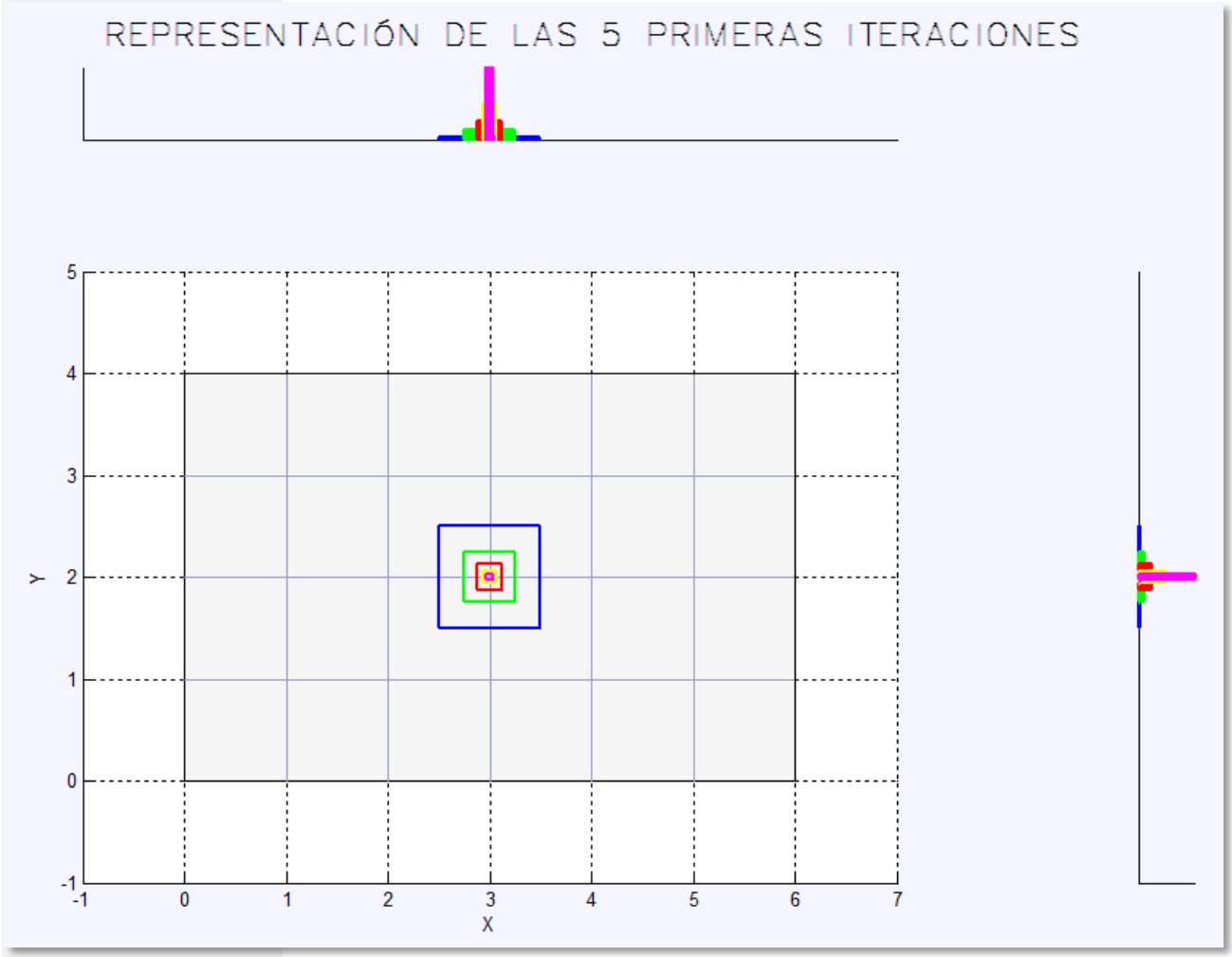
ÁREA [m²]	CARGA[ton/m²]	FLECHA w(3,2) [m]	ERROR RELATIVO [-]
1	48	-0.058248740130920	0.036598365491308
1/4	48·4	-0.059902373523794	0.009248192594997
1/8	48·8	-0.060321368173213	0.002318255068560
1/16	48·16	-0.060426468618376	0.000579952395741
1/32	48·32	-0.060452765757522	0.000145012399114
1/64	48·64	-0.060459341419025	0.000036254629840
1/128	48·128	-0.060460985421342	0.000009063749553
1/256	48·256	-0.060461396427131	0.000002265943324
1/512	48·512	-0.060461499178920	0.000000566486106
…	…	…	…
Carga Puntual de *48ton*		-0.060461533429539	0

