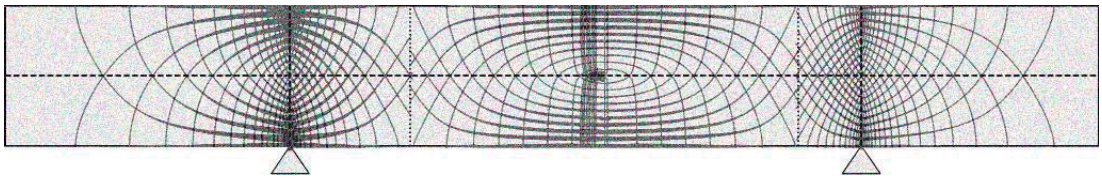
EL CAMINO DE LAS FUERZAS

Creación de aplicación gráfica interactiva para la generación automática de líneas isostáticas del campo elástico de tensiones planas en elementos estructurales unidimensionales a flexión



TFG

Estudiante: Diego Lidón Segura

Tutor: Fernando Gómez Martínez

01 julio 2021

I N D I C E

S I M B O L O S

A	Área de la sección
α1	Angulo 1 tensión principal
α2	Angulo 2 tensión principal
b	Base de la sección
dx	diferencial eje x
dy	diferencial eje y
dz	diferencial eje z
E	Módulo de Young
I(y)	Momento de Inercia
L	longitud de viga
(λ_1)	valor principal de compresión
(λ_2)	tensión principal de tracción
M(x)	Momento
$\{n\}$	vector normal
$N(x)$	Axil
q	carga
S(y)	Momento estático
(σ)	Tensión normal
{t}	vector de tensión
(T)	matriz-tensor de tensiones
(τ)	Tensión tangencial
τV	tensión tangencial de cortante
τR	tensión tangencial de rasante
$\{v1\}$	vector normal 1
$\{v2\}$	vector normal 2
$V(x)$	Cortante
x	Variable en el eje x

Resumen

Se pretende elaborar una herramienta gráfica que nos permita desarrollar de manera clara, fiable y visualmente atractiva el camino que recorren las fuerzas internas de los elementos estructurales que conforman cualquier tipo de proyecto arquitectónico. Estas fuerzas tensionales también conocidas como líneas isostáticas son generadas por las cargas aplicadas a los elementos estructurales.

Estando presentes en el interior de las estructuras que conforman las arquitecturas del hoy, nos marcamos como objetivo obtener un mayor grado de comprensión en la didáctica de dichos comportamientos, pudiendo así poner en valor su concepción en el campo de la enseñanza de una manera mucha más gráfica e intuitiva.

Ya existen estudios en los que se llegan a hallar las direcciones principales continuas sobre elementos finitos, y la finalidad principal de casi todos estos es la de optimizar el uso de material en la creación de dichos elementos para su puesta en obra.

Es por ello que se realiza este trabajo ya que no existe nada que hayamos podido comprobar que se le parezca sobre elementos unidimensionales aplicados a la enseñanza, y lo que existe se centra sobre todo en elementos de dos dimensiones.

Abstract

The aim of this work is to establish a graphic tool that allows us to develop in a clear, reliable and visually attractive way the path taken by the internal forces of the structural elements that make up any type of architectural project. These tensional forces, also known as isostatic lines, are generated by the loads applied to the structural elements.

Being present inside the structures that make up today's architectures, we set the objective of obtaining a greater degree of understanding in the didactics of said behaviors, thus being able to value their conception in the field of teaching in a much more graphical and intuitive way.

There are already studies in which the continuous main directions on finite elements are found, and the main purpose of almost all of these is to optimize the use of material in the creation of said elements for their implementation.

That is why this work is carried out since there is nothing that we have been able to verify that resembles it about one-dimensional elements applied to teaching, and what exists focuses mainly on two-dimensional elements.

INTRODUCCION

PREFACIO

"Aun cuando no se diga nada nuevo, un simple comentario permite apreciar un diferente sesgo de la cuestión."

Eduardo Torroja.

Junto a Fernando, mi tutor, el que me ha abierto las puertas a este tema de investigación, nos disponemos a aportar un simple y nuevo comentario a lo tensional de una estructura para así poder dar forma con exactitud a las líneas isostáticas presentes en su interior.

Este no es un libro de texto que deduzca las complejas leyes que rigen los estados tensionales que un técnico debe conocer, es por eso que se utiliza como apertura de este trabajo fin de grado. Es válido como introducción al concepto básico de una estructura y una vez habiendo simplificado el camino para el lector interesado, pueda extenderse más tarde en la teoría de conceptos más complejos.

Todo pensado de tal manera que pueda llegar al mayor número de público el entendimiento de algo tan básico y natural como las líneas isostáticas internas de una estructura.

Fernando Torroja empieza este capítulo II haciendo hincapié en que los principios tensionales que rigen todos los fenómenos resistentes son fundamentales para todo aquel que pretenda entender lo que ocurre en el interior de una estructura.

No sólo basta con canalizar los procesos teóricos, sino que es necesario haber meditado y experimentado sobre ello hasta lograr sentirlo como algo propio y de esta manera intuir de golpe los fenómenos de tensión y deformación que inclinan a trabajar a la estructura de una manera u otra.

Un arquitecto ha de llegar a un estado mencionado como "Einfühlung" (estado emocional) en el que pueda entender que lo deformatorio siempre va unido a todo lo tensional.

La heurística (indagación y descubrimiento) de la estructura va ligada desde un primer momento al conocimiento de su carácter resistente.

De la estructura se enmarca el propio material que la conforma, y se menciona que ha de ser capaz de soportar las fuerzas internas a las que se somete. Del tipo de material depende el resultado del estado de carga general de las fueras exteriores y por ende las acciones locales de cada fuerza en su interior.

Para entender las líneas internas de una estructura sometida a fuerzas exteriores, hay que saber que dichas fuerzas necesitan transmitirse y equilibrarse a través del sólido creando así estados de tensiones internos relacionados entre sí a la vez que individuales en cada punto de la estructura.

Podemos imaginar que la estructura se divide en miles de puntos internos. Si tomamos cada punto como un cubo elemental, podemos imaginar como en sus caras actúan solamente fuerzas normales, producto de las fuerzas que buscan transmitirse por el interior del sólido.

Si esas fuerzas que actúan en cada uno de esos puntos, las uniéramos con los puntos más cercanos posibles, al final obtendríamos una red de envolventes que es a lo que llamamos líneas isostáticas que son las que nos permitirán una buena representación del fenómeno tensional interno de una estructura.

"El imaginar la estructura deformándose, bajo la acción de las cargas a que se la somete, es indudablemente la mejor ayuda que se puede tener al tratar de imaginar, no sólo el estado de tensión del sólido, sino también el lugar y la forma en que el material puede fallar."

Eduardo Torroja

Libro "Razón y ser de los tipos estructurales: Eduardo Torroja, 1957, textos universitarios", 13, capitulo 2, páginas 23-34)

OBJETIVO

En este Trabajo Fin de Grado se pretende elaborar una herramienta gráfica que nos permita desarrollar de manera clara, fiable y visual el camino que recorren las fuerzas internas de los elementos estructurales que conforman cualquier tipo de proyecto arquitectónico.

En este caso, el estudio está enfocado en hallar la representación lineal (de una manera gradual y escalada) de las conocidas como líneas isostáticas, que son las tensiones generadas por las cargas aplicadas a los elementos estructurales y que recorren las entrañas de estos.

Estando presentes en el interior de los elementos estructurales que conforman las arquitecturas del presente, nos marcamos como objetivo obtener un mayor grado de comprensión en la didáctica de dichos comportamientos, pudiendo así poner en valor su concepción en el campo de la enseñanza, y de una manera mucha más gráfica e intuitiva, realzar su entendimiento pudiendo llegar a ser un concepto mucho más práctico para el usuario que se enfrente por primera vez a estos.

JUSTIFICACIÓN

Tras la búsqueda de estudios que se hayan realizado para la obtención de estas líneas isostáticas en elementos unidimensionales, nos damos cuenta de que no existe ningún tipo de programación que nos permita visualizar estas líneas continuas en elementos tipo viga, los trabajos que se han realizado hasta el día de hoy se llevan a cabo sobre todo en elementos bidimensionales como losas, muros, láminas, etc.

En estos trabajos vemos que se llegan a hallar las direcciones principales continuas sobre elementos finitos, y la finalidad principal de casi todos estos es la de optimizar el uso de material en la creación de dichos elementos para su puesta en obra.

Es por ello que realizamos este trabajo ya que no existe nada que hayamos podido comprobar que se le parezca sobre elementos unidimensionales aplicados a la enseñanza, y lo que existe se centra sobre todo en elementos de dos dimensiones.

METODOLOGÍA

Para lograr la producción de las líneas isostáticas de una manera automática y ajustada a cualquier ejemplo didáctico, lo primero que nos planteamos después de demostrar que es imposible su obtención manual exacta, es el lugar en el cual podemos programar dicho estudio para que los resultados sean a la vez que numéricos, gráficos.

Elegimos el programa Python ya que es un programa de código abierto y nos permite obtener tanto numérica como gráficamente sin ningún tipo de problema el trabajo que queremos llevar a cabo.

Una vez escogido el software donde desarrollar el estudio, debemos acotar el alcance que va a llegar a tener. Debe de ser un alcance modesto puesto que es la primera vez que se

realiza un trabajo de estas características; es por ello que descartamos las estructuras hiperestáticas, evitando así tener que manejar fórmulas muy tediosas para la obtención de sus solicitaciones. Se llega a la conclusión de que las vigas isostáticas son ideales para comenzar el estudio.

Dentro de las vigas isostáticas, nos centramos en su caso más complejo, una viga que consta de tres partes: un vano central y dos voladizos, uno a cada lado de la parte central. Las cargas que tendrán cada una de las partes de la viga serán una repartida y una puntual, pudiendo ser modificados sus valores para así disponer diferentes casos de estudio.

Una vez acotado el caso que vamos a estudiar nos disponemos a programar la expresión de dichas líneas isostáticas por el método más eficiente posible (más tarde veremos que necesitamos de un método discreto).

ESTADO DE LA CUESTIÓN

OPTIMIZACIÓN ESTRUCTURAL

Las líneas de tensión principales se pueden observar cómo pares de curvas ortogonales que indican las trayectorias de las tensiones internas en una estructura, es decir, se pueden entender como caminos idealizados a través de los cuales el material se comporta estructuralmente de una manera continua y óptima.

Este análisis de las líneas de isostáticas ofrece un enfoque geométrico bastante provocativo que puede permitir entrar en el campo de la optimización de material en la estructura arquitectónica. Se puede retirar material de aquellas zonas en las que la representación de las líneas sea nula. Esto ofrece a su vez una síntesis de diseño bastante fuerte; hermanado así diseño y estructura.

Bien es verdad que su aplicación en el diseño es bastante limitada, ya que la estandarización parametrizada de las líneas isostáticas de cualquier elemento arquitectónico puede ser un trabajo bastante complejo.

Hay muchos estudios sobre este tema, los cuales se han dedicado a la búsqueda de esta relación entre estética en el diseño de la estructura y su optimización consecuente, yendo más allá de las barreras que esto, en algunos casos puede llegar a suponer.

Estos trabajos tienen como premisa la investigación de una nueva concepción en el mundo de las estructuras. Introduciendo el mundo del diseño en el complejo campo del comportamiento físico permanente de la arquitectura; a su vez, permite conseguir el rendimiento óptimo de cualquier estructura con el fin de equilibrar diseño y estructura.

HERRAMIENTA GRAFICA

El grafismo de dichas líneas isostáticas, aparte de conllevar un trabajo matemático y físico bastante exigente para su obtención, presenta un problema real del cual hemos ido tomando nota en la lectura de los diferentes estudios que han caído en nuestras manos.

En muchos de los trabajos revisados para la realización de este TFG, se ha observado siempre un apartado dedicado a problemas usuales que surgían a la hora de obtener líneas tensionales internas en programas informáticos como Millipede (Panagiotis 2014) o Karamba (Preisinger 2015).

En muchos casos se daba que la representación de líneas era muy densa en una zona, acumulando una gran cantidad de tensiones que cruzaban el mismo punto del elemento estructural, rompiendo totalmente la uniformidad de la malla creada por estas. Por otro lado, esta alta densidad de líneas en una misma zona hace que, en los puntos por los que debieran pasar originalmente queden entonces vacíos; muchos de estos mismos estudios proponen soluciones a dichos contratiempos.

En algunos casos se opta por el dibujo lógico que deberían tener estas líneas, guiándose por el sentimiento y la intuición (de manera informatizada). En otros muchos más casos, se llevan a cabo correcciones en los cálculos, llevando a obtener resultados muchos más precisos en la obtención de las líneas isostáticas.

Toda esta información nos mantiene alerta y nos ofrecen soluciones variopintas a la hora de enfrentarnos a estos percances que pueden ocurrir en nuestro estudio.

HERRAMIENTA DE SOFTWARE

Pero cabe repetir que todos estos trabajos se han llevado a cabo en elementos finitos como un muro o una viga, pero el estudio que queremos abordar nosotros es un proceso radicalmente diferente.

Estos trabajos realizan subdivisiones en puntos concretos de una malla, y de estos mismos se obtiene vectores orientado hacia donde van las tensiones.

Se obtiene el módulo y el ángulo (el autovalor y el autovector), pero no es eso lo que buscamos nosotros; lo que estamos haciendo precisamente no es elegir previamente una serie de puntos en un elemento estructural y a partir de ahí empezar a representar las líneas isostáticas, si no que elegimos los puntos que nos hacen falta para definir una línea de manera continua representada en el elemento unidimensional.

TEORÍA DE LA FLEXIÓN
(TENSIONES NORMALES Y TANGENCIALES)

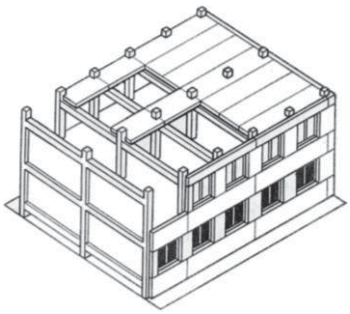
- **ESTRUCTURA / BARRA / REBANADA / PUNTO**

Una estructura se puede dividir en una serie de subsistemas menores internos de los cuales se consiguen obtener toda la información relativa para el cálculo total de aquellos fenómenos que ocurren en su interior. Es así que, si tomamos como sistema global y principal una estructura de cualquier edificio, de esta podemos obtener un subsistema menor al cual llamaremos "barra" o viga; de este mismo subsistema lograremos una partición menor que se conoce como "rebanada" o sección; y de la cual podemos estudiar en última instancia otro subsistema llamado punto.

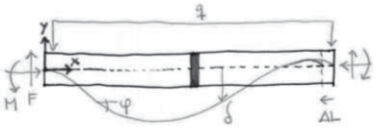
En orden obtenemos la siguiente regla:

"Estructura > barra > rebanada > punto"

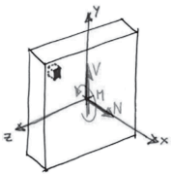
Estructura



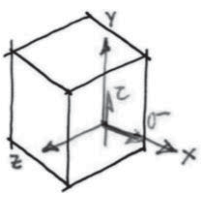
Viga-Barra



Sección-Rebanada



Punto



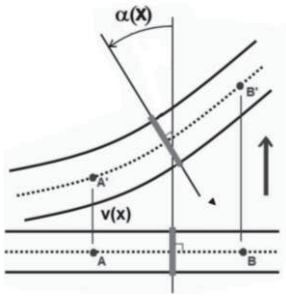
La segregación de todos estos subsistemas dentro de la estructura es muy importante a la hora del estudio de las líneas isostáticas de una viga.

En cada punto que contiene una viga existirán unas tensiones internas conocidas como: tensiones tangenciales y normales (τ y σ) las cuales se obtienen mediante el cálculo de todas las propiedades de los subsistemas que anteceden al punto.

- **HIPOTESIS DE NAVIER-BERNOUILLI**

Para el ámbito de estudio es importante conocer qué ocurre en el interior de las barras, rebanadas y puntos cuando aplicamos cargas que los deformarán.

De dos de los estudiosos más importantes de sus épocas (Claude-Louis Marie Henri Navier y Jakob Bernouilli), se obtiene una hipótesis que estudia el comportamiento interno de las secciones de una barra a la cual se le aplica una deformación.



Si dividimos una barra en rebanadas, después de aplicar una deformación a dicha barra, las secciones efectuarán un giro junto con la deformación aplicada, pero se mantendrán siempre perpendiculares al eje central de dicha barra.

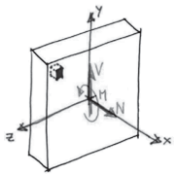
- **SOLICITACIONES**

Volviendo al esquema principal de la "Estructura > barra > rebanada > punto", y teniendo en el punto que hallar unas tensiones tangenciales y normales (τ y σ); estas se hallan a partir de unas solicitaciones aplicadas dentro de la rebanada: las solicitaciones de Axil *N(x)*, Cortante *V(x)* y Momento *M(x)*.

REBANADA

Solicitaciones

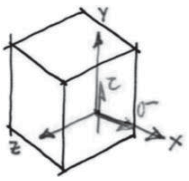
N(x), *V(x)* y *M(x)*



PUNTO

Tensiones

τ y σ



Para poder entonces calcular las Tensiones tangencial y normal (τ y σ) de un punto cualquiera de la barra, previamente hemos de conocer las solicitaciones que hay en la rebanada en la que se encuentra dicho punto.

Se conoce que la integral de la carga que actúa en la sección de una viga, será el cortante que se produce en el interior de la misma; a su vez, la integral de este cortante será el momento producido en el mismo lugar de esta; de manera inversa, la derivada del momento es el cortante, y la derivada de este la carga aplicada inicial.

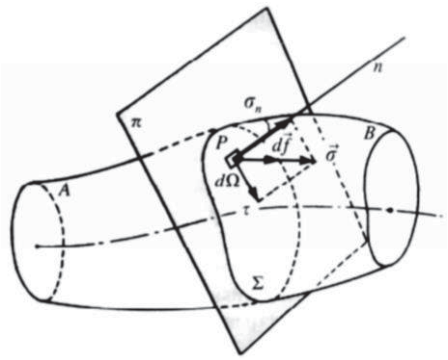
Así se conocen las relaciones existentes entre solicitaciones aplicadas a una barra.