IV.31:*Análisis de convergencia de la flecha en el centro de la placa. Tabla recopilatoria*

Se puede apreciar que la flecha en el centro de la placa tiende hacia el valor encontrado anteriormente a medida que aumentamos el número de iteraciones. Es por eso que podemos afirmar que **la solución converge.**



IV.32: *Convergencia de la flecha [m] en el centro de la placa y error relativo. Representación gráfica*



IV.33: *Representación gráfica de las 5 primeras iteraciones en su preproceso*

En este apartado no hay que confundir, sin embargo, la convergencia de los resultados con su veracidad. El valor encontrado no es el exacto, puesto que no se ha desarrollado la carga en un número de series de Fourier suficiente.

CONCLUSIONES

Como conclusión de este trabajo final de grado, puedo decir que en este documento se han respondido a las cuatro preguntas siguientes:

¿**Qué** es una placa? – CAPÍTULO I – Teoría de placas

¿**Cómo** se usa el programa? – CAPÍTULO II – Manual de usuario

¿**Por qué** funciona el programa? – CAPÍTULO III – Programación

¿**Qué** fiabilidad tiene dicho programa? – CAPÍTULO IV – Análisis de resultados

Mediante este trabajo he conseguido diseñar y programar un *software* de cálculo de placas delgadas:

FIABLE. Así lo demuestran los diferentes ejemplos de validación en el *apartado 1* del *capítulo IV*, donde se ha podido comprobar que los resultados obtenidos son coincidentes en ambos métodos de resolución, Navier y Levy, con ejemplos extraídos de la bibliografía.

REFERENCIADO: Basado en teorías clásicas de deformaciones (ver *capítulo I*) y tensiones de placas y resueltos por métodos directos bien conocidos, extraídos de diversas fuentes (ver *Bibliografía*)

ÚTIL: Obtenemos automáticamente la localización de los puntos de más interés desde el punto de vista ingenieril, como pueden ser el máximo o el mínimo de deformaciones, giros, momentos flectores y torsores, esfuerzos cortantes… y su valor. Parámetros básicos en el diseño de placas.

VISUAL: Con una interfaz gráfica con la que el usuario visualiza tanto la entrada de los datos como los resultados obtenidos. Se ha hecho énfasis en el juego que tiene la representación gráfica de resultados, tanto por el hecho de poder mover el punto de vista allí donde se quiera como por el hecho de poder hacer un estudio de los resultados sección por sección.

CÓMODO: Fácil de manejar y de navegar por sus pantallas, con una interfaz simple. Con las opciones de cargar y guardar datos que le dan mucho más juego. Además cuenta con un manual de usuario que explica detalladamente todo lo que un usuario nuevo debe saber para usar el *software* correctamente, así como con ejemplos ya resueltos que pueden cargarse y visualizarse.

ROBUSTO: No se cuelga frente a posibles errores, sino que avisa al usuario de dónde puede estar el error y lo devuelve a la pantalla anterior para que lo pueda corregir.

En definitiva, una buena herramienta que puede ser usada perfectamente por personal docente para completar la formación de los alumnos en éste ámbito concreto de la resistencia de materiales.

También podría ser usada para verificar el cálculo de algún otro *software* que use algún otro tipo de método para calcular el mismo tipo de placas, como por ejemplo, mediante métodos numéricos como la discretización en elementos finitos, o por el método de las bandas finitas o de las diferencias finitas.

Como conclusión final, puedo decir que se han alcanzado los objetivos planteados en el inicio.

ANEJOS

1. TEORÍA CLÁSICA DE LA ELASTICIDAD.

TENSIÓN PLANA

*Ver **fuentes [05], [09]-[11]***

1.1. RELACIÓN TENSIÓN - DEFORMACIÓN

La ecuación constitutiva para materiales elásticos es la siguiente:

(VI.01)

$$\underline{\underline{\sigma}} = \underline{\underline{\underline{\underline{C}}}} \cdot \underline{\underline{\varepsilon}}$$

$\boldsymbol{\sigma}$ y $\boldsymbol{\varepsilon}$ son matrices cuadradas de dimensiones 3x3, y $\boldsymbol{C}$ el tensor de orden 4 y dimensiones 3x3x3x3 que los relaciona.

La relación entre $\boldsymbol{\sigma}$ y $\boldsymbol{\varepsilon}$ se va a encontrar, pues, encontrando los 81 coeficientes del tensor $\boldsymbol{C}$.

Sin embargo, considerando propiedades de isotropía y homogeneidad del material, se puede demostrar que la relación *(VI.01)* se simplifica enormemente y se transforma en la siguiente:

(VI.02)

$$\underline{\underline{\sigma}} = \lambda \cdot Tr\left(\underline{\underline{\varepsilon}}\right) \cdot \underline{\underline{1}} + 2 \cdot \mu \cdot \underline{\underline{\varepsilon}}$$

Donde $Tr(\cdot)$ es el operador traza (suma de las componentes de la diagonal), y $\underline{1}$ representa la matriz identidad 3x3. Ahora, nuestra relación entre las tensiones y las deformaciones depende sólo de 2 parámetros (no de 81): $\boldsymbol{\lambda}$ y $\boldsymbol{\mu}$, llamados **constantes de Lamé.**

Nos interesa la relación inversa, la deformación en función de la tensión. Para conseguir esa expresión, aislamos $\boldsymbol{\varepsilon}$ de *(VI.02)*. Debido a la presencia del operador $Tr(\cdot)$, aislar $\boldsymbol{\varepsilon}$ no es tan directo como parece. Primero hay que aplicar $Tr(\cdot)$ a cada lado de la expresión *(2)* para encontrar $Tr(\boldsymbol{\varepsilon})$ en función de $Tr(\boldsymbol{\sigma})$:

(VI.03)

$$Tr\left(\underline{\underline{\sigma}}\right) = 3 \cdot \lambda \cdot Tr\left(\underline{\underline{\varepsilon}}\right) + 2 \cdot \mu \cdot Tr\left(\underline{\underline{\varepsilon}}\right)$$

$$\downarrow$$

$$Tr\left(\underline{\underline{\varepsilon}}\right) = Tr\left(\underline{\underline{\sigma}}\right) \cdot \frac{1}{2 \cdot \mu + 3 \cdot \lambda}$$

Una vez llegados a este punto, encontramos $\boldsymbol{\varepsilon}$ aislando $\boldsymbol{\varepsilon}$ en *(VI.02)* y aplicando la relación vista en *(VI.03):*

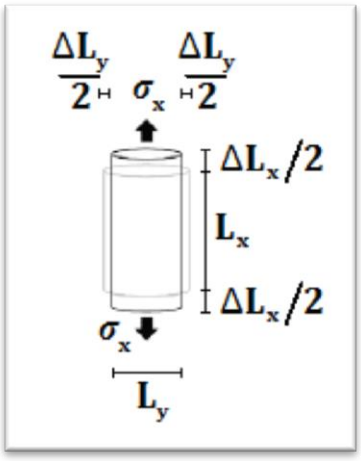
(VI.04)

$$\underline{\underline{\varepsilon}} = \frac{1}{2\,\mu} \cdot \underline{\underline{\sigma}} - \frac{\lambda}{2 \cdot \mu} \cdot Tr\left(\underline{\underline{\varepsilon}}\right) \cdot \underline{\underline{1}} = \frac{-\lambda \cdot Tr\left(\underline{\underline{\sigma}}\right)}{\left(2 \cdot \mu + 3 \cdot \lambda\right) \cdot 2 \cdot \mu} \cdot \underline{\underline{1}} + \frac{1}{2 \cdot \mu} \cdot \underline{\underline{\sigma}}$$

Las expresiones *(VI.02)* y *(VI.04)* nos dan la tensión en función de la deformación y viceversa. Estas relaciones están expresadas en términos de λ y μ. A nosotros, como ingenieros, nos interesa expresarlas en función de dos parámetros que tengan un sentido físico: los parámetros elásticos E y γ.

El siguiente apartado trata de encontrar la relación entre los parámetros { λ y μ } y los parámetros { E y γ }.