• TENSIÓN EN UN PUNTO Y SUS COMPONENTES

La tensión aplicada en cualquier punto de la rebanada se descompone en las dos tensiones ya mencionadas previamente: Tensión normal (σ) y Tensión tangencial (τ).

Ambas tensiones dependen del plano de corte en el que se encuentre la rebanada que contiene al punto del cual estamos calculando dichas tensiones.

Este plano tiene dos direcciones que tomaremos como principales: una es la perpendicular al mismo que se denomina tensión tangencial, y otra paralela contenida en dicho plano de corte que es la que se conoce como tensión normal.



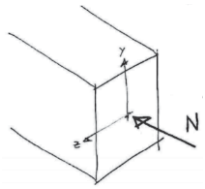
• RELACIÓN SOLICITACIONES / TENSIONES

La tensión tangencial (τ), es decir, la que es tangente al plano de la rebanada, está muy relacionada con el cortante $V(x)$.

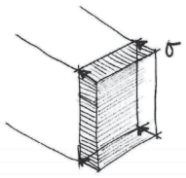
La tensión normal (σ), es decir, la que es perpendicular al plano de la sección, está relacionada con el axil $N(x)$.

- **AXIL, N(x) / TENSIÓN NORMAL, σ**

El axil es una solicitación perpendicular a la sección, es por ello que está relacionado con la tensión normal (σ). Existe una relación directa entre el área de la sección en la que esta tensión actúa y la solicitación, pues una se opone a la otra.

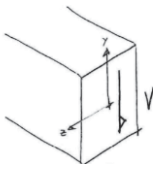


$$\sigma = \frac{N}{A}$$

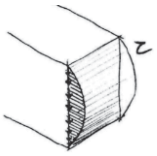


- **CORTANTE, V(x) / TENSIÓN TANGENCIAL, τ**

La solicitación de cortante *V(x)* produce unas tensiones tangenciales al plano de la sección; se relacionan el cortante, el momento estático de la sección, la base y el momento de inercia en la fórmula que define esta sección (Fórmula de Collignon).



$$\tau = \frac{V(x) \cdot S(y)}{b \cdot I(y)}$$



- **DEMOSTRACIÓN FÓRMULA DE COLLIGNON**

Es la fórmula que sirve para obtener la tensión tangencial de una sección, sometida a cortante, relacionando así solicitación y tensión.

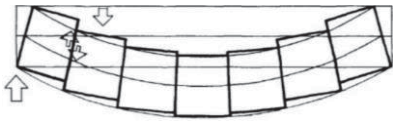
Cortante / Rasante

Si las "dovelas" o "lajas" en las que se puede descomponer una viga no estuvieran unidas entre sí, es decir, no fueran un único elemento, estas piezas deslizarían unas sobre otras. Los esfuerzos que las mantienen unidas son el cortante y rasante, respectivamente.

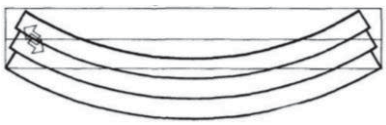
El rasante no es una solicitación de barra ya que no actúa a lo largo del eje x, que es el único eje que tiene dimensión real en una pieza unidimensional.

Es por ello que se busca una relación entre cortante y rasante para evitar referirse así al esfuerzo rasante.

Cortante: tensiones tangenciales

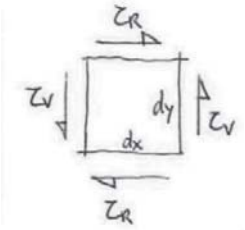


Rasante: tensiones tangenciales



Ambos esfuerzos (cortante y rasante) producen tensiones tangenciales de cortante y de rasante (τV, τR) sobre las caras verticales y horizontales, respectivamente, del punto al que afecta la rebanada. Estos esfuerzos se compensan entre sí.

Véase esquema:



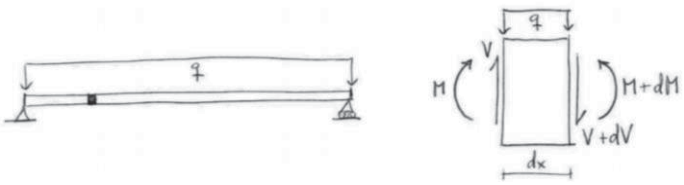
Se demuestra, mediante el Teorema de Cauchy, que ambas tensiones tangenciales son iguales, sin más que calcular la resultante de tensiones en cada cara y tomar momentos respecto de una esquina del elemento diferencial:

$$(\tau_R \cdot dx \cdot dz) \cdot dy - (\tau_V \cdot dy \cdot dz) \cdot dx = 0 \quad \Rightarrow \quad \tau_R = \tau_V$$

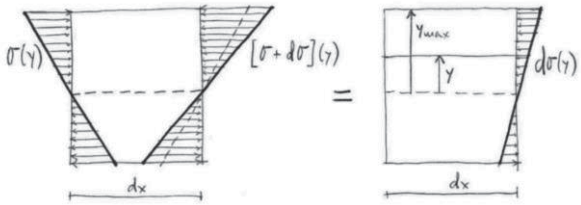
Por tanto, se puede utilizar τR para calcular τV.

El objetivo es obtener una relación entre $V(x)$ y τ, es decir, conocer qué tensión tangencial en cada punto de la sección produce el cortante.

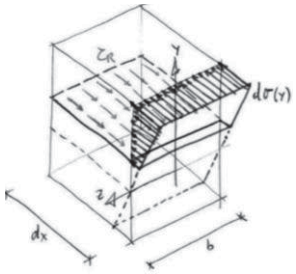
Para ello se toma una rebanada de una viga en un caso de flexión simple.



Sustituyendo los momentos por sus tensiones normales generadas, se tiene una resultante de tensiones equivalente al incremento de momento en una sección respecto de la opuesta:



Si se toma sólo el volumen de rebanada situado por encima de una cota "y" arbitraria, y considerando que se debe satisfacer el equilibrio de fuerzas horizontales, se concluye que debe aparecer en la superficie horizontal cortada una tensión rasante horizontal de sentido opuesto a la resultante de tensiones normales sobre la sección.



Planteando el equilibrio de fuerzas horizontales (y considerando que la fuerza producida por una tensión es el producto de dicha tensión por la superficie donde actúa):

$$\tau(y)\cdot dx\cdot b(y)=\int_{y}^{y_{max}} d\sigma(y)\cdot b(y)\cdot dy \quad \Rightarrow$$

$$\Rightarrow \quad \tau(y)=\frac{\int_{y}^{y_{max}} d\sigma(y)\cdot b(y)\cdot dy}{b(y)\cdot dx}$$

Se sustituye el diferencial de tensión normal por su valor según la Ley de Navier:

$$d\sigma(y)=-\frac{dM(x)}{I}y$$

Queda, por tanto:

$$\tau(y) = \frac{\int_{y}^{y_{max}} -\frac{dM(x)}{I} y \cdot b(y) \cdot dy}{b(y) \cdot dx} \quad \Rightarrow$$

$$\Rightarrow \quad \tau(y) = -\frac{dM(x)}{I \cdot b(y) \cdot dx} \int_{y}^{y_{max}} y \cdot b(y) \cdot dy$$

En esta última expresión, se sustituye el momento por su relación diferencial con el cortante:

$$-\frac{dM(x)}{dx} = V(x)$$

Y la integral es, por definición, igual al momento estático de la parte de la sección situada por encima de la fibra "y" considerada:

$$\int_{y}^{y_{max}} y \cdot b(y) \cdot dy = S_{(y)}$$

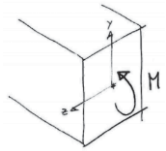
Luego:

$$\tau(y) = -\frac{dM(x)}{I \cdot b(y) \cdot dx} \int_{y}^{y_{max}} y \cdot b(y) \cdot dy \quad \Rightarrow \quad \tau(y) = \frac{V(x) \cdot S(y)}{I \cdot b(y)}$$

- **MOMENTO, M(x) / TENSIÓN NORMAL, σ**

El momento que se produce en el eje central de la sección de la viga, origina dos solicitaciones perpendiculares a la sección. Ambas en sentidos opuestos que crean unas tensiones normales opuestas.

La relación para hallar la tensión en un punto de la sección se da entre el momento, la inercia de la sección y la distancia del punto que se calcula respecto al eje de inercia de la sección.



$$\sigma = -\frac{M}{I}\, y$$

- **DEMOSTRACIÓN FÓRMULA DE NAVIER EN FLEXIÓN SIMPLE**

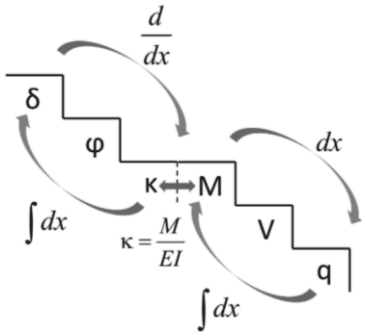
Es la fórmula que sirve para obtener la tensión normal de una sección, sometida a flexión simple, relacionando así solicitación y tensión.

La fórmula de la tensión en cada fibra, *σ(y)*, a partir del momento solicitación, se obtiene sustituyendo la expresión de la curvatura:

$$K = \frac{M}{E\,I} \quad \Rightarrow \quad \sigma(y) = -E\,K \cdot y \quad \Rightarrow \quad \sigma(y) = -E \cdot \frac{M}{E\,I}\, y \quad \Rightarrow$$

$$\Rightarrow \quad \sigma(y) = -\frac{M}{I}\, y$$

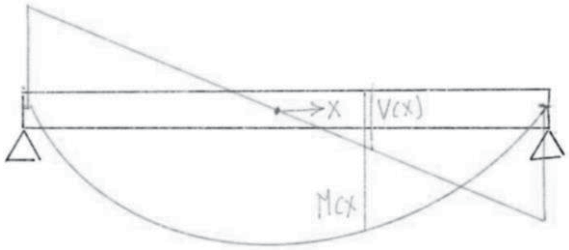
Quedan así relacionadas las solicitaciones y las tensiones. Dicha relación puede esquematizarse de la siguiente manera:



- **TENSIONES Y DIRECCIONES PRINCIPALES**

Se considera una viga recta como un elemento unidimensional en la cual el Eje *x* se posiciona a lo largo del eje longitudinal de la misma.

De esta viga se puede hallar la ley de Cortante *V(x)* y Momento *M(x)*, por ejemplo, para una viga con carga constante podemos obtener fácilmente sus solicitaciones en cualquier punto del Eje *x* de la viga; es decir, para cualquier punto o variable de la viga se obtienen dos funciones que dependen de *(x)*.

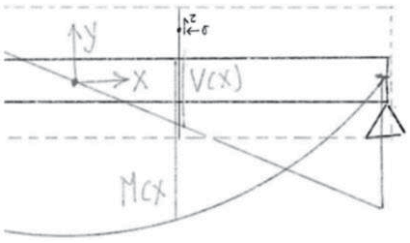


Se considera de esta viga una segunda dimensión en su eje plano frontal; de dicho plano se toma una rebanada.

En cada una de las fibras (o puntos) de esta nueva rebanada, tendrá existencia un estado tensional definido por la solicitación de Cortante *V(x)* y Momento *M(x)* así como por su cota en *(y)*; dicho estado tensional queda determinado en el mismo eje de referencia orientado en los ejes cartesianos.

En cualquier punto de la sección, por el hecho de estar a una cota *(y)* se hallará una expresión traducida en una tensión normal (σ) que será perpendicular a la superficie de la sección, y una tensión tangencial (τ) que se orientará dependiendo de la posición de la fibra que se estudie.

En este ejemplo, el punto que se estudia se encuentra en la parte superior-derecha de la viga, por tanto, se obtiene una tensión normal (σ) y una tensión tangencial (τ) orientada hacia arriba.



Estas dos expresiones de tensiones se hallan por las leyes de Collignon y de Navier; es decir, a partir de *(x)* obtengo el Momento *M(x)* y el Cortante *V(x)*; a su vez, de *M(x)* se obtiene $\sigma\ (x, y)$ y de *V(x)* resulta la tensión $\tau\ (x, y)$.

LEY DE COLLIGNON

Depende de variables *(x, y)*

$$\tau(x,y) = \frac{V(x) \cdot S(y)}{I\ b}$$

LEY DE NAVIER

Depende de variables *(x, y)*

$$\sigma(x,y) = -\frac{M(x)}{I}\ y$$

Así queda este esquema:

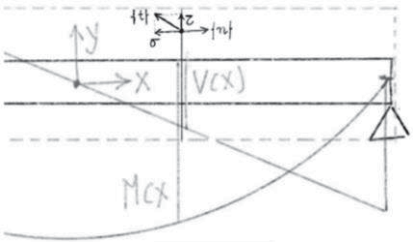
$$x \begin{cases} V(x) \\ M(x) \end{cases} \begin{cases} \tau(x,y) \\ \sigma(x,y) \end{cases}$$

En cada plano de corte (definido con su vector normal $\{n\}$, por cada punto que se estudia, obtendremos un vector tensión que se denomina como $\{t\}$ y que está compuesto por $\sigma\,(x, y)$ y $\tau\,(x, y)$.

Lo que aquí se define no deja de ser una aplicación lineal, una función cuya constante es una matriz. Función en la cual, por cada vector normal del plano $\{n\}$, obtendremos un vector tensión $\{t\}$ que afecta a un punto en particular de ese plano.

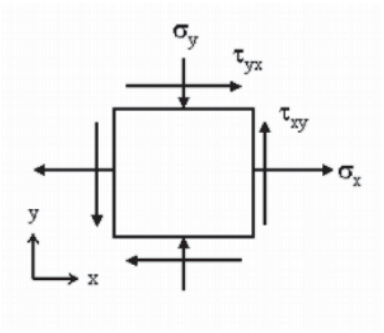
Es decir, para obtener el vector tensión $\{t\}$, se debe multiplicar por la matriz-tensor de tensiones (T) el vector normal del plano $\{n\}$ el cual contiene el punto concreto de estudio.

$$\{t\} = (T)\{n\}$$



Si se percibe un elemento que contenga al punto de la sección en cuestión, se ha de saber que por el hecho de estar en esa posición dentro de la viga se verá afectado por unas fuerzas de compresión en el plano horizontal; a su vez, en el plano vertical se hallan unas fuerzas tangenciales que, por el teorema de Cauchy, si se toman momentos en el cubo diferencial se compensará la fuerza aplicada en la cara vertical con la fuerza tangencial aplicada en la cara horizontal del elemento.

Véase en el siguiente ejemplo:

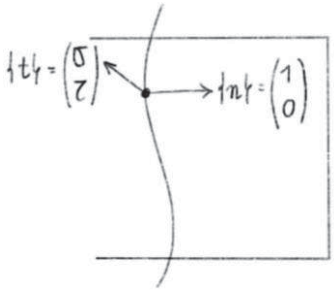


- **HALLAR EL VECTOR TENSIÓN EN CUALQUIER PUNTO**

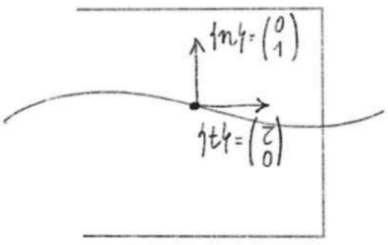
Por lo tanto, se sabe que el estado tensional se puede leer de la siguiente manera:

Si en la viga hay un corte vertical por el punto el cual quiero calcular, esta sección tendrá un vector normal $\{n\}$ que es perpendicular al plano de corte; pero si se observa el elemento del ejemplo anterior, aún quedan dos fuerzas de compresión (σ) y tangencial (τ) hacia la misma que forman parte del vector de tensión $\{t\}$.

Es decir, para el vector normal $\{n\} = \begin{pmatrix}1\\0\end{pmatrix}$, el vector de tensión que actúa en dicha sección será $\{t\} = \begin{pmatrix}\sigma\\\tau\end{pmatrix}$



Sin embargo, si el corte por ese mismo punto fuese un corte horizontal, el vector normal sería $\{n\} = \begin{pmatrix}0\\1\end{pmatrix}$, entonces la tensión vista en el elemento estudiado sería la rasante al plano de corte (τ), y totalmente perpendicular al vector $\{n\}$; es decir, el vector de tensión actuante en este plano horizontal sería $\{t\} = \begin{pmatrix}\tau\\0\end{pmatrix}$



Con estos dos ejemplos se puede hallar la matriz-tensor de tensiones (*T*):

Siendo $\{t\} = (T)\{n\}$, del primer corte vertical se obtiene:

$$\{t\} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}\{n\} \quad \Rightarrow \quad \begin{pmatrix} \sigma \\ \tau \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Donde la primera columna de la matriz-tensor de tensiones (*T*) es necesariamente el vector de tensión {*t*} y la segunda podrá ser cualquier cosa, pues va multiplicado por el valor 0 de la matriz del vector normal {*n*}, tal que:

$$\begin{pmatrix} \sigma \\ \tau \end{pmatrix} = \begin{pmatrix} \sigma & ? \\ \tau & ? \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Siendo en el segundo caso$\{t\} = (T)\{n\}$, del corte horizontal se obtiene:

$$\{t\} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}\{n\} \quad \Rightarrow \quad \begin{pmatrix} \tau \\ 0 \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Donde la segunda columna de la matriz-tensor de tensiones (*T*) es necesariamente el vector de tensión {*t*} y la primera podrá ser cualquier cosa, pues va multiplicado por el valor 0 de la matriz del vector normal {*n*}, tal que:

$$\begin{pmatrix} \tau \\ 0 \end{pmatrix} = \begin{pmatrix} ? & \tau \\ ? & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Como se puede apreciar, en ambos ejemplos se definen unas matrices-tensor de tensiones (*T*) incompletas pero que se complementan la una a la hora, obteniendo finalmente una matriz de tensor de tensiones (*T*) válida para toda la viga tal que:

$$(T) = \begin{pmatrix} \sigma & \tau \\ \tau & 0 \end{pmatrix}$$

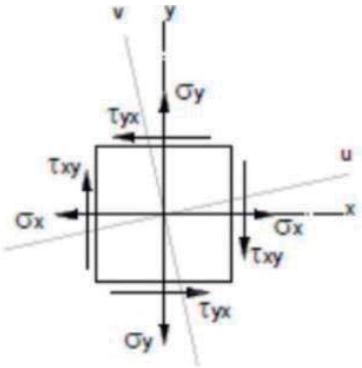
La matriz-tensor de tensiones es la matriz de una aplicación lineal donde a cada dirección perpendicular al plano de corte le corresponde una tensión.