1.2. ENSAYO UNIAXIAL



VI.01: *Representación gráfica de un ensayo uniaxial* [7]

Un ensayo uniaxial se caracteriza por la aplicación de una tensión en una única dirección:

$$\underline{\underline{\sigma}} = \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad (VI.05)$$

Insertando *(VI.05)* en *(VI.04)* podemos obtener el tensor de deformaciones producidas en este caso:

$$\underline{\underline{\varepsilon}} = \frac{-\lambda \cdot Tr(\underline{\underline{\sigma}})}{(2\cdot\mu + 3\cdot\lambda)\cdot 2\cdot\mu} \cdot \underline{\underline{1}} + \frac{1}{2\cdot\mu} \cdot \underline{\underline{\sigma}} =$$

$$= \begin{bmatrix} \frac{-\lambda}{2\cdot\mu+3\cdot\lambda}+1 & 0 & 0 \\ 0 & \frac{-\lambda}{2\cdot\mu+3\cdot\lambda} & 0 \\ 0 & 0 & \frac{-\lambda}{2\cdot\mu+3\cdot\lambda} \end{bmatrix} \cdot \frac{\sigma_x}{2\cdot\mu} \qquad (VI.06)$$

[7] Elaboración propia a partir de la imagen extraída de la ***fuente [20]***

En la expresión *(VI.06)* podemos encontrar, entre otros, $\boldsymbol{\varepsilon_x}$, y $\boldsymbol{\varepsilon_y}$ en función de $\boldsymbol{\sigma_x}$.

Si definimos ***E*** **Módulo de Young** y ***γ*** **parámetro nu** de la siguiente manera, obtenemos:

(VI.07)

$$E = \frac{\sigma_x}{\varepsilon_x} = \ldots = \frac{(2 \cdot \mu + 3 \cdot \lambda) \cdot \mu}{\mu + \lambda}$$

$$\nu = \frac{-\varepsilon_y}{\varepsilon_x} = \ldots = \frac{\lambda}{2 \cdot (\mu + \lambda)}$$

Las ecuaciones *(VI.07)* nos relacionan los parámetros { $\boldsymbol{\lambda}$ y $\boldsymbol{\mu}$ } con los parámetros { ***E*** y ***γ*** }. Nos interesa la relación inversa, por lo que procedemos a aislar $\boldsymbol{\lambda}$ y $\boldsymbol{\mu}$ de las ecuaciones *(VI.07)*:

(VI.08)

$$\mu = \frac{E}{2 \cdot (1 + \nu)} \qquad \lambda = \frac{E \cdot \nu}{(1 - 2 \cdot \nu) \cdot (1 + \nu)}$$

1.3. LEY DE HOOKE GENERALIZADA

Si sustituimos las ecuaciones *(VI.08)* en *(VI.04)* obtenemos:

(VI.09)

$$\underline{\underline{\varepsilon}} = \frac{-\lambda \cdot Tr(\underline{\underline{\sigma}})}{(2 \cdot \mu + 3 \cdot \lambda) \cdot 2 \cdot \mu} \cdot \underline{\underline{1}} + \frac{1}{2 \cdot \mu} \cdot \underline{\underline{\sigma}} = \ldots = \frac{-\nu}{E} \cdot Tr(\underline{\underline{\sigma}}) \cdot \underline{\underline{1}} + \frac{1 + \nu}{E} \cdot \underline{\underline{\sigma}}$$

la relación entre tensión y deformación expresada en términos del Módulo de Young y el parámetro nu.

Utilizando la siguiente notación ingenieril:

(VI.10)

$$\varepsilon_{jj} \overset{not}{=} \varepsilon_{x_j} \qquad \sigma_{jj} \overset{not}{=} \sigma_{x_j}$$

$$\varepsilon_{ij} \overset{not}{=} \gamma_{x_i x_j} \qquad \sigma_{ij} \overset{not}{=} 2 \cdot \tau_{x_i x_j}$$

Podemos reescribir las componentes de lo tensores de deformación y tensión, que sabemos que son simétricos. Las ecuaciones que obtenemos *(VI.11)* se presentan en la siguiente página:

$$\varepsilon_x = \frac{1}{E}\left[\ \sigma_x\ -\nu\left(\sigma_y+\sigma_z\right)\right] \qquad \gamma_{xy} = \frac{2\cdot\left(1+\nu\right)}{E}\cdot\tau_{xy}$$

$$\varepsilon_y = \frac{1}{E}\left[\ \sigma_y\ -\nu\left(\sigma_x+\sigma_z\right)\right] \qquad \gamma_{xz} = \frac{2\cdot\left(1+\nu\right)}{E}\cdot\tau_{xz} \qquad \textit{(VI.11)}$$

$$\varepsilon_z = \frac{1}{E}\left[\ \sigma_z\ -\nu\left(\sigma_x+\sigma_y\right)\right] \qquad \gamma_{yz} = \frac{2\cdot\left(1+\nu\right)}{E}\cdot\tau_{yz}$$

Solo presentamos 6 componentes debido a la simetría de ambos tensores.

Estas seis componentes se pueden reorganizar en un vector columna 6x1, que expresaría la relación de la siguiente forma simplificada:

$$\begin{bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{xz}\\ \gamma_{yz}\end{bmatrix} = \frac{1}{E}\cdot\begin{bmatrix}1 & -\nu & -\nu & 0 & 0 & 0\\ -\nu & 1 & -\nu & 0 & 0 & 0\\ -\nu & -\nu & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 2\cdot\left(1+\nu\right) & 0 & 0\\ 0 & 0 & 0 & 0 & 2\cdot\left(1+\nu\right) & 0\\ 0 & 0 & 0 & 0 & 0 & 2\cdot\left(1+\nu\right)\end{bmatrix}\cdot\begin{bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{xz}\\ \tau_{yz}\end{bmatrix} \qquad \textit{(VI.12)}$$

La expresión *(VI.12)* indica de manera compacta las tensiones que se obtienen en función de las deformaciones presentes. Para encontrar la relación inversa, no habría más que invertir la matriz. Pero para estudiar la tensión plana nos conviene que la expresión *(VI.12)* se presente de esta manera.

1.4. TENSIÓN PLANA

La tensión plana se caracteriza por tener componentes de tensión únicamente sobre el plano *XY*, por lo que las condiciones que se deben cumplir son:

(VI.13)

$$\sigma_z = 0 \qquad \tau_{xz} = 0 \qquad \tau_{yz} = 0$$

Imponiendo *(VI.13)* en *(VI.12)* obtenemos:

(VI.14)

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \frac{1}{E} \cdot \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\cdot(1+\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\cdot(1+\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\cdot(1+\nu) \end{bmatrix} \cdot \begin{bmatrix} \sigma_x \\ \sigma_y \\ 0 \\ \tau_{xy} \\ 0 \\ 0 \end{bmatrix}$$

En tal caso, podemos simplificar aún más el problema tachando la tercera, la quinta y la sexta fila. Obtendremos el sistema 3x3 en *(VI.15)* y las relaciones *(VI.16)*:

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \frac{1}{E} \cdot \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\cdot(1+\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\cdot(1+\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\cdot(1+\nu) \end{bmatrix} \cdot \begin{bmatrix} \sigma_x \\ \sigma_y \\ 0 \\ \tau_{xy} \\ 0 \\ 0 \end{bmatrix}$$

(VI.15)

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \frac{1}{E} \cdot \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 2\cdot(1+\nu) \end{bmatrix} \cdot \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}$$

Con

(VI.16)

$$\gamma_{xz} = \gamma_{yz} = 0 \qquad \varepsilon_z = \frac{-\nu}{E} \cdot (\sigma_x + \sigma_y)$$

Si invertimos la matriz en *(VI.15)* obtendremos finalmente la relación que buscábamos desde el principio: las tensiones en función de las deformaciones en un estado de tensión plana.

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \frac{E}{(1+\nu)\cdot(1-\nu)} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \qquad (VI.17)$$

Esta ecuación es la base del capítulo I.

2. CÁLCULO EN MATLAB DETALLADO

Ver navier.txt

2.1. NAVIER

En este anejo se van a explicar detalladamente los procesos y las variables internas más relevantes que se desarrollan dentro de las funciones de cálculo.

Líneas 18-28
Líneas 30-99

En primer lugar se definen las **variables globales** de preproceso y de postproceso. Acto seguido el programa se dispone a **rellenar** el valor de estas variables de preproceso con los datos que ha introducido el usuario.