Con esta matriz queda perfectamente definido cual es el comportamiento tensional en cada punto de la viga.

Se ha de tener en cuenta que para dos de las direcciones en las que se orienta el vector normal, existirán unas tensiones que son paralelas al mismo. A estos casos se les conoce como direcciones principales, y por lo tanto en esos puntos el vector tensión es un múltiplo de el vector normal; es decir, que están relacionados por un número que es el multiplicador.

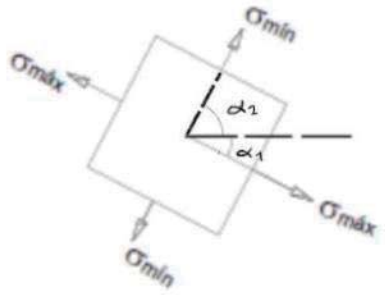
El número o números que multiplican al vector normal para obtener el vector tensión son los conocidos como valores propios y son precisamente la tensión de compresión y de tracción.

Es decir, del ejemplo que teníamos anteriormente, ahora podemos obtener este:



Se observa que este estado puede ser sustituido por una compresión y una tracción, ambas perpendiculares entre sí. Estos valores son el valor principal de compresión (λ_1), y la tensión principal de tracción (λ_2).

Esas direcciones en las que no existen tensiones tangenciales, son las direcciones principales y se definen por sus ángulos (α_1 y α_2).



Diagonalización de la matriz-tensor de tensiones.

Este proceso se obtiene partiendo de:

$$(T) = \begin{pmatrix} \sigma & \tau \\ \tau & 0 \end{pmatrix}$$

Aquella situación en la que el vector normal y el vector de tensiones son múltiplos uno de otro; esto quiere decir que el vector tensión sería igual a un múltiplo del vector normal $\{t\} = \lambda\,\{n\}$, pero sabemos que por la definición de la ecuación tensorial el vector de tensión es $\{t\} = (T)\{n\}$, por lo tanto, quedaría:

$$(T)\{n\} = \lambda\,\{n\} \quad \Rightarrow \quad (T)\{n\} - \lambda\,\{n\} = 0 \quad \Rightarrow \quad [(T) - \lambda(I)]\,\{n\} = 0$$

Lo cual resulta en un sistema de ecuaciones homogéneo cuya solución no trivial existe cuando el determinante de la matriz dad por $[(T) - \lambda(I)] = (M)$ sea igual a cero:

$$|(T) - \lambda(I)| = 0 \quad \Rightarrow \quad \left|\begin{pmatrix}\sigma & \tau\\ \tau & 0\end{pmatrix} - \begin{pmatrix}\lambda & 0\\ 0 & \lambda\end{pmatrix}\right| = 0 \quad \Rightarrow \quad \begin{vmatrix}\sigma - \lambda & \tau\\ \tau & -\lambda\end{vmatrix} = 0$$

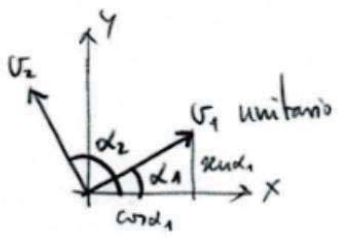
$$-\lambda\,\sigma + \lambda^2 - \tau^2 = 0 \quad \Rightarrow \quad \lambda^2 - \lambda\,\sigma - \tau^2 = 0$$

Esto es una simple ecuación de segundo grado de la cual obtengo dos soluciones; hay dos escalares que son el coeficiente de equivalencia entre el vector normal y el vector tensiones:

Tensión principal de compresión $\quad \lambda_1 = \dfrac{\sigma}{2} - \sqrt{\dfrac{\sigma^2}{4} + \tau^2}$

Tensión principal de tracción $\quad \lambda_2 = \dfrac{\sigma}{2} + \sqrt{\dfrac{\sigma^2}{4} + \tau^2}$

Una vez obtenidas estas tensiones hay que sacar cuales son las direcciones, que se obtienen volviendo al punto de la ecuación $[(T) - \lambda(I)]\,\{n\} = 0$, para sacar qué vectores normales son los que corresponden a lo visto anteriormente.



Siendo así que el vector normal $\{v1\}$ cumple que:

$$[(T) - \lambda_1(I)]\{v1\} = 0 \quad \Rightarrow \quad \begin{pmatrix} \sigma - \lambda_1 & \tau \\ \tau & -\lambda_1 \end{pmatrix} \begin{pmatrix} cos\ \alpha_1 \\ sen\ \alpha_1 \end{pmatrix} = 0$$

Desarrollando la segunda ecuación:

$$\tau \cdot cos\ \alpha_1 - \lambda_1 \cdot sen\ \alpha_1 = 0 \quad \Rightarrow \quad \tau - \lambda_1 \cdot tg\ \alpha_1 = 0 \quad \Rightarrow \quad tg\ \alpha_1 = \frac{\tau}{\lambda_1}$$

Se obtienen tanto α_1 como α_2:

$$\alpha_1 = arctg \frac{\tau}{\lambda_1} \quad \Rightarrow \quad \alpha_2 = \alpha_1 + 90^o$$

También puede obtenerse α_2 a partir de su autovalor, análogamente:

$$\alpha_2 = arctg \frac{\tau}{\lambda_2}$$

Según esta definición, α_i no está definido cuando $\lambda_2 = 0$, pues es un denominador nulo.

Pero, ¿cuándo se da esta posibilidad?

α_1 no está definido cuando $\lambda_1 = 0$;

$$0 = \frac{\sigma}{2} - \sqrt{\frac{\sigma^2}{4} + \tau^2} \quad \Rightarrow \quad \frac{\sigma}{2} = \sqrt{\frac{\sigma^2}{4} + \tau^2} \quad \Rightarrow \quad \sigma = 2\sqrt{\frac{\sigma^2}{4} + \tau^2} \quad \Rightarrow$$

$$\Rightarrow \quad \sigma = \sqrt{4\frac{\sigma^2}{4} + 4\tau^2} \quad \Rightarrow \quad \sigma = \sqrt{\sigma^2 + 4\tau^2}$$

Si $\sigma < 0$, esta igualdad es imposible, puesto que el término de la derecha es positivo. Si $\sigma \geq 0$, entonces:

$$\sigma^2 = \sigma^2 + 4\tau^2$$

$$\tau = 0$$

Entonces

α_1 no está definido cuando: $\sigma \geq 0$

$\tau = 0$

Análogamente para α_2,

$$\lambda_2 = 0$$

$$0 = \frac{\sigma}{2} + \sqrt{\frac{\sigma^2}{4} + \tau^2} \quad \Rightarrow \quad -\sigma = \sqrt{\sigma^2 + 4\tau^2}$$

Si $\sigma < 0$, no se produce la igualdad. Si $\sigma \leq 0$, entonces $\tau = 0$, luego:

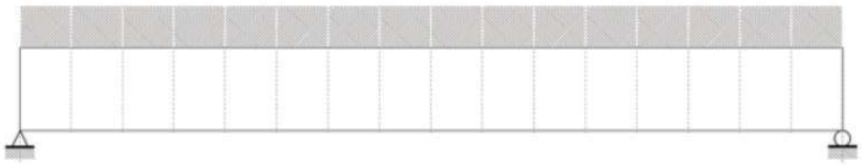
α_1 no está definido cuando: $\sigma \leq 0$

$\tau = 0$

- **FORMULA DE LINEAS ISOSTATICAS**

A continuación, se demuestra como bien se ha dicho anteriormente, que con el cálculo podemos llegar a lograr una representación exacta de dichas líneas, pero sería un trabajo arduo y bastante largo, obteniendo como resultado final una ecuación explícita imposible de integrar analíticamente, por eso es necesaria una solución discreta.

En el siguiente ejemplo se presenta una viga biapoyada con origen en el centro de geometría de la pieza.



Se obtienen las solicitaciones y tensiones de manera explícita en función de la geometría y la posición de la fibra *(x, y)*.

Cortante *V(x)*:

$$V(x) = \frac{q \cdot L}{2} - q\left(x + \frac{L}{2}\right) \quad \Rightarrow \quad V(x) = q\left(\frac{L}{2} - x - \frac{L}{2}\right) \quad \Rightarrow \quad V(x) = -q$$

Momento *M(x)*:

$$M_{(x)} = \frac{q\,L}{2}\left(x + \frac{L}{2}\right) - \frac{q\left(x + \frac{L}{2}\right)^2}{2} \quad \Rightarrow \quad M_{(x)} = \frac{q\,Lx}{2} + \frac{q\,L^2}{4} - \frac{q}{2}\left(x^2 + \frac{L^2}{4} + 2x\frac{L}{2}\right) \Rightarrow$$

$$M_{(x)} = q\left(\frac{Lx}{2} + \frac{L^2}{4} - \frac{x^2}{2} - \frac{L^2}{8} - \frac{Lx}{2}\right) \quad \Rightarrow \quad M_{(x)} = q\left(-\frac{x^2}{2} + \frac{L^2}{8}\right) \quad \Rightarrow$$

$$\Rightarrow \quad M_{(x)} = \frac{q}{2}\left(\frac{L^2}{4} - x^2\right) \quad \Rightarrow \quad M_{(x)} = \frac{qL^2}{8} - \frac{qx^2}{2} \quad \Rightarrow \quad M_{(x)} = -\frac{q}{2}x^2 + \frac{qL^2}{8}$$

Se sabe que:

LEY DE COLLIGNON $$\tau(x,y) = \frac{V(x) \cdot S(y)}{I\,b}$$

Depende de variables *(x, y)*

LEY DE NAVIER

$$\sigma(x,y) = -\frac{M(x)}{I}\, y$$

Depende de variables *(x, y)*

- **TENSIÓN NORMAL, σ**

$$\sigma(x,y) = -\frac{M(x)}{I}\, y \quad \Rightarrow \quad \sigma(x,y) = \frac{\frac{q}{2}x^2 - \frac{qL^2}{8}}{I}\, y \quad \Rightarrow \quad \sigma(x,y) = \frac{q}{2\,I}x^2 y - \frac{qL^2}{8\,I}y$$

- **TENSIÓN TANGENCIAL, τ**

$$\tau(x,y) = \frac{V(x)\cdot S(y)}{I\, b} \quad \Rightarrow \quad \tau_{(x,y)} = \frac{-q\, x\, (\frac{b\, h^2}{8} - \frac{b\, y^2}{2})}{b\, I} \quad \Rightarrow$$

$$\Rightarrow \quad \tau_{(x,y)} = -\frac{q\, b\, h^2}{8\, b\, I}x + \frac{q\, b}{2\, b\, I}x\, y^2 \quad \Rightarrow \quad \tau_{(x,y)} = \frac{q}{2\, I}x\, y^2 - \frac{q\, h^2}{8\, I}x$$

Expresando las direcciones principales como la solución de la ecuación

$tg2\alpha = \frac{2\tau}{\sigma}$, y haciendo $tg2\alpha = \frac{2tg\alpha}{1-tg^2\alpha}$, queda:

$$\frac{2tg\alpha}{1-tg^2\alpha} = \frac{2\tau}{\sigma} \quad \Rightarrow \quad 1 - tg^2\alpha = \frac{\sigma}{2\tau}\cdot 2\, tg\alpha \quad \Rightarrow \quad 1 - tg^2\alpha = \frac{\sigma}{\tau}\cdot\, tg\alpha \quad \Rightarrow$$

$$\Rightarrow \quad 1tg^2\alpha + \frac{\sigma}{\tau}\cdot\, tg\alpha - 1 = 0$$

Se integra $tg\alpha$ como la pendiente de la línea (función) isostática en un punto *x*, luego si la isostática es *y(x)*, entonces $y'(x) = tg\alpha$:

$$y'^2 + \frac{\sigma}{\tau}y' - 1 = 0$$

Cuya solución como ecuación de segundo grado en *y'* es:

$$y' = \frac{-\frac{\sigma}{\tau} \pm \sqrt{\frac{\sigma^2}{\tau^2} + 4}}{2} \quad \Rightarrow \quad y' = -\frac{\sigma}{2\tau} \pm \sqrt{\frac{\sigma^2}{4\tau^2} + 1} \quad \Rightarrow$$

$$\Rightarrow \quad y'(x) = -\frac{\sigma_{(x,y)}}{2\tau_{(x,y)}} \pm \sqrt{\frac{\sigma^2{}_{(x,y)}}{4\,\tau^2{}_{(x,y)}} + 1} \quad \Rightarrow$$

$$\Rightarrow \quad y'(x) = -\frac{\frac{q}{2I}x^2y - \frac{qL^2}{8I}y}{2\frac{q}{2I}x\,y^2 - 2\frac{q\,h^2}{8I}x} \pm \sqrt{\frac{\left(\frac{q}{2I}x^2y - \frac{qL^2}{8I}y\right)^2}{4\,(\frac{q}{2I}x\,y^2 - \frac{q\,h^2}{8I}x)^2} + 1} \quad \Rightarrow$$

$$\Rightarrow \quad y'(x) = -\frac{\frac{L^2}{8}y - \frac{1}{2}x^2y}{\frac{h^2}{4}x - x\,y^2} \pm \sqrt{\frac{\frac{1}{4}x^4\,y^2 + \frac{L^2}{64}y^2 - \frac{L^2}{8}x^2\,y^2}{4\left(\frac{1}{4}x\,y^4 + \frac{h^4}{64}x^2 - \frac{h^2}{8}x^2\,y^2\right)} + 1} \quad \Rightarrow$$

$$\Rightarrow \quad y'(x) = -\frac{\frac{L^2}{8}y - \frac{1}{2}x^2y}{\frac{h^2}{4}x - x\,y^2} \pm \sqrt{\frac{\frac{1}{4}x^4\,y^2 + \frac{L^2}{64}y^2 - \frac{L^2}{8}x^2\,y^2}{x^2\,y^4 + \frac{h^4}{16}x^2 - \frac{h^2}{2}x^2\,y^2} + 1}$$

Como ya se ha comentado previamente, esta es una ecuación diferencial explícita imposible de integrar analíticamente, por eso se decide continuar el problema mediante un programa informático que permita obtener una solución discreta.

PROGRAMACIÓN DE LA HERRAMIENTA

• POR QUÉ PYTHON

Como bien ya se ha comentado anteriormente en estas páginas, elegimos el software de programación Python ya que es un programa de código abierto y nos permite obtener tanto numérica como gráficamente sin ningún tipo de problema el trabajo que queremos llevar a cabo.

A la hora de escribir el código se asemeja en muchos parámetros a otros programas que ya se sabían utilizar como Matlab, wxMaxima, etc.

A continuación, se muestra cómo se ha llevado a cabo la escritura de la resolución de dicho problema en cuestión, así como el comentario sobre el trabajo realizado en el programa Python.