Sin embargo, por el camino el software realiza ciertas acciones relevantes:

Líneas 51-52

- Se **ordenan las variables *x1, x2, y1* e *y2***, de manera que *x1* sea menor o igual que *x2* para cada carga. Ídem para la coordenada *y.*

Líneas 42, 55-62

- Se **crean unas nuevas variables *x1_ch, x2_ch, y1_ch* e *y2_ch*.** Estas variables definen los límites de cada carga. En caso de carga repartida, estas variables coinciden con las variables globales *x1, x2, y1* e *y2*. En caso de carga puntual o lineal, estas variables vienen afectadas por otra variable: *epsilon.* De manera que: ***x1_ch=x1-epsilon; x2_ch=x2+epsilon; y1_ch=y1-epsilon; y2_ch=y2+epsilon;*** así pues, se han transformado todas las cargas existentes a cargas repartidas.

Líneas 38, 55-62

- Se **crea una nueva variable *coeff*.** Esta variable se usa para modificar el valor de las cargas puntuales y lineales, ya que han sido convertidas a repartidas. Si la carga es repartida, vale 1. Si la carga es lineal, vale 1/(2·*epsilon*). Si la carga es puntual, vale $1/(2\cdot epsilon)^2$.

Líneas 87-88

- Se **calculan las longitudes de la placa *Lx y Ly***.

Líneas 91-99

- Se **calcula la malla**. Por tanto, se rellenan las variables globales ***x_eval* e *y_eval,*** que tienen forma vectorial, con las diferentes coordenadas de los nodos de la malla en función de las particiones que haya definido el usuario.

A partir de aquí, el software dispone de todos los elementos necesarios para empezar a calcular.

Líneas 105-116

Lo primero que computa son los coeficientes q_{mn}, de la siguiente manera con la matriz q de dimensiones MxN:

$$q(m,n) = \frac{4}{L_x L_y} \cdot \lim_{epsilon \to 0} \sum_{i=1}^{nch} \int_{x1\_ch}^{x2\_ch} \int_{y1\_ch}^{y2\_ch} coeff \cdot f \cdot sin\left(\frac{m\pi(x - X1)}{L_x}\right) \cdot sin\left(\frac{n\pi(y - Y1)}{L_y}\right) dy\, dx$$

Líneas 119-125

De esta manera se incorpora el parámetro *coeff* en el cálculo y la variable *epsilon* ya desaparece en cualquier caso. El siguiente paso consiste en calcular los coeficientes w_{mn}:

$$w(m,n) = \frac{1}{D \cdot \pi^4 \cdot \left[\frac{m^2}{{L_x}^2} + \frac{n^2}{{L_y}^2}\right]} \cdot q(m,n)$$

Líneas 128-136

Acto seguido, se define la matriz *w_fourier*, de dimensiones también *M*x*N*. En cada componente alberga la siguiente expresión:

$$w\_fourier(m,n) = w(m,n) \cdot sin\left(\frac{m\pi X}{L_x}\right) \cdot sin\left(\frac{n\pi Y}{L_y}\right)$$

Línea 145

Con esto, ya podemos **calcular** la función de la flecha, que se corresponde con la variable global de postproceso ***func_w**:*

$$func\_w = \sum_{m=1}^{M} \sum_{n=1}^{N} w\_fourier(m,n)$$

Líneas 147-163

Esta expresión está escrita en lenguaje *simbólico* y en función de X e Y, coordenadas en la esquina de la placa. Más adelante se realizará un cambio de variable. Sin embargo, esta forma de escribir la flecha nos conviene para **encontrar las otras variables globales de postproceso** apartir de las derivadas y las operaciones correspondientes: *func_gx, func_gy, func_mx, func_my, func_mxy, func_qx, func_qqx, func_vx, func_qy, func_qqy* y *func_vy.*

Para ello, hay que tener en cuenta ciertas consideraciones:

- **Se escriben directamente como** un tipo de variable ***function handle***. De manera que el proceso de evaluación de las coordenadas en ellas sea muy eficiente.

- Se realiza un **cambio de variable** automáticamente, de *X* a *x-X1* y de *Y* a *y-Y1*.