• SCRIPT

```
# Import modules
import math
import matplotlib.pyplot as plt
import matplotlib.patches as ptch
import numpy as np
import scipy.optimize as opt
import scipy.misc as msc
import sympy as sp


# Mathematical fix values
x = sp.Symbol('x')
# Definir la x como variable de las funciones solicitación, para poder expresar
V(x) como derivada de M(x)
pi = math.pi


#==========================================#
#================ USER INPUT ==============#
#==========================================#

# Input geometry
L1 = 0      # [m]
L2 = 0      # [m]
L3 = 0      # [m]

b = 0.30    # [m]
h = 1.00    # [m]

# Input loads
q1 = 0      # [kN/m]
q2 = 0      # [kN/m]
q3 = 0      # [kN/m]

P1 = 0      # [kN]
P2 = 0      # [kN]
P3 = 0      # [kN]
```

```
d1 = 0      # [m]
d2 = 0      # [m]
d3 = 0      # [m]

# Parameters for the appearence of graphic
distortion_Y = 1.5
# Distortion of Y axes in order to better see the lines inside the beam when h <<
L, to the detriment of orthogonality
equidist_left = h/20
# Chosen distance between initial points in the established left section
equidist_mid = h/20
# Chosen distance between initial points in the established middle section
equidist_right = h/20
# Chosen distance between initial points in the established right section
plot_loads = 0
# 0: No; 1: Yes
plot_internal_forces = 0
# 0: No; 1: Yes
plot_isostatic_lines = 1
scale_M = 0.01
# Scale of kNm respect to m on the y axis
scale_V = 0.01
# Scale of kN respect to m on the y axis

# Parameters for the accuracy of the graphic
L0 = L1 + L2 + L3
# Not an input parameter but necessary for defining mod_v
mod_v = L0/300
# Distance between each point in every line
dx = 1e-6
# Accuracy for the calculation of the derivative in V(x). It can cause the plot of
additional, undesired lines
'if necessary, modify factor multiplying pixels in variable linewidth_max'

#============================================#
#============================================#
#============================================#


# Useful geometry values [m]
x_A = 0                                         # Left cantilever
x_a = d1                                        # Position of the left
concentrated load
x_B = L1                                        # Left restaint
x_b = L1 + d2                                   # Position of the middle
concentrated load
x_C = L1 + L2                                   # Right restaint
x_c = L1 + L2 + d3                              # Position of the ight
concentrated load
x_D = L1 + L2 + L3                              # Right cantilever
bottom = -h/2                                   # Bottom fibre
top = h/2                                       # Top fibre

# Mechanical cross section properties
A = b*h                                         # Area [m**2]
I = b*h**3/12                                   # Inertia [m**4]
def S(y):
    return b*h**2/8 - b*y**2/2
# First order static moment of the cross section [m**3]


# Reactions in the restraints [kN]
```

```
    R_C = (-P1*(L1-d1) + P2*d2 + P3*(L2+d3) - q1*L1**2/2 + q2*L2**2/2 +
    q3*L3*(L2+L3/2)) / L2
    R_B = P1 + P2 + P3 + q1*L1 + q2*L2 + q3*L3 - R_C


    # Cross section internal forces

    def M(x):                           # Bending moment [kN·m]
        if x < x_A:
            return 0
        elif x <= x_a:
            return -q1*x**2/2
        elif x <= x_B:
            return -q1*x**2/2 - P1*(x-x_a)
        elif x <= x_b:
            return -q1*L1*(x-L1/2) - P1*(x-x_a) + R_B*(x-x_B) - q2*(x-x_B)**2/2
        elif x <= x_C:
            return -q1*L1*(x-L1/2) - P1*(x-x_a) + R_B*(x-x_B) - q2*(x-x_B)**2/2 -
    P2*(x-x_b)
        elif x <= x_c:
            return -q1*L1*(x-L1/2) - P1*(x-x_a) + R_B*(x-x_B) - q2*L2*(x-L1-L2/2) -
    P2*(x-x_b) + R_C*(x-x_C) - q3*(x-x_C)**2/2
        elif x <= x_D:
            return -q1*L1*(x-L1/2) - P1*(x-x_a) + R_B*(x-x_B) - q2*L2*(x-L1-L2/2) -
    P2*(x-x_b) + R_C*(x-x_C) - q3*(x-x_C)**2/2 - P3*(x-x_c)
        else:
            return 0

    # def Mabs(x):
    #     return abs(M(x))

    # def V(x):                         # Shear force [kN] explicit expression.
    Comportamiento erróneo, buscar si está ligado a puntos de no continuidad
    #     if x < x_A:
    #         return 0
    #     elif x <= x_a:
    #         return -q1*x
    #     elif x <= x_B:
    #         return -q1*x - P1
    #     elif x <= x_b:
    #         return -q1*L1 - P1 + R_B - q2*(x-x_B)
    #     elif x <= x_C:
    #         return -q1*L1 - P1 + R_B - q2*(x-x_B) - P2
    #     elif x <= x_c:
    #         return -q1*L1 - P1 + R_B - q2*L2 - P2 + R_C - q3*(x-x_C)
    #     elif x <= x_D:
    #         return -q1*L1 - P1 + R_B - q2*L2 - P2 + R_C - q3*(x-x_C)- P3
    #     else:
    #         return 0


    # def V(x):                           # Shear force [kN] as derivative of M(x)
    def V_dM(x):                        # Shear force [kN] as derivative of M(x)
        if x < x_A:
            return 0
        elif x <= x_a:
            return msc.derivative(M, x, dx=dx)
        elif x <= x_B:
            return msc.derivative(M, x, dx=dx)
        elif x <= x_b:
            return msc.derivative(M, x, dx=dx)
        elif x <= x_C:
            return msc.derivative(M, x, dx=dx)
        elif x <= x_c:
```

```
            return msc.derivative(M, x, dx=dx)
        elif x <= x_D:
            return msc.derivative(M, x, dx=dx)
        else:
            return 0

    def V(x):
        if abs(V_dM(x)) < dx:
            return 0
        else:
            return V_dM(x)

    # Fibre stresses
    def sigma(x,y):                         # Normal stress
        return -M(x)*y/I*1E-3               # [N/mm**2]

    def tau(x,y):                           # Tangential stress
        return -V(x)*S(y)/(b*I) *1E-3       # [N/mm**2]


    # Principal stresses
    def lambda1(x,y):
    # Compression principal stress
        return (sigma(x,y)/2)-(((sigma(x,y)**2)/4)+tau(x,y)**2)**0.5    # [N/mm**2]

    def lambda2(x,y):
    # Tension principal stress
        return (sigma(x,y)/2)+(((sigma(x,y)**2)/4)+tau(x,y)**2)**0.5    # [N/mm**2]


    # Principal directions
    def alfa1(x,y):                                 # Compression direction
        return math.atan(tau(x,y)/lambda1(x,y))     # [rad]

    def alfa2(x,y):                                 # Tension direction
        return math.atan(tau(x,y)/lambda2(x,y))     # [rad]


    # Obtaining maximum principal tension for graphic thickness of the lines
    def Minv_abs(x):
    # Definition of the absolute value of the inverse of M(x) because there is no
    maximize tool available, only minimize
        if x >= x_A and x <= x_D:
            if M(x) == 0:
                return float("inf")         # Infinite value
            else:
                return abs(1/M(x))

    if q1 == 0 and P1 == 0:
        Section_Mmax_opt_left = x_B
    elif q1 == 0 and P1 > 0:
        Section_Mmax_opt_left = x_a
    else:
        Section_Mmax_opt_left = x_A

    if q1 == 0 and P1 == 0:
        Section_Mmax_opt_right = x_C - 1e-2
    elif q1 == 0 and P1 > 0:
        Section_Mmax_opt_right = x_c - 1e-2
    else:
        Section_Mmax_opt_right = x_D
```

```
    x_Mmax_left = opt.fmin(Minv_abs, Section_Mmax_opt_left, xtol=dx)
    # Searching the cross-section with maximum M(x) and sigma. As it finds the closest
    relative minimum instead of the absolute one, minimisation in each part of the
    beam must be carried out
    x_Mmax_mid = opt.fmin(Minv_abs, (x_B+x_C)/2, xtol=dx)
    x_Mmax_right = opt.fmin(Minv_abs, Section_Mmax_opt_right, xtol=dx)
    if Minv_abs(x_Mmax_left) == min(Minv_abs(x_Mmax_left), Minv_abs(x_Mmax_mid),
    Minv_abs(x_Mmax_right)):
        x_Mmax = x_Mmax_left
    elif Minv_abs(x_Mmax_mid) == min(Minv_abs(x_Mmax_left), Minv_abs(x_Mmax_mid),
    Minv_abs(x_Mmax_right)):
        x_Mmax = x_Mmax_mid
    else:
        x_Mmax = x_Mmax_right

    lambda_max = max(abs(lambda1(x_Mmax, top)), abs(lambda2(x_Mmax, top)), \
                     abs(lambda1(x_Mmax, bottom)), abs(lambda2(x_Mmax, bottom)))
    # Maximum principal stress of any sign


    # Obtaining abscissas of M(x)=0 in interior span for defining the three zones of
    plotting lines

    aa = q2/2
    # term a of the 2nd degree equation for both sides roots
    bb_B = q1*L1 + P1 - R_B - q2*x_B
    # term b of the 2nd degree equation for the left side root
    bb_C = q1*L1 + P1 - R_B - q2*x_B + P2
    # term b of the 2nd degree equation for the right side root
    cc_B = -q1*L1**2/2 - P1*x_a + R_B*x_B + q2*x_B**2/2
    # term c of the 2nd degree equation for the left side root
    cc_C = -q1*L1**2/2 - P1*x_a + R_B*x_B + q2*x_B**2/2 - P2*x_b
    # term c of the 2nd degree equation for the right side root

    if aa == 0:
    # M(x) linear if there is no constant load
        if bb_B != 0:
            x_M0_B = -cc_B/bb_B
        else:
            x_M0_B = (x_B + x_C)/2
        if bb_C != 0:
            x_M0_C = -cc_C/bb_C
        else:
            x_M0_C = (x_B + x_C)/2
    else:
        if bb_B**2 - 4*aa*cc_B < 0:
    # If there is no root, choose the sections of the restraints; if there is, choose
    the root contained in the interval
            x_M0_B_1 = (x_B + x_C)/2
            x_M0_B_2 = (x_B + x_C)/2
        else:
            x_M0_B_1 = (-bb_B + (bb_B**2 - 4*aa*cc_B)**0.5)/(2*aa)
            x_M0_B_2 = (-bb_B - (bb_B**2 - 4*aa*cc_B)**0.5)/(2*aa)
        if bb_C**2 - 4*aa*cc_C < 0:
            x_M0_C_1 = (x_B + x_C)/2
            x_M0_C_2 = (x_B + x_C)/2
        else:
            x_M0_C_1 = (-bb_C + (bb_C**2 - 4*aa*cc_C)**0.5)/(2*aa)
            x_M0_C_2 = (-bb_C - (bb_C**2 - 4*aa*cc_C)**0.5)/(2*aa)

        if x_B <= x_M0_B_1 and x_M0_B_1 <= (x_B + x_C)/2:
            x_M0_B = x_M0_B_1
        elif x_B <= x_M0_B_2 and x_M0_B_2 <= (x_B + x_C)/2:
            x_M0_B = x_M0_B_2
```

```
        else:
            x_M0_B = (x_B + x_C)/2

        if (x_B + x_C)/2 <= x_M0_C_1 and x_M0_C_1 <= x_C:
            x_M0_C = x_M0_C_1
        elif (x_B + x_C)/2 <= x_M0_C_2 and x_M0_C_2 <= x_C:
            x_M0_C = x_M0_C_2
        else:
            x_M0_C = (x_B + x_C)/2


    # Generate graphic
    fig_ratio = 3/2
    # Ratio of the graphic
    scale_pix = 10
    # Factor for visualization
    horiz_length = scale_pix*2
    # Pixels
    fig, ax = plt.subplots(figsize = (horiz_length, horiz_length/fig_ratio))
    # For ensuring equal scaling of axes
    ax.set_xlim(x_A-0.5, x_D+0.5)
    # Abscissa with 0.5 m margin in each side
    ax.set_ylim((-L0/2-0.5)/(distortion_Y*fig_ratio),
    (L0/2+0.5)/(distortion_Y*fig_ratio))  # Ordinate keeping the proportion
    # ax.spines['right'].set_color('none')
    # Hide the right axe
    # ax.spines['top'].set_color('none')
    # Hide the top axe
    plt.xlabel("x [m]", fontsize=15)
    plt.ylabel("y [m] ;   V(x) [" + str(scale_V) + " kN] ;   M(x) [" + str(scale_M) +
    " kNm]", fontsize=15)


    #====================#
    # BUILDING THE LINES #
    #====================#

    if plot_isostatic_lines == 1 :

        # Generic graphic options for isostatic lines

        if x_Mmax > x_B + dx and x_Mmax < x_C - dx:
            equidist = equidist_mid
        elif x_Mmax < x_B:
            equidist = equidist_left
        else:
            equidist = equidist_right
    # Selecting the portion of the beam in which the maximum stress is attained
        linewidth_max = 100*distortion_Y*equidist
    # Maximum width of the lines adjusted for not overlapping if maximum stresses are
    placed along the ordinate axe


        # LEFT PART

        Section_ini_left = x_B
    # Chosen initial abscissa for building the lines
        n_lines = math.ceil(h/equidist_left)
    # Number of lines

        # Compression lines
        i = 0
    # ith sostatic line
        while i <= n_lines:
    # Going across the fibers from top to bottom
```

```
            # Left branch
            a = (Section_ini_left, top - i*h/n_lines)
    # Initial fiber (top)
            pointsX = [a[0]]
    # X coordinate of initial point
            pointsY = [a[1]]
    # Y coordinate of initial point
            while a[1]>= bottom and a[1]<= top and a[0]>= x_A and a[0]<= x_M0_B \
                and (not (sigma(a[0],a[1]) >= 0 and tau(a[0],a[1]) == 0)):
    # Building the left line within the boundaries of the beam and skipping points
    where lambda1 = 0
                alfa_1 = alfa1(a[0],a[1])
    # Angle at current point
                v = -mod_v*np.array([math.cos(alfa_1), math.sin(alfa_1)])
    # Vector director for next point (array)
                a = tuple(np.array(a) + v)
    # Next point as tuple
                pointsX.append(a[0])
    # List of X coordinates of current and next point
                pointsY.append(a[1])
    # List of Y coordinates of current and next point
                ax.plot(pointsX, pointsY, c = 'r', \
                        linewidth =
    max(linewidth_max*abs(lambda1(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
    # Plot segment of line
                pointsX = [a[0]]
    # Removing current point from coordinates X
                pointsY = [a[1]]
    # Removing current point from coordinates Y
            # Right branch
            a = (Section_ini_left, top - i*h/n_lines)
            pointsX = [a[0]]
            pointsY = [a[1]]
            while a[1]>= bottom and a[1]<= top and a[0]>= x_A and a[0]<= x_M0_B \
                and (not (sigma(a[0],a[1]) >= 0 and tau(a[0],a[1]) == 0)):
                alfa_1 = alfa1(a[0],a[1])
                v = mod_v*np.array([math.cos(alfa_1), math.sin(alfa_1)])
                a = tuple(np.array(a) + v)
                pointsX.append(a[0])
                pointsY.append(a[1])
                ax.plot(pointsX, pointsY, c = 'r', \
                        linewidth =
    max(linewidth_max*abs(lambda1(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]

            i += 1
    # Next line (iteration)

        # Tension lines
        i = 0
        while i <= n_lines:
            # Left branch
            a = (Section_ini_left, top - i*h/n_lines)
            pointsX = [a[0]]
            pointsY = [a[1]]
            while a[1]>= bottom and a[1]<= top and a[0]>= x_A and a[0]<= x_M0_B \
                and (not (sigma(a[0],a[1]) <= 0 and tau(a[0],a[1]) == 0)):
                alfa_2 = alfa2(a[0],a[1])
                v = -mod_v*np.array([math.cos(alfa_2), math.sin(alfa_2)])
                a = tuple(np.array(a) + v)
```

```
                pointsX.append(a[0])
                pointsY.append(a[1])
                ax.plot(pointsX, pointsY, c = 'b', \
                        linewidth =
    max(linewidth_max*abs(lambda2(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]
            # Right branch
            a = (Section_ini_left, top - i*h/n_lines)
            pointsX = [a[0]]
            pointsY = [a[1]]
            while a[1]>= bottom and a[1]<= top and a[0]>= x_A and a[0]<= x_M0_B \
                and (not (sigma(a[0],a[1]) <= 0 and tau(a[0],a[1]) == 0)):
                alfa_2 = alfa2(a[0],a[1])
                v = mod_v*np.array([math.cos(alfa_2), math.sin(alfa_2)])
                a = tuple(np.array(a) + v)
                pointsX.append(a[0])
                pointsY.append(a[1])
                ax.plot(pointsX, pointsY, c = 'b', \
                        linewidth =
    max(linewidth_max*abs(lambda2(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]

            i += 1


        # MIDDLE PART

        if float(x_Mmax) > x_B + dx and float(x_Mmax) < x_C - dx:
            Section_ini_mid = float(x_Mmax_mid)
    # Chosen initial abscissa for building the lines
        else:
            Section_ini_mid = (x_B+x_C)/2
        n_lines = math.ceil(h/equidist_mid)
    # Number of lines

        # Compression lines
        i = 0
    # ith sostatic line
        while i <= n_lines:
    # Going across the fibers from top to bottom
            # Left branch
            a = (Section_ini_mid, top - i*h/n_lines)
    # Initial fiber (top)
            pointsX = [a[0]]
    # X coordinate of initial point
            pointsY = [a[1]]
    # Y coordinate of initial point
            while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_B and a[0]<= x_M0_C \
                and (not (sigma(a[0],a[1]) >= 0 and tau(a[0],a[1]) == 0)):
    # Building the left line within the boundaries of the beam and skipping points
    where lambda1 = 0
                alfa_1 = alfa1(a[0],a[1])
    # Angle at current point
                v = -mod_v*np.array([math.cos(alfa_1), math.sin(alfa_1)])
    # Vector director for next point (array)
                a = tuple(np.array(a) + v)
    # Next point as tuple
                pointsX.append(a[0])
    # List of X coordinates of current and next point
```

```
            pointsY.append(a[1])
# List of Y coordinates of current and next point
            ax.plot(pointsX, pointsY, c = 'r', \
                    linewidth =
max(linewidth_max*abs(lambda1(a[0],a[1]))/lambda_max,1), \
                    linestyle="-", antialiased=True)
# Plot segment of line
            pointsX = [a[0]]
# Removing current point from coordinates X
            pointsY = [a[1]]
# Removing current point from coordinates Y
        # Right branch
        a = (Section_ini_mid, top - i*h/n_lines)
        pointsX = [a[0]]
        pointsY = [a[1]]
        while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_B and a[0]<= x_M0_C \
            and (not (sigma(a[0],a[1]) >= 0 and tau(a[0],a[1]) == 0)):
            alfa_1 = alfa1(a[0],a[1])
            v = mod_v*np.array([math.cos(alfa_1), math.sin(alfa_1)])
            a = tuple(np.array(a) + v)
            pointsX.append(a[0])
            pointsY.append(a[1])
            ax.plot(pointsX, pointsY, c = 'r', \
                    linewidth =
max(linewidth_max*abs(lambda1(a[0],a[1]))/lambda_max,1), \
                    linestyle="-", antialiased=True)
            pointsX = [a[0]]
            pointsY = [a[1]]

        i += 1
# Next line (iteration)

    # Tension lines
    i = 0
    while i <= n_lines:
        # Left branch
        a = (Section_ini_mid, top - i*h/n_lines)
        pointsX = [a[0]]
        pointsY = [a[1]]
        while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_B and a[0]<= x_M0_C \
            and (not (sigma(a[0],a[1]) <= 0 and tau(a[0],a[1]) == 0)):
            alfa_2 = alfa2(a[0],a[1])
            v = -mod_v*np.array([math.cos(alfa_2), math.sin(alfa_2)])
            a = tuple(np.array(a) + v)
            pointsX.append(a[0])
            pointsY.append(a[1])
            ax.plot(pointsX, pointsY, c = 'b', \
                    linewidth =
max(linewidth_max*abs(lambda2(a[0],a[1]))/lambda_max,1), \
                    linestyle="-", antialiased=True)
            pointsX = [a[0]]
            pointsY = [a[1]]
        # Right branch
        a = (Section_ini_mid, top - i*h/n_lines)
        pointsX = [a[0]]
        pointsY = [a[1]]
        while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_B and a[0]<= x_M0_C \
            and (not (sigma(a[0],a[1]) <= 0 and tau(a[0],a[1]) == 0)):
            alfa_2 = alfa2(a[0],a[1])
            v = mod_v*np.array([math.cos(alfa_2), math.sin(alfa_2)])
            a = tuple(np.array(a) + v)
            pointsX.append(a[0])
            pointsY.append(a[1])
```

```
                ax.plot(pointsX, pointsY, c = 'b', \
                        linewidth =
max(linewidth_max*abs(lambda2(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]

            i += 1


        # RIGHT PART

        Section_ini_right = x_C
# Chosen initial abscissa for building the lines
        n_lines = math.ceil(h/equidist_right)
# Number of lines

        # Compression lines
        i = 0
# ith sostatic line
        while i <= n_lines:
# Going across the fibers from top to bottom
            # Left branch
            a = (Section_ini_right, top - i*h/n_lines)
# Initial fiber (top)
            pointsX = [a[0]]
# X coordinate of initial point
            pointsY = [a[1]]
# Y coordinate of initial point
            while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_C and a[0]<= x_D \
                and (not (sigma(a[0],a[1]) >= 0 and tau(a[0],a[1]) == 0)):
# Building the left line within the boundaries of the beam and skipping points
where lambda1 = 0
                alfa_1 = alfa1(a[0],a[1])
# Angle at current point
                v = -mod_v*np.array([math.cos(alfa_1), math.sin(alfa_1)])
# Vector director for next point (array)
                a = tuple(np.array(a) + v)
# Next point as tuple
                pointsX.append(a[0])
# List of X coordinates of current and next point
                pointsY.append(a[1])
# List of Y coordinates of current and next point
                ax.plot(pointsX, pointsY, c = 'r', \
                        linewidth =
max(linewidth_max*abs(lambda1(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
# Plot segment of line
                pointsX = [a[0]]
# Removing current point from coordinates X
                pointsY = [a[1]]
# Removing current point from coordinates Y
            # Right branch
            a = (Section_ini_right, top - i*h/n_lines)
            pointsX = [a[0]]
            pointsY = [a[1]]
            while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_C and a[0]<= x_D \
                and (not (sigma(a[0],a[1]) >= 0 and tau(a[0],a[1]) == 0)):
                alfa_1 = alfa1(a[0],a[1])
                v = mod_v*np.array([math.cos(alfa_1), math.sin(alfa_1)])
                a = tuple(np.array(a) + v)
                pointsX.append(a[0])
                pointsY.append(a[1])
```

```
                ax.plot(pointsX, pointsY, c = 'r', \
                        linewidth =
    max(linewidth_max*abs(lambda1(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]

            i += 1
    # Next line (iteration)

        # Tension lines
        i = 0
        while i <= n_lines:
            # Left branch
            a = (Section_ini_right, top - i*h/n_lines)
            pointsX = [a[0]]
            pointsY = [a[1]]
            while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_C and a[0]<= x_D \
                and (not (sigma(a[0],a[1]) <= 0 and tau(a[0],a[1]) == 0)):
                alfa_2 = alfa2(a[0],a[1])
                v = -mod_v*np.array([math.cos(alfa_2), math.sin(alfa_2)])
                a = tuple(np.array(a) + v)
                pointsX.append(a[0])
                pointsY.append(a[1])
                ax.plot(pointsX, pointsY, c = 'b', \
                        linewidth =
    max(linewidth_max*abs(lambda2(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]
            # Right branch
            a = (Section_ini_right, top - i*h/n_lines)
            pointsX = [a[0]]
            pointsY = [a[1]]
            while a[1]>= bottom and a[1]<= top and a[0]>= x_M0_C and a[0]<= x_D \
                and (not (sigma(a[0],a[1]) <= 0 and tau(a[0],a[1]) == 0)):
                alfa_2 = alfa2(a[0],a[1])
                v = mod_v*np.array([math.cos(alfa_2), math.sin(alfa_2)])
                a = tuple(np.array(a) + v)
                pointsX.append(a[0])
                pointsY.append(a[1])
                ax.plot(pointsX, pointsY, c = 'b', \
                        linewidth =
    max(linewidth_max*abs(lambda2(a[0],a[1]))/lambda_max,1), \
                        linestyle="-", antialiased=True)
                pointsX = [a[0]]
                pointsY = [a[1]]

            i += 1



    #=========================#
    # PLOT GEOMETRY AND LOADS #
    #=========================#


    # Plot beam perimeter, axe and sections
    ax.add_patch(ptch.Rectangle((x_A, bottom), x_D, h, fill=False, linewidth =
    0.2*scale_pix))  # Perimeter
    ax.plot((x_A,x_D),(0,0), 'k--', linewidth = 0.15*scale_pix)
    # X axe
    ax.plot((x_M0_B,x_M0_B),(bottom,top), 'k:', linewidth = 0.15*scale_pix)
    # Left section where density of lines changes
```

```
ax.plot((x_M0_C,x_M0_C),(bottom,top), 'k:', linewidth = 0.15*scale_pix)
# Right section where density of lines changes
ax.plot((x_B,x_B),(bottom,top), 'k--', linewidth = 0.15*scale_pix)
# x_B section
ax.plot((x_C,x_C),(bottom,top), 'k--', linewidth = 0.15*scale_pix)
# x_C section



# Plot restraints
ax.add_patch(ptch.Polygon([[x_B-0.2,bottom-0.2],[x_B,bottom],[x_B+0.2,bottom-
0.2]], closed=True,fill=False))   # plot restraint B
ax.add_patch(ptch.Polygon([[x_C-0.2,bottom-0.2],[x_C,bottom],[x_C+0.2,bottom-
0.2]], closed=True,fill=False))   # plot restraint C


# Plot loads

if plot_loads == 1:

    if q1 > 0:
# plot q1

ax.add_patch(ptch.Polygon([[x_A,top],[x_A,top+q1/100],[x_B,top+q1/100],[x_B,top]],
closed=False,fill=False,linewidth=0.1*scale_pix))
        ax.arrow(x_A,top+0.2,0,-
0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0.1*scal
e_pix)
        ax.arrow(x_B,top+0.2,0,-
0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0.1*scal
e_pix)
        ax.annotate(q1,xy=((x_A+x_B)/2-0.3, top+q1/100+0.05), xytext=(0, 0),
textcoords='offset points', fontsize=16)            # Cambiar quizá los puntos de
anotación por desplazamiento de pixels, siguiente coma
        ax.annotate('kNm',xy=((x_A+x_B)/2-0.3, top+q1/100+0.05), xytext=(25, 0),
textcoords='offset points', fontsize=16)
    if q2 > 0:
# plot q2

ax.add_patch(ptch.Polygon([[x_B,top],[x_B,top+q2/100],[x_C,top+q2/100],[x_C,top]],
closed=False,fill=False,linewidth=0.1*scale_pix))
        ax.arrow(x_B,top+0.2,0,-
0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0.1*scal
e_pix)
        ax.arrow(x_C,top+0.2,0,-
0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0.1*scal
e_pix)
        ax.annotate(q2,xy=((x_B+x_C)/2-0.3, top+q2/100+0.05), xytext=(0, 0),
textcoords='offset points', fontsize=16)
        ax.annotate('kNm',xy=((x_B+x_C)/2-0.3, top+q2/100+0.05), xytext=(25, 0),
textcoords='offset points', fontsize=16)
    if q3 > 0:
# plot q3

ax.add_patch(ptch.Polygon([[x_C,top],[x_C,top+q3/100],[x_D,top+q3/100],[x_D,top]],
closed=False,fill=False,linewidth=0.1*scale_pix))
        ax.arrow(x_C,top+0.2,0,-
0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0.1*scal
e_pix)
        ax.arrow(x_D,top+0.2,0,-
0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0.1*scal
e_pix)
        ax.annotate(q3,xy=((x_C+x_D)/2-0.3, top+q3/100+0.05), xytext=(0, 0),
textcoords='offset points', fontsize=16)
        ax.annotate('kNm',xy=((x_C+x_D)/2-0.3, top+q3/100+0.05), xytext=(25, 0),
textcoords='offset points', fontsize=16)

    if P1 > 0:
```

```
        ax.arrow(x_a,top+P1/100,0,-
P1/100+0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0
.1*scale_pix)    #plot de P1
        ax.annotate(P1,xy=(x_a-0.2, top+P1/100+0.05), xytext=(0, 0),
textcoords='offset points', fontsize=16)
        ax.annotate('kN',xy=(x_a-0.2, top+P1/100+0.05), xytext=(25, 0),
textcoords='offset points', fontsize=16)
    if P2 > 0:
        ax.arrow(x_b,top+P2/100,0,-
P2/100+0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0
.1*scale_pix)    #plot de P2
        ax.annotate(P2,xy=(x_b-0.2, top+P2/100+0.05), xytext=(0, 0),
textcoords='offset points', fontsize=16)
        ax.annotate('kN',xy=(x_b-0.2, top+P2/100+0.05), xytext=(25, 0),
textcoords='offset points', fontsize=16)
    if P3 > 0:
        ax.arrow(x_c,top+P3/100,0,-
P3/100+0.1,head_width=0.05,head_length=0.1,fc="k",ec="k",clip_on=False,linewidth=0
.1*scale_pix)    #plot de P3
        ax.annotate(P3,xy=(x_c-0.2, top+P3/100+0.05), xytext=(0, 0),
textcoords='offset points', fontsize=16)
        ax.annotate('kN',xy=(x_c-0.2, top+P3/100+0.05), xytext=(25, 0),
textcoords='offset points', fontsize=16)


# Plot spans

ax.plot((x_A,x_B),(bottom-0.4,bottom-0.4), 'k-', linewidth = 0.05*scale_pix)
# L1 line
ax.plot((x_B,x_C),(bottom-0.4,bottom-0.4), 'k-', linewidth = 0.05*scale_pix)
# L2 line
ax.plot((x_C,x_D),(bottom-0.4,bottom-0.4), 'k-', linewidth = 0.05*scale_pix)
# L3 line

ax.plot((x_A,x_A),(bottom-0.35,bottom-0.45), 'k-', linewidth = 0.1*scale_pix)
# x_A auxiliar line
ax.plot((x_B,x_B),(bottom-0.35,bottom-0.45), 'k-', linewidth = 0.1*scale_pix)
# x_B auxiliar line
ax.plot((x_C,x_C),(bottom-0.35,bottom-0.45), 'k-', linewidth = 0.1*scale_pix)
# x_C auxiliar line
ax.plot((x_D,x_D),(bottom-0.35,bottom-0.45), 'k-', linewidth = 0.1*scale_pix)
# x_D auxiliar line

if L1 > 0:
    # ax.annotate('L1 = ',xy=(L1/2, -1.1*h), xytext=(-16, 0), textcoords='offset
points', fontsize=16)
    ax.annotate(L1,xy=(L1/2-0.15, bottom-0.35), xytext=(0, 0), textcoords='offset
points', fontsize=16)
    ax.annotate('m',xy=(L1/2-0.15, bottom-0.35), xytext=(25, 0),
textcoords='offset points', fontsize=16)
if L2 > 0:
    # ax.annotate('L2 = ',xy=(L1+L2/2, -1.1*h), xytext=(-16, 0),
textcoords='offset points', fontsize=16)
    ax.annotate(L2,xy=(L1+L2/2-0.15, bottom-0.35), xytext=(0, 0),
textcoords='offset points', fontsize=16)
    ax.annotate('m',xy=(L1+L2/2-0.15, bottom-0.35), xytext=(25, 0),
textcoords='offset points', fontsize=16)
if L3 > 0:
    # ax.annotate('L3 = ',xy=(L2+L1+L3/2, -1.1*h), xytext=(-16, 0),
textcoords='offset points', fontsize=16)
    ax.annotate(L3,xy=(L2+L1+L3/2-0.15, bottom-0.35), xytext=(0, 0),
textcoords='offset points', fontsize=16)
    ax.annotate('m',xy=(L2+L1+L3/2-0.15, bottom-0.35), xytext=(25, 0),
textcoords='offset points', fontsize=16)


    # Plot internal forces functions V(x) and M(x)


    t = np.arange(x_A, x_D, 0.01)
```

```
if plot_internal_forces == 1:
    y_M = []
    for i in range(len(t)):
        y_M.append(-scale_M*M(t[i]))
    ax.plot (t, y_M, c='dodgerblue', linewidth = 0.3*scale_pix)

    y_V = []
    for i in range(len(t)):
        y_V.append(scale_V*V(t[i]))
    ax.plot (t, y_V, c='g', linewidth = 0.3*scale_pix)
```

- **COMENTARIO SOBRE EL SCRIPT**

IMPORTACIÓN DE MÓDULOS

El programa Python funciona a través de módulos de programa que son lo que importan funciones las cuales ayudan a resolver las distintas necesidades que tenemos, como por ejemplo modulo matemático para resolver ecuaciones o módulos de ploteado que iremos comentando a lo largo de este trabajo.

Desde un principio, y después de haber elegido los módulos a través de los cuales vamos a resolver este estudio, lo primero que nos encontramos son los parámetros modificables por el usuario, los cuales definen:

Luces de la viga por tramos, llegando a tener hasta un máximo de tres tramos, así como la base y la altura del propio elemento estructural.

Cargas repartidas, cargas puntuales y las distancias por tramos a los que actúan esas cargas puntuales.

PARÁMETROS GEOMÉTRICOS PARA LA DEFINICIÓN DEL GRÁFICO O PLOTEADO FINAL

distorion_Y: es el valor que indica precisamente cuanto se distorsiona artificialmente el eje Y para que la representación gráfica de las líneas en el interior de la viga sea de una mayor calidad. Si el valor de este dato de escala gráfica fuera de 1; es decir representación de escala real de la viga en alzado, todo el dibujo de líneas isostáticas se vería más compacto. Bien cierto es que sería más real en su representación, pues la intersección entre líneas ha de ser en ángulo recto, pero en la didáctica se puede llegar a entender menos debido a la estrechez entre líneas, es por eso que existe este valor.

Los valores de: equidist_left, equidist_mid y equidist_rigth se utilizan para imponer la cantidad de líneas isostáticas que queremos obtener por tramo

Así los ejes estarán compensados y los dibujos serán muy intuitivos en la práctica.

Los demás valores: plot_loads, plot_internal_forces, son valores que se utilizan para representar la escala de las fuerzas que se representan en la gráfica; scale_M y scale_V, son igualmente los valores que se modifican en cada caso para que la obtención de la representación gráfica de la ley de cortante y momento siempre quede dentro del cuadro de dibujo y no se salga.

En los diferentes ejemplos variarán en algunos casos bastantes para garantizar así que obtengamos toda la información relevante en el mismo dibujo.

Los siguientes parámetros también entra a formar parte de la definición de la gráfica de la de los resultados que queremos obtener de la viga:

L0 es la longitud total de la viga.

El parámetro mod_v es el encargado de repartir la distancia entre puntos dentro de las líneas isostáticas; si dividimos la altura de la sección por un número muy grande es entonces cuando obtenemos una línea mucho más curva y más exacta. En nuestras representaciones finales se decide dividir el canto de la viga entre trescientos, resultando así un dibujo más atractivo y bastante limpio.

El valor dx no es más que la cantidad de error a la que queremos acercar la derivada del momento para obtener el cortante, no profundizaremos ahora en la explicación de dicho parámetro pues se dará su explicación más adelante.

Los valores definidos con el prefijo x_ son aquellos que dan una idea de dónde se sitúa cada carga con respecto del punto en el que se calculará el momento. Son datos muy útiles a lo largo de todo el proceso de programación, junto con los parámetros de top y bottom que ayudan a definir la estructura gráfica.

PROPIEDADES MECÁNICAS

Como propiedades mecánicas de la sección necesarias se añaden al script: el Área, la Inercia y el Momento Estático de la sección.

REACCIONES EN LOS APOYOS

Cálculo de las reacciones dadas en los apoyos, útiles para hallar los momentos y cortantes.

MOMENTO DE LA SECCIÓN

Se define entonces el momento que hay en la viga con la mayor cantidad de datos posibles, se ordenan de manera que todo quede mucho más claro y se escribe en el programa por tramos.

CORTANTE DE LA SECCIÓN

En la siguiente línea se comenta el propio cortante obtenido como derivada del momento, pero este no es totalmente exacto. Para obtener la derivada lo que se programa es una aproximación ínfima hasta llegar al número más concreto posible de dicha derivada (derivada obtenida por aproximación (métodos numéricos), hasta conseguir un error mínimo).

La ventaja principal proporciona una función continua, ya que los saltos, es decir, líneas verticales de la gráfica, en realidad el programa los considera como líneas casi verticales con pendiente casi infinita. La continuidad de la función permite que se ejecuten sin problema todas las operaciones posteriores que requieren del cálculo de derivadas de orden superior como por ejemplo podría ser la segunda derivada del momento para calcular el momento máximo, etc.

Hay ocasiones en las que a través de esta manera no se respetan las equidistancias de representación gráficas, eso viene dado precisamente por el valor de error dx que aproxima la derivada hasta que se afine todo lo posible a lo que sería su representación explícita; si este valor aumentase saldrían en el gráfico de nuevo otras formas representadas con otro tipo de errores menores.

Este valor tendrá una cierta influencia en la suciedad de las imágenes que se plotean. En el caso de una viga biapoyada el cual resulta como una imagen totalmente limpia, venía dado por el hecho de que estos valores eran normales y no se forzaban.

Aceptamos estos mínimos resquicios de suciedad en el dibujo para poder trabajar de manera más efectiva con las derivadas del momento a la hora de hallar el cortante. Se aborda el problema a través de un mínimo error, pues si lo hacemos a través del valor explícito de la derivada (el cual debería de ser el valor a utilizar, pues sería el exacto), el programa nos devuelve un ploteado extraño. Al no saber el porqué de este resultado, es por lo que decidimos quedarnos con unos valores de dibujo que nos resulten con un poco de ruido, pero igualmente explícitos para el fin al que queremos llegar.

SIGMA Y TAU

TENSIONES DE COMPRESIÓN Y TRACCIÓN

ÁNGULOS DE COMPRESIÓN Y TRACCIÓN

Se añaden Sigma como tensión normal y Tau como la tensión tangencial, así como las tensiones de compresión y tracción principales. Alfa2 se ha definido de manera similar a Alfa1 explícitamente y no como la suma de Alfa 1 más noventa grados, como en un principio se planteó, es por eso que se detalla explícitamente.