- Se **sustituye el operador '*' por el operador '.*'**, de manera que se puedan sustituir en las expresiones anteriores varios puntos a la vez.

Línea 167

Por último, queda **realizar estas tres operaciones** citadas anteriormente también **a la variable *func_w***.

Líneas 175-265

Una vez tenemos todas las expresiones analíticas computadas correctamente, procedemos a encontrar el valor de las funciones solución en cada nodo de la malla; es decir, procedemos a **encontrar las variables globales de postproceso *W, Gx, Gy, Mx, My, Mxy, Qx, QQx, Vx, Qy, QQy y Vy***, que son matrices de dimensiones *O*x*P*, de igual tamaño que la malla (lógicamente).

Una vez realizados estos cálculos, ya se han obtenido las variables globales de postproceso relevantes. Éstas se van a almacenar en el *workspace* global, mientras que todas las otras variables que hemos ido definiendo por el camino en el *workspace* propio de la función *navier* van a desaparecer al finalizar este script.

Ver levy.txt

2.2. LEVY

Hasta la línea 80, el código de Levy es idéntico al de Navier. A excepción de la línea 18, en la que Levy define también las coordenadas globales de postproceso *id1* e *id2*, que se corresponden con las condiciones de contorno *Y1* e *Y2*. En las líneas 80 y 81, Levy rellena estas variables globales con el valor que el usuario les ha dado (0 para contorno apoyado, 1 para contorno empotrado y 2 para contorno libre).

A partir de este punto, las líneas comprendidas entre la 82 y la 121 son idénticas a las líneas escritas en Navier, comprendidas entre la 79 y la 118. Esto incluye el cálculo de los coeficientes q_{mn}.

Líneas 122-130

El **cálculo de *w***, sin embargo, es un poco distinto, ya que se aprovecha para incorporar junto a los coeficientes w_{mn} el seno en *y* que desarrolla la serie de Fourier, porque así nos conviene:

$$w(m,n) = \frac{1}{D \cdot \pi^4 \cdot \left[\frac{m^2}{{L_x}^2} + \frac{n^2}{{L_y}^2}\right]} \cdot q(m,n) \cdot sin\left(\frac{n\pi\left(y - L_y/2\right)}{L_y}\right)$$

Línea 133

Nótese que el eje *y* en este caso pasa por el centro de la placa. Acto seguido, **se define el vector *ww*,**, de dimensiones *m*x*1*, que se corresponde con los coeficientes w'_m. Estos coeficientes se corresponden con el término independiente a A_m, B_m, C_m y D_m que aparece dentro del paréntesis de la ecuación (*I.50)* del capítulo 1 de este documento. Sin embargo, escrito con los ejes desplazados por comodidad de programación.

$$ww(m) = \sum_{n=1}^{N} w\_fourier(m,n)$$

Hemos visto la parte que corresponde con la solución particular. Veamos ahora la homogénea.

Escribimos en lenguaje simbólico la expresión Y_m que aparece en el capítulo 1 en la ecuación *(I.49)*, con sus incógnitas A_m, B_m, C_m y D_m definidas como componentes de los vectores incógnita *A, B, C* y *D*. Para ello, **creamos** un vector de *M* componentes llamado ***Y_levy:***

Líneas 136-149

$$Y\_levy(m) = A(m) \cdot cosh(\varphi) + B(m) \cdot \varphi \cdot sinh(\varphi) + C(m) \cdot sinh(\varphi) + D(m) \cdot \varphi \cdot cosh(\varphi)$$

Con

$$\varphi = \frac{m\pi Y}{L_x}$$

Una vez más el eje *y* pasa por el centro de la placa.

Acto seguido, **se define el vector *w_fourier*** (ahora es un vector) de dimensiones *M*x*1*. En cada componente alberga la siguiente expresión, que se corresponde con el término *m* de la serie de Fourier que describe la flecha:

Líneas 152-156

$$w\_fourier(m) = sin\left(\frac{m\pi X}{L_x}\right) \cdot (Y\_levy(m) + ww(m))$$

Con esta expresión o sus derivadas ya podemos **escribir** las condiciones de contorno. Éstas se escriben en función de *id1* e *id2*. Las variables que usamos para escribirlas se llaman ***eq1, eq2, eq3* y *eq4*.** Estas variables son vectores nuevamente de longitud *M*.