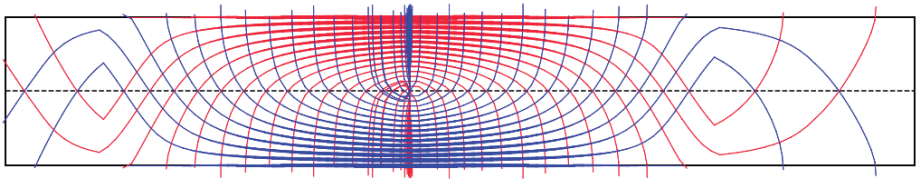
EL MOMENTO INVERSO

Para saber según el momento máximo cual es la tensión máxima (en valor absoluto) en cada punto, y así poder plotear con un espesor gradual las líneas isostáticas y que no se superpongan entre ellas, lo que se hace es el cálculo del Momento Inverso en valor absoluto. Una vez obtenido este, lo minimizamos y es así de donde se obtiene que el valor de x máxima estará en una tensión que en cada caso se encontrará a una distancia diferente de ese eje x.

PUNTO DE CORTE DEL MOMENTO CON EL EJE DE ABSCISAS EN LA LUZ INTERIOR DE LA VIGA PARA LA DEFINICIÓN DE LAS SECCIONES DE PLOTEADO

Previamente se habían ploteado los ejemplos de una manera diferente; en vez de plotear en tres zonas distintas, lo que se hacía era plotear en una sola parte de la sección de la crujía y obteníamos así unas representaciones de líneas isostáticas no del todo completas pues en su interior quedaban muchos huecos que no ofrecían información alguna.

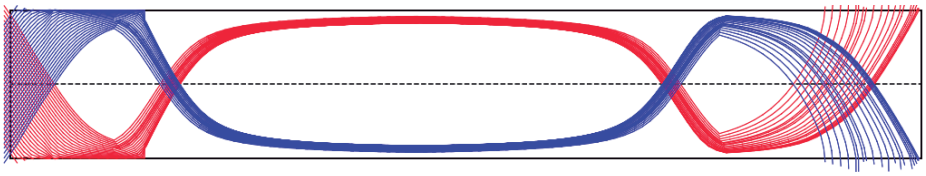
Cuando se ploteaba con una sola sección a lo largo de toda la viga (sección central), todo lo que rodeaba al interior en la mitad de la viga era bastante claro, pero a medida que la representación se acercaba a los extremos el dibujo empezaba a carecer de calidad y representación; la información no era clara y perdíamos detalle



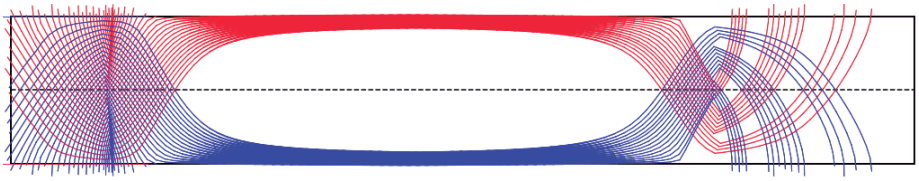
Sección de ploteado central

Se pensó entonces en plotear las isostáticas desde una sección diferente, pero no se sabía dónde colocar la sección a través de la cual obtener todas esas líneas de una manera más precisa.

Se testeó el modelo de diferentes maneras; ploteando desde el extremo izquierdo, quedaba muy denso y con mucha información por esa zona de la viga, pero a medida que se alejaba, volvían a aparecer esas faltas enormes en blanco en la representación gráfica; Si se llevaba a cabo la misma acción desde una zona un poco más separada del extremo y antes de llegar al apoyo, se generaba un gráfico un poco más definido quizás, pero igualmente escaso de información; Justamente ploteada en el apoyo, la información en el extremo izquierdo era muy abundante, pero seguía existiendo una falta de parámetros gráficos en el resto de la viga.

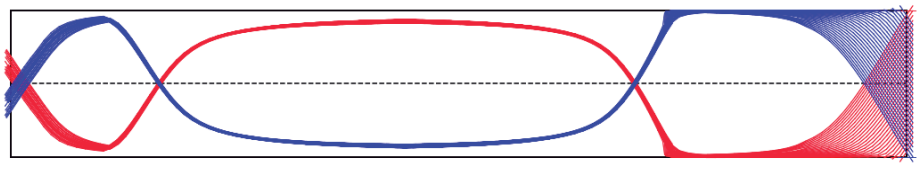


Sección de ploteado extremo izquierdo



Sección de ploteado entre extremo izquierdo y apoyo

De manera inversa se probaron también la representación de las mismas secciones por la parte derecha obteniendo los siguientes gráficos.



Sección de ploteado extremo derecho

Sección de ploteado entre extremo derecho y apoyo

Del último ejemplo surgió la idea final de representación, "cortar" las representaciones por tramos para que se plotearan aquellos gráficos los cuales ofrezcan una información detallada por zonas.

Es por ello que se piensa en la ley de momentos para la representación por tramos de estas líneas. El punto de corte del momento con el eje no es que coincida exactamente con el punto de corte de las líneas isostáticas con el eje neutro, para nada es así, pero se pensó que más o menos podría ser un buen punto de partida para obtener una representación gráfica por tramos interesante.

Se lleva a cabo una definición por zonas debido al corte del momento con el eje, es decir, con valor cero de momento. Hay que tener en cuenta que, en algunos casos, ese corte del momento con el eje no existiría, pero entonces la representación se llevaría a cabo a partir de los apoyos que serían entonces los puntos de referencia claves para la representación optima.

El corte del momento con el eje se obtiene a través de la ley de momentos que parte del tramo x_B y del tramo x_C, se igualan a cero y se resuelve la ecuación de segundo grado obteniendo lo que vale la x en cada parte del tramo. O sea, obtenemos cuando vale el momento cero cerca del apoyo B o cerca del apoyo conocido como C.

Hay varios casos en los que ese corte no existe, por ejemplo, cuando dentro de la raíz obtenemos un número negativo; si no existe el valor a de la ecuación de segundo grado porque no hay carga repartida en la mitad de la viga, entonces se tiene que dividir entre cero. Si alguna de esas opciones se llega a dar, lo que hacemos es decirle al programa que se quede en el apoyo a calcular las líneas de tensiones de la sección.

Toda la parte del script que comprende de la línea 204 a la 261, sirve para definir lo que acabamos de comentar, las secciones a partir de las cuales se representan las líneas isostáticas en la viga por tramos.

Se deja fuera de juego en comentarios la manera de llegar al mismo punto de corte minimizando una función, es decir, obteniendo donde llega a cero. Esta manera no era la mejor para abordar esta cuestión, pues cualquier función en la que se minimice lo que está haciendo es coger los mínimos más cercanos, conduciendo a error al programa; por ello se comenta y se decide no seguir por este camino, si no escoger lo que hemos comentado anteriormente.

REPRESENTACIÓN GRÁFICA O ÚTILES DE PLOTEADO

fig_ratio: es el factor de escala que hace que los ejes estén perfectamente escalados, este valor se verá modificado por el comentado anteriormente distorion_Y, y de manera muy leve. Pero es necesaria su definición en el programa para que en la práctica sea sencillo de ver qué ocurre en el interior de la viga.

horizotal_length: es el valor de longitud de pixeles que hace que la definición de la figura sea mayor o menor. Va multiplicado por otro factor que es el scale_pix, que por defecto lo escribimos con un valor de diez obteniendo así una muy grata resolución en la obtención de las líneas isostáticas.

Lo que se define la longitud en base a esa proporción de pixeles. Tanto la longitud como los ejes contienen estos parámetros interiorizados para que todo quede dentro de una escala exacta. Los ejes x e y están divididos por la ratio de la figura en todo momento. Más abajo en esas mismas líneas se definen los parámetros que plotearan en la gráfica los ejes de coordenadas.

CONSTRUCCIÓN DE LÍNEAS ISOSTÁTICAS

La manera en la que se plotean las líneas se divide indiferentemente del tramo al que se refiera en dos partes: parte de compresión (representación en rojo) y parte de tracción (representación en azul).

Los datos preliminares indican el punto de la sección a partir de la cual se van a empezar a dibujar las líneas isostáticas; esto proviene como se ha explicado previamente, del momento con valor cero y corte en el eje de abscisas, o en su defecto, si este valor no pudiera ser hallado, por el apoyo más cercano a cada tramo.

Se insertan en el script primero el cálculo y ploteado de las líneas de compresión y después por ende las de tracción. La diferencia principal diferencia entre una representación y otra es que en las isostáticas de compresión se inserta el valor de Alfa1 y en las de tracción el de Alfa 2, pero el proceso es prácticamente igual en ambos casos para la escritura de la programación.

Se da uso a una función recursiva, y dentro de se reitera de nuevo otra nueva función recursiva, pero cada una significa lo siguiente:

Antes de nada, lo que se hace es poner un contador el cual tiene la función crucial de plotear una curva después de otra hasta que el valor de compresión o tracción resulten igual a cero en el interior de la viga, es entonces cuando ese contador se agota y se termina el cálculo de líneas, pasando a otro tramo hasta que se finalice el proceso y se plotee totalmente lo que buscamos.

Este movimiento quedaría dentro del primer comentario "while", que permite ir realizando el trabajo que acabamos de comentar. Pero dentro de este proceso de hallar cada línea lo que se escribe en el programa son los saltos entre puntos para que se puedan dibujar esas propias líneas.

Esta operación se lleva a cabo hasta que se agote bien porque se acabe la sección y no haya más puntos o bien porque la línea isostática no tenga valor, entonces cambia de curva y el contador automáticamente salta a la siguiente línea para calcularla. Este procedimiento es reiterativo para así hallar toda la representación de las isostáticas tensionales internas de la viga.

El número de líneas que contendrá el contador vendrá definido por el parámetro de n_lines, que no es más que el canto de la sección dividido entre el valor equidist_left/mid/right (dependiendo del tramo en que se encuentre el cálculo). Si el contador es igual a cero, este se posiciona en el punto alto de la sección en la que se encuentre

(o más bajo si son de tracción las líneas), entonces se le ordena que mientras que el contador sea menor o igual que dicho número calcule líneas isostáticas de manera automática. Aquí se verifica que siempre obtendremos resultado en el interior de la viga, sin dar lugar a fallos.

Obligamos al programa a que cumpla la orden de que cuando acabe de calcularse una línea, el contador empiece a calcular en la siguiente línea isostática. El conjunto de puntos definido por punto points_x es la primera coordenada de cada punto y el valor de points_y será la segunda coordenada. Una vez situado el contador para halla una nueva curva, entonces se construye la misma hacia ambos lados (derecha e izquierda).

Primero se construye una curva del punto del contador inicial hacia la izquierda y luego desde ese mismo punto hacia la derecha. Esto aparece todo dentro de los comandos de los dos comentarios "while".

Lo único que cambia dentro del cálculo de isostáticas a derecha o izquierda de la sección es que en una se le fuerza a que vaya a la izquierda multiplicando la función con un menos y a las de la derecha el símbolo será siempre positivo.

Pero primero ha de comprobarse que el punto donde se sitúa el contador está inscrito dentro del rectángulo para que cuando salga del rectángulo pare y se vuelva a otro punto diferente. Para ello se añaden varias condiciones separadas por la palabra "and" para obtener esto hemos de decirle al programa que la coordenada en el eje y debe de ser mayor que la fibra inferior y también que sea menor o igual que la fibra superior; y que la coordenada en el eje x ha de ser mayor o igual que la parte izquierda de la viga y que esta misma sea menor o igual que la parte derecha de la viga. Habría que añadir una última condición para que el punto del contador salte si uno de los dos parámetros sigma o tau es igual a cero. Si todas estas condiciones geométricas suceden, obligando a que el punto quede dentro de la viga, entonces se procederá al cálculo y ploteado de la línea tensional que esperamos.

Calcula el ángulo, el vector de movimiento de un punto a otro como la distancia que le hallamos dado desde un principio y que es un valor que por definición mantenemos como una división del canto entre trescientos.

La tupla viene definida por estos vectores que se hallan anteriormente, este es el comando que me sirve para plotear las líneas que vienen definidas por los vectores.

V se define como un vector: menos el coseno de alfa (pues se define la parte izquierda, cuando sea derecha este valor será positivo) más coseno de alfa; esta es una interacción de módulos, así hallamos las líneas que definen las isostáticas que queremos hallar.

El valor a de origen es una tupla, que es como si dijéramos un punto en el espacio y el valor dado como v si es un vector. Para sumar vectores, debo convertir el valor a en un vector, entonces se suman ambos obteniendo un nuevo vector, pero para plotear este nuevo vector que se obtiene entonces el programa da de deshacer el cambio que se ha realizado previamente del parámetro a.

Una vez que ya tengo el nuevo punto a entonces le digo que a lo que se define anteriormente como el punto de partida se le añadan las coordenadas de esto último que es lo que programa define como "append". Ahora se tiene una entidad que ha almacenado ambas coordenadas nuevas para luego plotearlas, que es cada uno de los trozos de la curva que esperamos obtener finalmente. Este proceso se plotea y una vez ploteado, entonces

empieza el proceso de nuevo, pues ya se ha acabado la indentación de esta parte del programa.

Para definir una curva de compresión y otra de tracción se le añade color a la línea de ploteado, color rojo para las de compresiones y azul para las tracciones, diferenciando así de una manera mucho más grafica nuestro objetivo.

Este proceso se repite igual para la línea de tracción, pero como bien hemos anotado antes, el seno y coseno son positivos, y cuando cambia de tramo las condiciones iniciales cambian según la sección escogida para empezar a calcular las líneas.

El algoritmo que teníamos para hallar las líneas de compresión a partir de cada sección hacia la izquierda y después hacia la derecha, y de igual modo las líneas de tracción hacia la izquierda y derecha, se escribe ahora tres veces según tramos de la sección a partir de los cuales se tienen que plotear las líneas isostáticas (left_part, middle_part y right_part).

PLOTEADO GEOMETRÍA DE CARGAS, SECCIONES, ANOTACIONES

En la última parte del script se encuentran escritos aquellos pequeños comandos que hacen que la representación del ploteado final haga que el trabajo encuentre un carácter mucho más aceptable en su presentación.

Lo primero que se observa dentro de esta parte del script es la representación a escala gráfica de las líneas que simularían las fuerzas o cargas que actúan sobre la viga a calcular. De una manera reiterada y repitiendo el proceso para cada carga, al programa se le ordena que dibuje un polígono abierto de tres lados para las cargas repartidas, invertido y que sus extremos acaben con una flecha la cual indica la dirección de esta carga corrida. De una misma manera, pero solamente con líneas verticales y a escala según el valor de cada carga, se representan aquellas cargas puntuales.

Cada una de estas representaciones tiene una condición de no ser representada cuando el valor inicial impuesto para cada carga sea igual a cero, así no habrá problemas si en algún caso no existiera carga en alguno de los tres vanos posibles a calcular dentro de todos los casos de vigas que podríamos encontrar.

En el final de cada uno de estos comentarios se emplea una línea para que el mismo valor de cada carga aparezca representado en la parte superior de cada polígono o línea.

Lo siguiente que se lleva a cabo son las líneas que miden el ancho de cada vano, de igual o parecida manera que hemos dibujado y nombrado las cargas, ahora se deja todo en base a los anchos de los vanos, obteniendo así una representación leal de cuanto valor tendría cada uno de ellos en la realidad, y de manera latente, tener presente la escala gráfica de la viga.

Finalmente, se plotean las fuerzas internas de la viga, es decir las leyes de momento y de cortante presentes en cada uno de los modelos de cálculo. Estos gráficos son modificables en su escala en los parámetros de modificación de escala del inicio del script, pudiendo así manejar una representación que se adapte a la pantalla del ploteado

COMENTARIO DE LOS DIFERENTES RESULTADOS OBTENIDOS

01 VIGA BIAPOYADA

Con carga repartida
Con carga puntual

02 VIGA CON VOLADIZO A LA DERECHA

Con carga repartida
Con carga puntual
Con carga repartida más puntual

03 VIGA CON VOLADIZOS GRANDES

Con carga repartida
Con carga puntual
Con carga repartida más puntual

04 VIGA CON VOLADIZOS PEQUEÑOS

Con carga repartida
Con carga puntual
Con carga repartida más puntual

05 VIGA CON CARGA SOLO EN VOLADIZOS

Con carga repartida
Con carga puntual

06 VIGA EN VOLADIZO

Con carga repartida
Con carga puntual

07 VIGA CON DATOS DIFERENTES

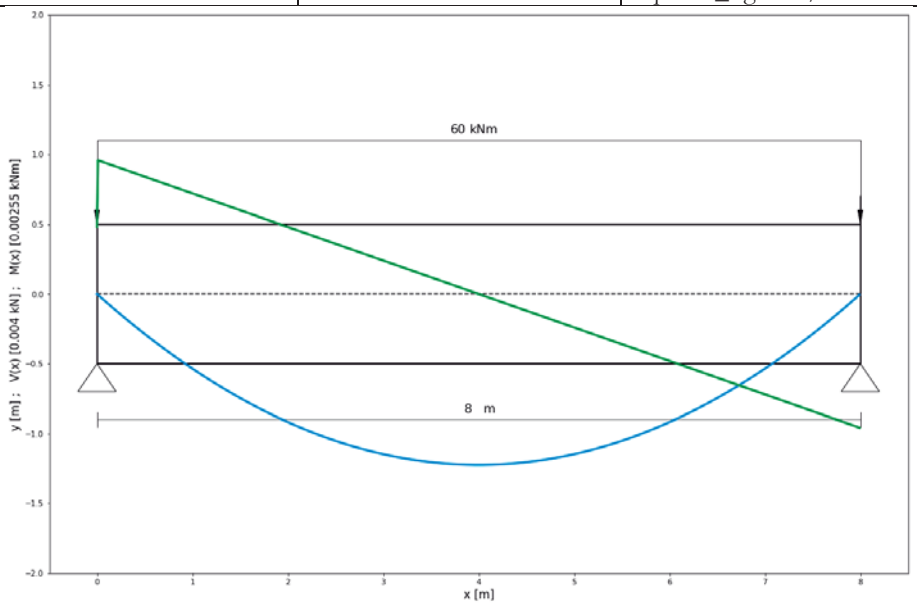
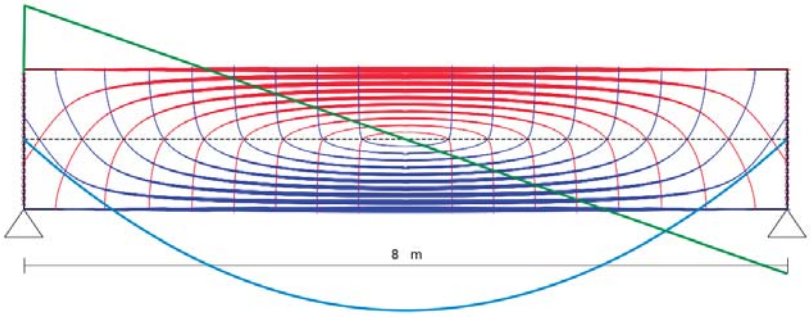
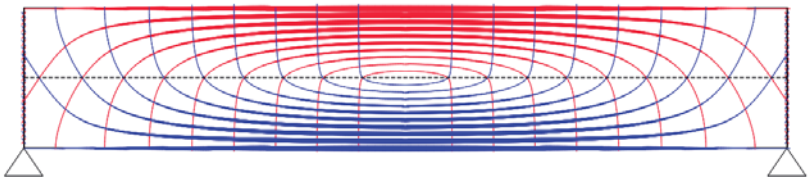
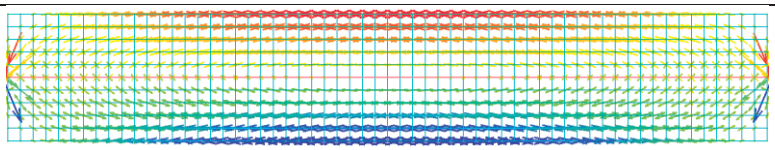
Con carga repartida
Con carga puntual
Con carga repartida más puntual

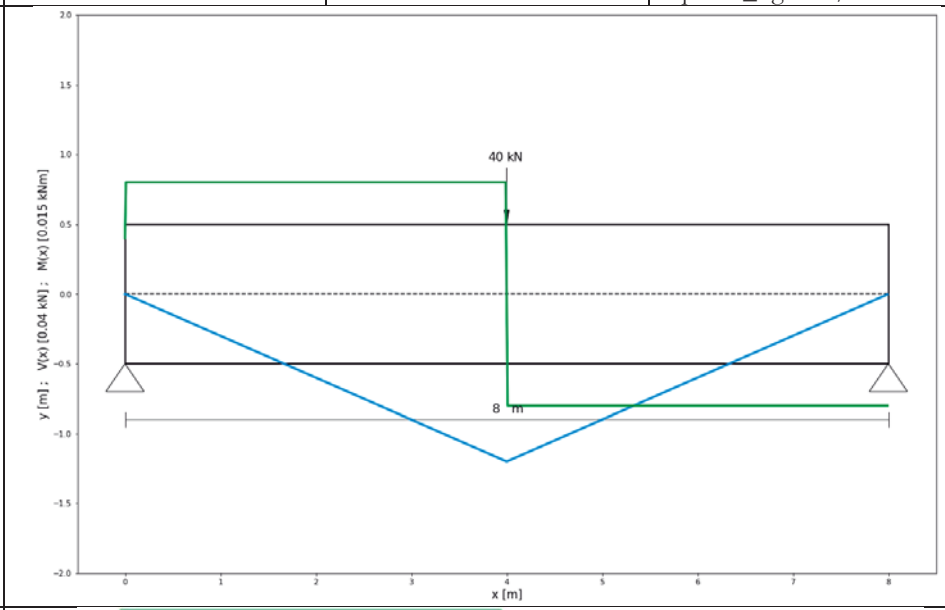
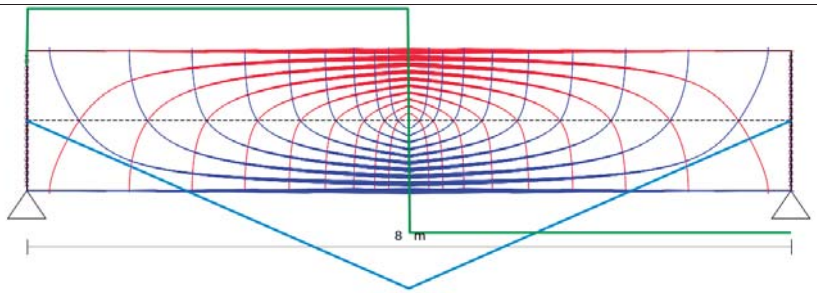
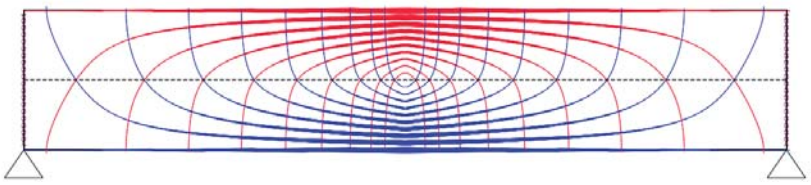
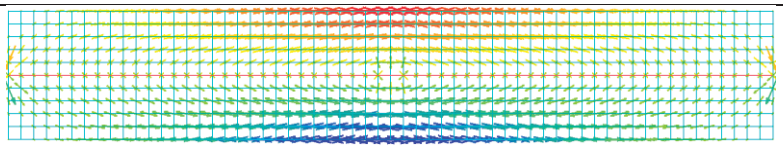
De manera preliminar ha de explicarse que habrá una diferencia que se observará en cada una de las fichas que a continuación se presentan entre los elementos finitos y las líneas isostáticas de la viga:

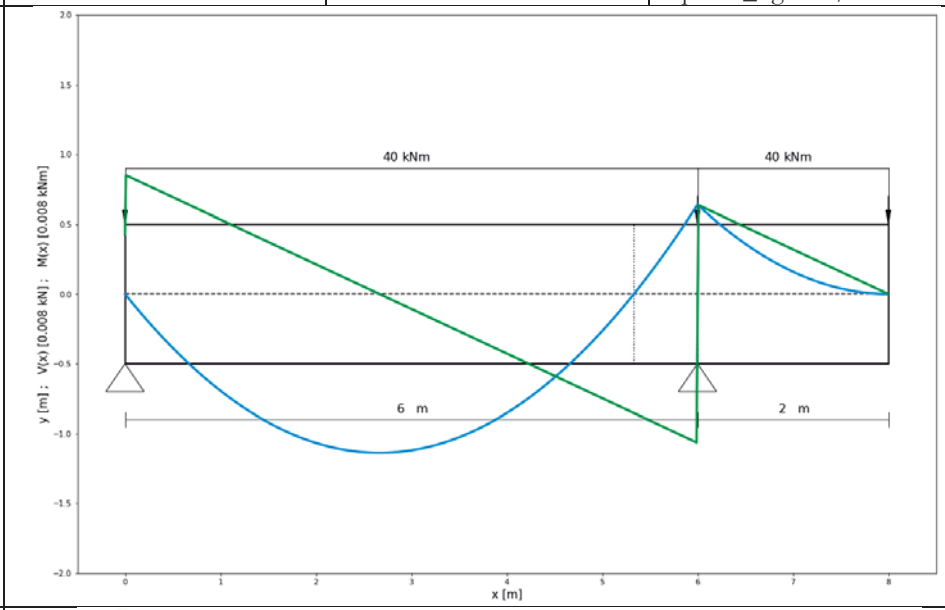
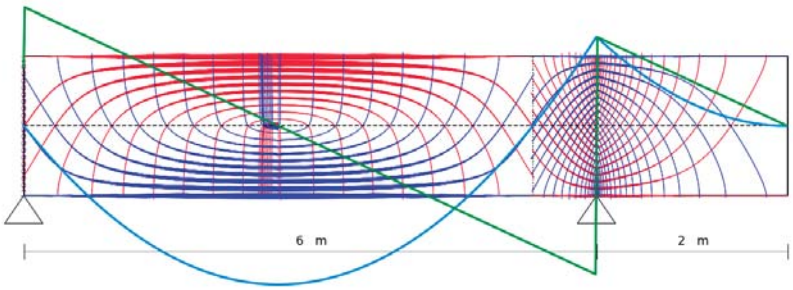
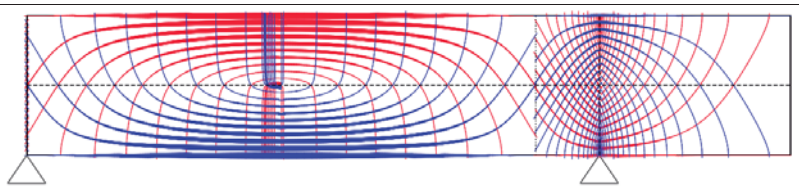
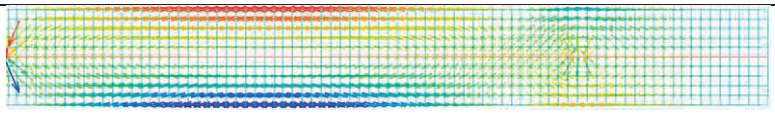
En los elementos finitos (EF) lo que se está modelizando es una estructura bidimensional mientras que, en el otro caso de líneas isostáticas, lo que se dimensiona es un ejemplo unidimensional en la cual se considera que la sección tiene altura real (son dos cosas totalmente distintas).

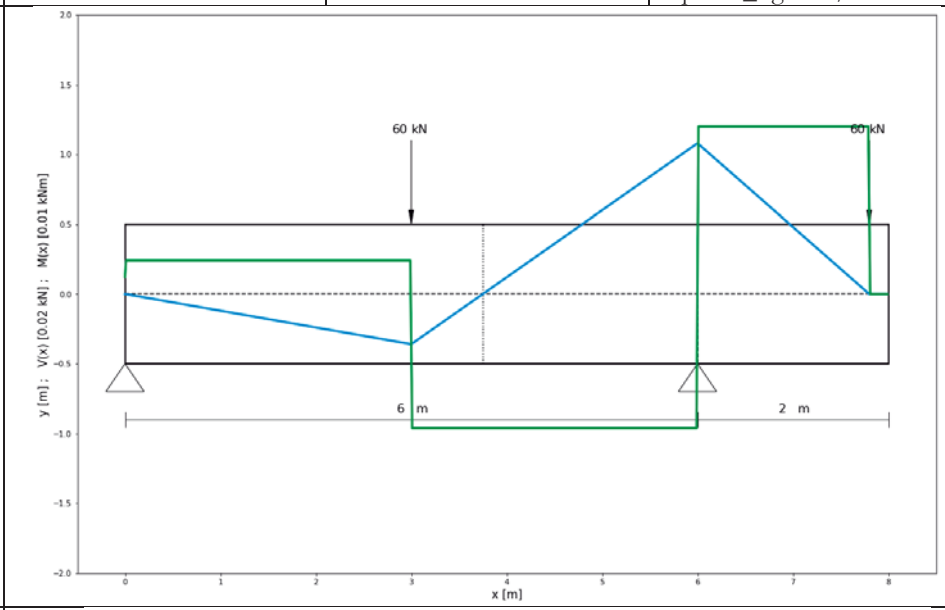
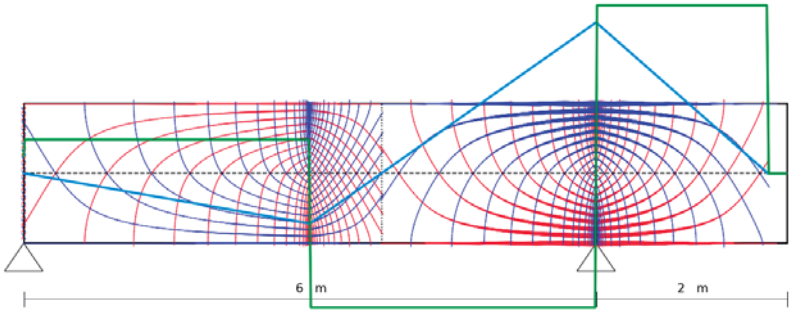
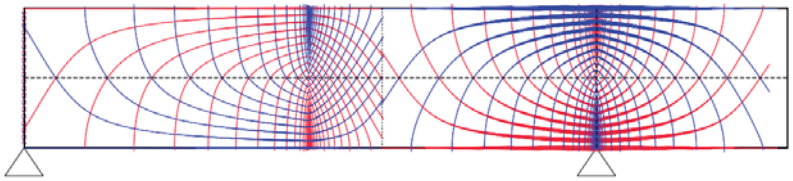
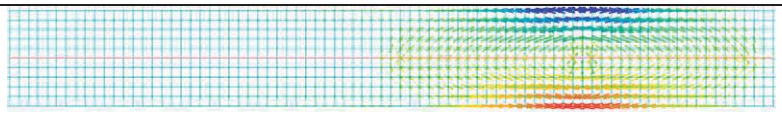
En un caso (el de las líneas isostáticas), es una sección extruida a lo largo de un eje mientras que en el caso de los elementos finitos es como si fuera una superficie extruida de manera perpendicular al papel.

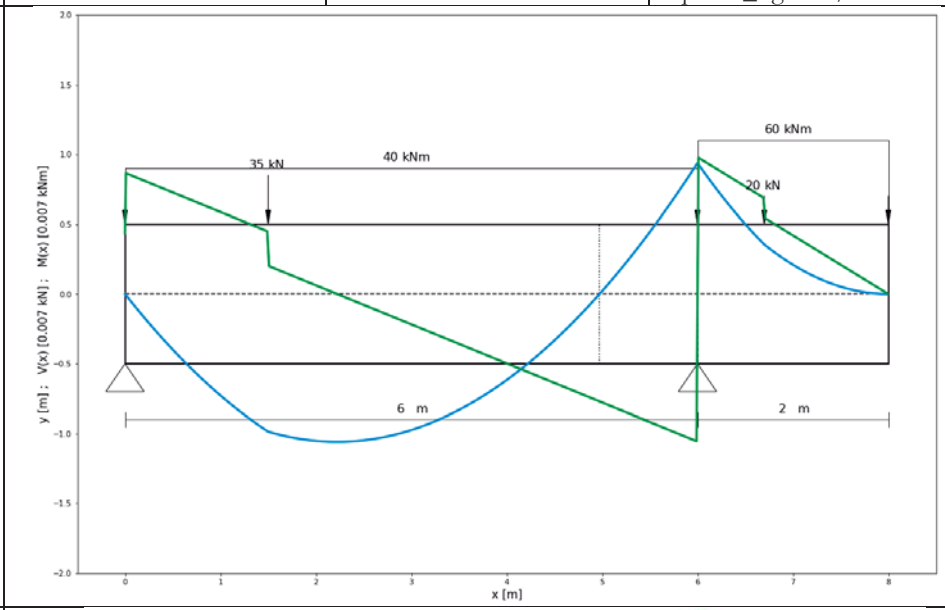
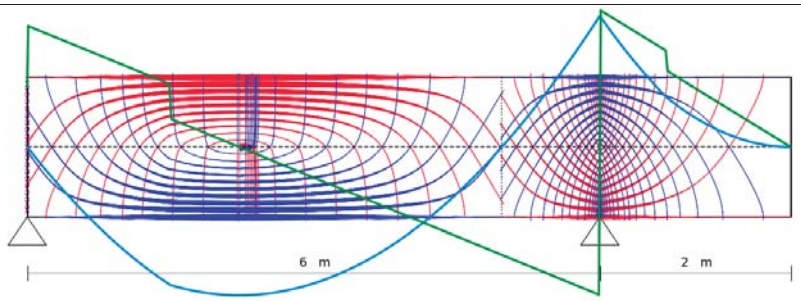
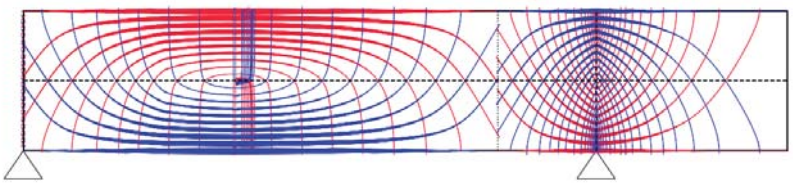
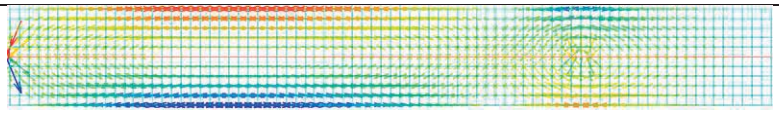
Todo lo que se acaba de comentar, se manifiesta en los apoyos supuestos de los elementos finitos, donde habrá una concentración de fuerza sustancialmente mayor, ya que confluyen todas en ese nudo. Sin embargo, en el otro caso, como la viga tiene un canto con altura considerable es como si el cortante o fuerza actuante no se concentrase en un punto solo, si no repartida en toda la sección y no solo en un punto.