Líneas 159-218

Hay que tener en cuenta que una vez se han escrito estas ecuaciones, se les quita a todas ellas el término

$$sin\left(\frac{m\pi X}{L_x}\right)$$

tal y como se explica en el apartado 2.3 del capítulo 1.

Ahora tenemos para cada iteración *m* 4 expresiones que **igualadas a cero** y **evaluadas en –Ly/2 y Ly/2** son numéricas, y **constituyen un sistema de ecuaciones** algebraicas del que **podemos obtener las variables *AA, BB, CC* y *DD***.

Líneas 221-226

Una vez se han obtenido estos 4 vectores, se **sustituyen en *w_fourier* por las incógnitas *A, B, C* y *D***.

Línea 227

Línea 238

Una vez tenemos *w_fourier* completo, se suman sus componentes para **obtener la función analítica *func_w***

Línea 240 en adelante

A partir de este punto, el método de Levy **se desarrolla** exactamente **igual** que el de Navier a partir de la línea 147, **con una única diferencia**: ahora *Y* debe ser sustituida por *y-Y1-Ly/2* (y no por *y-Y1* como en Navier), para colocar los ejes de coordenadas a su posición original.

BIBLIOGRAFÍA

TEORÍA DE PLACAS

[01] Argüelles, R., *Cálculo de estructuras. Tomo II*, Grefol (1981) ISBN 84-600-2412-1

[02] Billington, D., *Thin Shell concrete structures,* McGraw-Hill (1965) ISBN 0070052794

[03] Cheung, I. and Neal, B., *Finite Strip method in structural analysis,* Pergamon Press (1976) ISBN 978-0-08-018308-4

[04] Corchero, R., *Cálculo de estructuras (resolución práctica),* Rugarte (1993) ISBN 84-7493-110-X

[05] Love, A.E.H., *A treatise on the mathematical theory of elasticity*, Dover Publications, Inc. (1944) ISBN 0-486-60174-9

[06] Suárez B., Canet J. and Codina R., *Placas* [Manuscrito]

[07] Timoshenko, S. and Woinowsky-Krieger, S., *Teoría de placas y láminas*, URMO (1975) ISBN 84-314-0116-8

[08] Ugural, A., *Stresses in plates and shells,* McGraw-Hill (1981) ISBN 0070657696

TEORÍA DE LA ELASTICIDAD

[09] Chung, T., *General Continuum Mechanics*, Cambridge University Press (2007) ISBN 978-0-521-87406-9

[10] Mase, T., Smelser, R., Mase, G., *Continuum Mechanics for engineers,* CRC Press (2010) ISBN 9781420085389

[11] Oliver, X. and Agelet de Saracíbar, C., *Mecánica de medios continuos para ingenieros,* Edicions UPC (2000) ISBN: 84-8301-412-2

DESARROLLO EN SERIES DE FOURIER

[12] Estela, M. and Saà, J., *Cálculo con soporte interactivo en moodle*, Pearson Prentice Hall (2008) ISBN 978-84-832-2480-9

[13] Hernández, O., *Métodos de Fourier en la física y la ingeniería*, Trillas (1973)

[14] Seeley, R., *Introducción a las series e integrales de Fourier*, Reverté (1970)

PROGRAMACIÓN EN MATLAB

[15] Biran, A., Breiner, M., *What every engineer should know about MATLAB,* CRC Press (2010) ISBN 978-1-4398-1020-0

[16] Marchand, P. and Holland, O., *Graphics and GUIs with MATLAB,* Chapman & Hall/CRC (2003) ISBN 1-58488-320-0

[17] The MathWorks, Inc., *Building GUIs with MATLAB,* (1997)

[18] The MathWorks, Inc., *MATLAB Documentation Center:*
http://www.mathworks.es/es/help/index.html

APORTACIONES GRÁFICAS

[19] Archivo Learn-Maths:
http://learn-math.info/history/photos/Love_2.jpeg

[20] Biblioteca Virtual SciELO:
http://scielo.isciii.es/img/revistas/romm/v5n1/revision1_fig1.gif

[21] Wikimedia Commons:
http://commons.wikimedia.org/wiki/File:Gustav_Robert_Kirchhoff.jpg