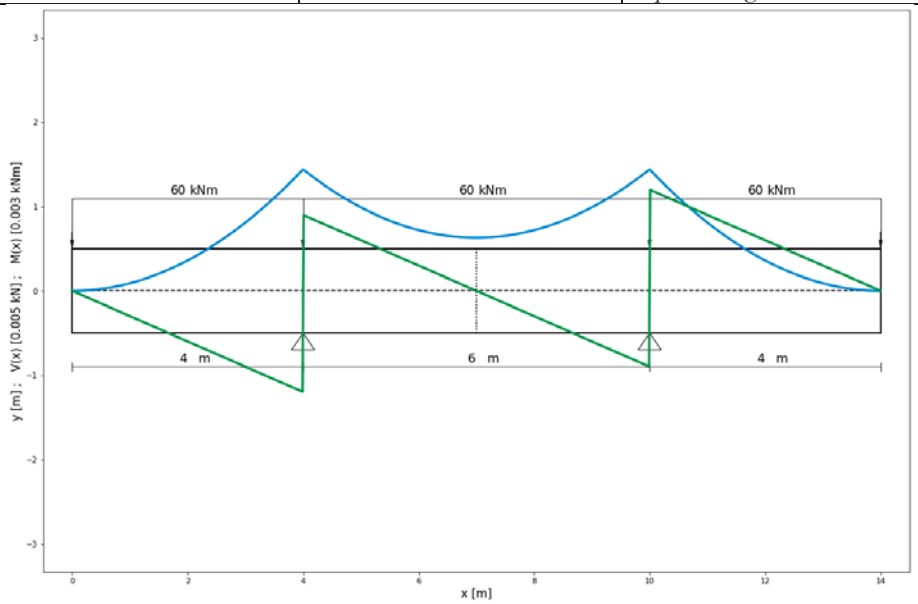
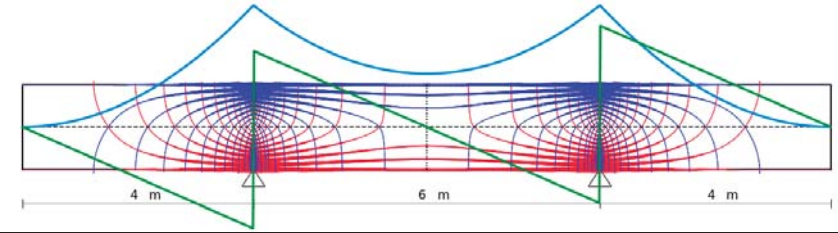
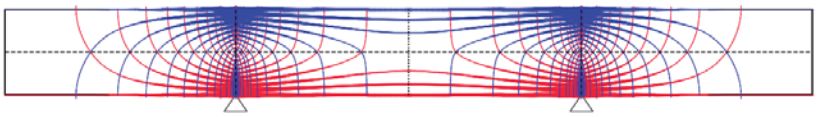
CASO	01	TIPO	Viga biapoyada	
Descripción	Viga de un vano con carga repartida uniforme			
Input	b=0.3 h=1 L1=0, L2=8, L3=0		q1=0, q2=60, q3=0 P1=0, P2=0, P3=0 d1=0, d2=0, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	A diferencia del siguiente ejemplo, en este caso las curvas de las líneas isostáticas aguantan más en el eje horizontal porque el momento es, a lo largo de todo el recorrido de la viga, con una constancia de valor positivo mayor. Normalmente estas curvas se dibujan más parabólicas (como puede verse en diversos ejemplos de Torroja), pero sus recorridos son casi siempre u horizontales o verticales. Debido a un factor de escala vertical para la obtención de una mayor resolución de la gráfica, el cruce entre curvas isostáticas que debiera ser en ángulo recto no se llega a producir.			

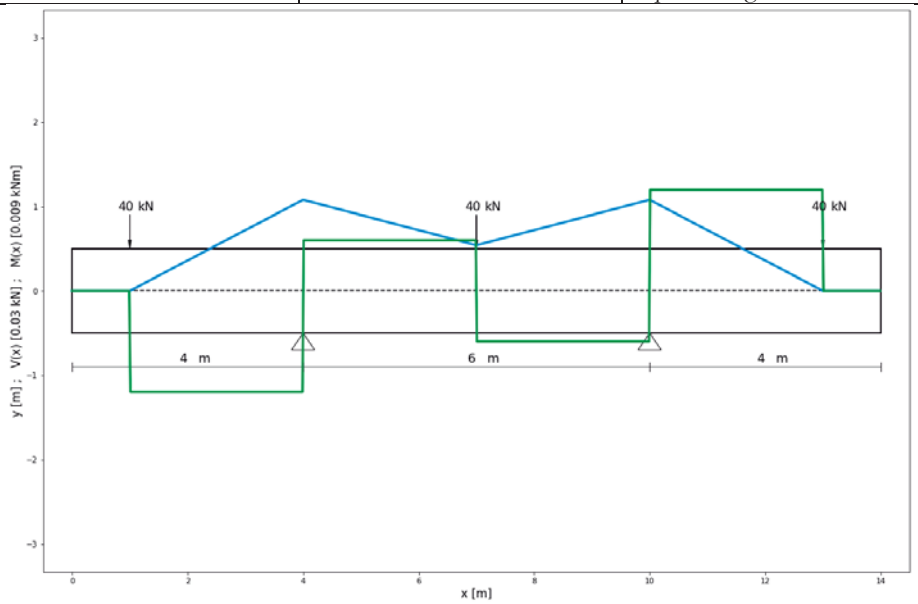
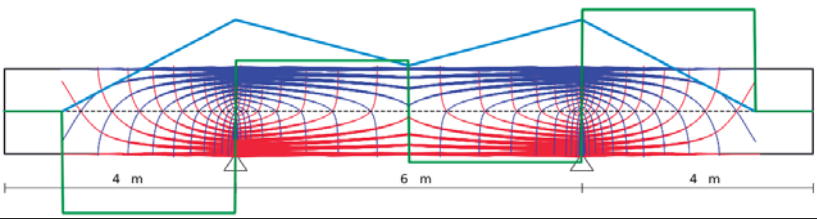
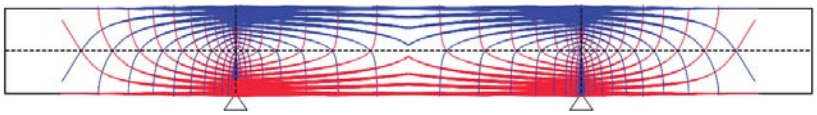
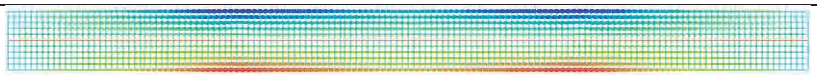
CASO	01	TIPO	Viga biapoyada	
Descripción	Viga de un vano con carga puntual centrada			
Input	b=0.3 h=1 L1=0, L2=8, L3=0		q1=0, q2=0, q3=0 P1=0, P2=40, P3=0 d1=0, d2=4, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	En este segundo caso las curvas de las líneas isostáticas aguantan menos en el eje horizontal y tienden a buscar rápidamente la verticalidad pues el momento es de valor positivo en una menor medida. Tanto en este caso, como en diversos casos sucesivos, cuando hay un salto grande en la ley de cortantes, y su valor cambia de positivo a negativo (o viceversa), en la representación de las líneas aparecen picos. Al ser un valor lineal, en el eje central el valor tensional de las líneas se hace casi cero, y su ángulo de partida será de unos cuarenta y cinco grados.			

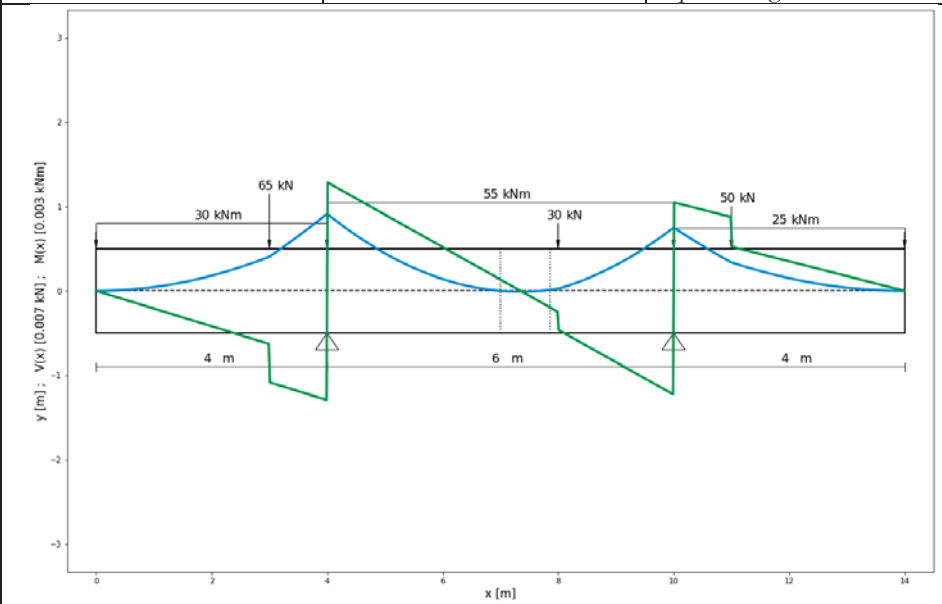
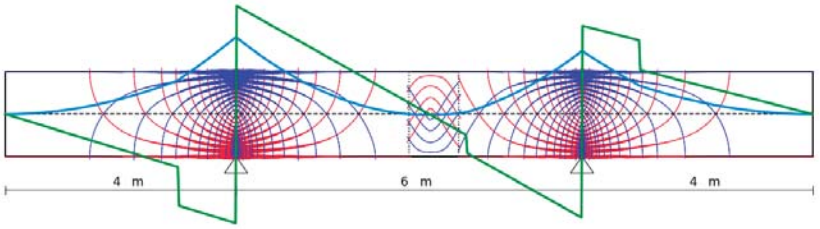
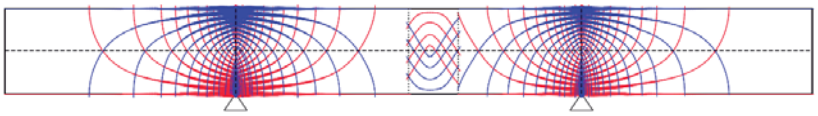
CASO	02	TIPO	Viga con voladizo a la derecha
Descripción	Viga con un voladizo con cargas repartidas uniformes		
Input	b=0.3 h=1 L1=0, L2=6, L3=2	q1=0, q2=40, q3=40 P1=0, P2=0, P3=0 d1=0, d2=4, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	Cabe destacar el punto en la sección de la viga en que se da la transición de las líneas isostáticas que se han representado. Debería ahora ver si es fruto de una casualidad (o no) el que la segunda curva representada de la parte izquierda de la viga (o vano principal) tanto a tracción como a compresión coincide justo en el punto en que la ley de momentos es igual a cero. Es importante observar también en muchos de estos casos como muchas de las líneas isostáticas representadas mueren en las caras superior e inferior de la viga, y cuáles de ellas continúan su recorrido intentando unirse en la representación con sus homónimas más cercanas. En el centro del vano hay una pequeña distorsión producida por el valor mínimo de error que introducíamos en el cálculo de la derivada del momento para hallar el cortante.		

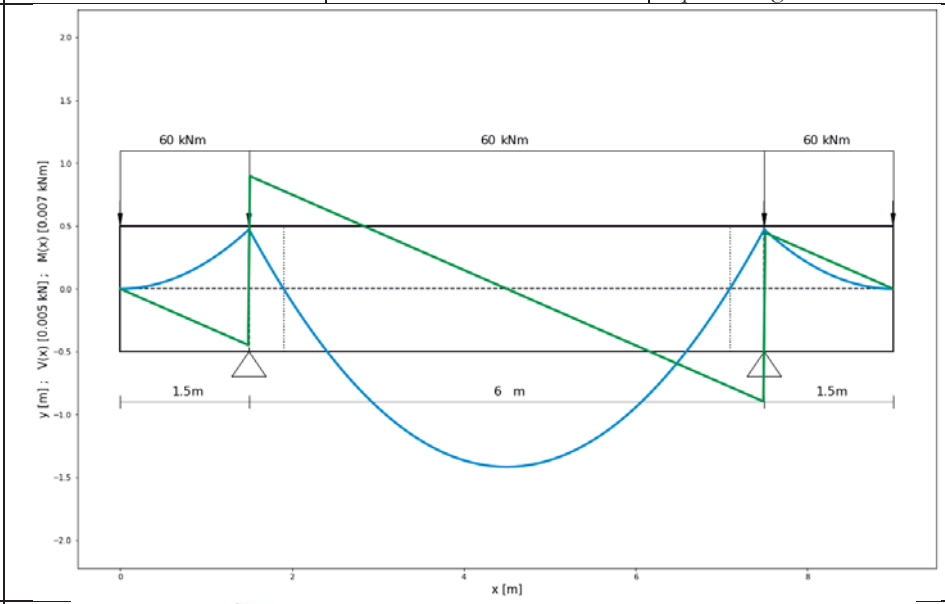
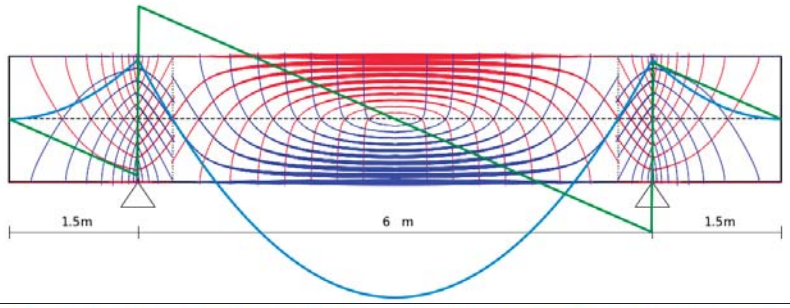
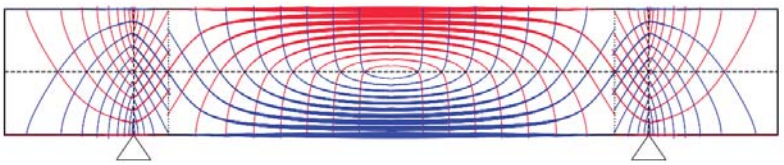
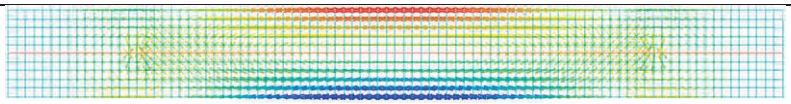
CASO	02	TIPO	Viga con voladizo a la derecha	
Descripción	Viga con un voladizo con cargas puntuales descentradas			
Input	b=0.3 h=1 L1=0, L2=6, L3=2		q1=0, q2=0, q3=0 P1=0, P2=60, P3=60 d1=0, d2=3, d3=1.8	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	En este caso en el que las pendientes de las leyes de esfuerzos son tan asimétricas se observa que el cambio de horizontal a vertical es mucho más pronunciado cuanto más rápido se produzca la asimetría anteriormente mencionada. En este caso podemos observar como de casualidad tanto una curva de tracción como de compresión se unen y forman una sola continua.			

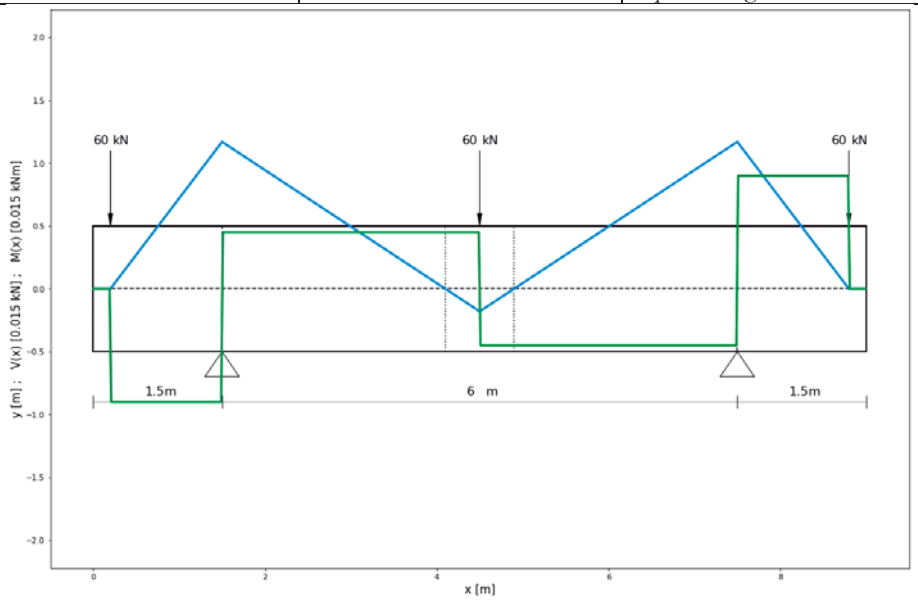
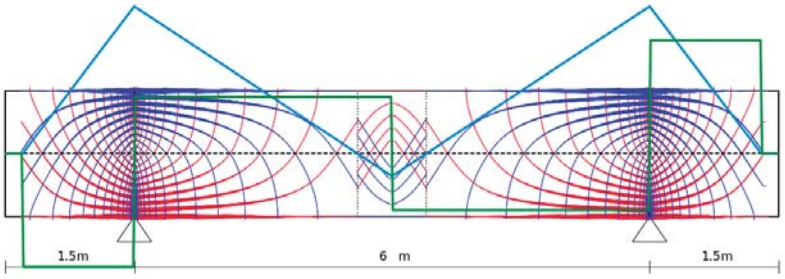
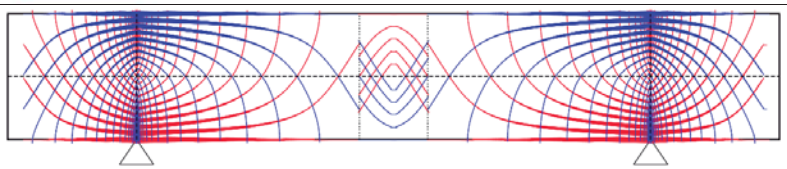
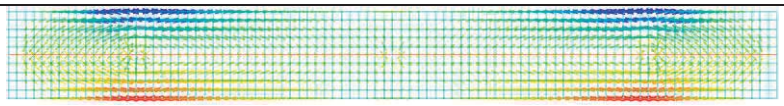
CASO	02	TIPO	Viga con voladizo a la derecha
Descripción	Viga con un voladizo con cargas repartidas y puntuales desiguales		
Input	b=0.3 h=1 L1=0, L2=6, L3=2	q1=0, q2=40, q3=60 P1=0, P2=35, P3=20 d1=0, d2=1.5, d3=0.8	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	Como está presente un salto tan grande de cortante ocurre lo mismo que en el caso anterior que ya se ha comentado, aparece un pico en la parte de la representación derecha de líneas isostáticas. Ambos casos (este y el anterior) son muy parecidos. Este tendrá mayor densidad de representación porque tiene mayor carga.		

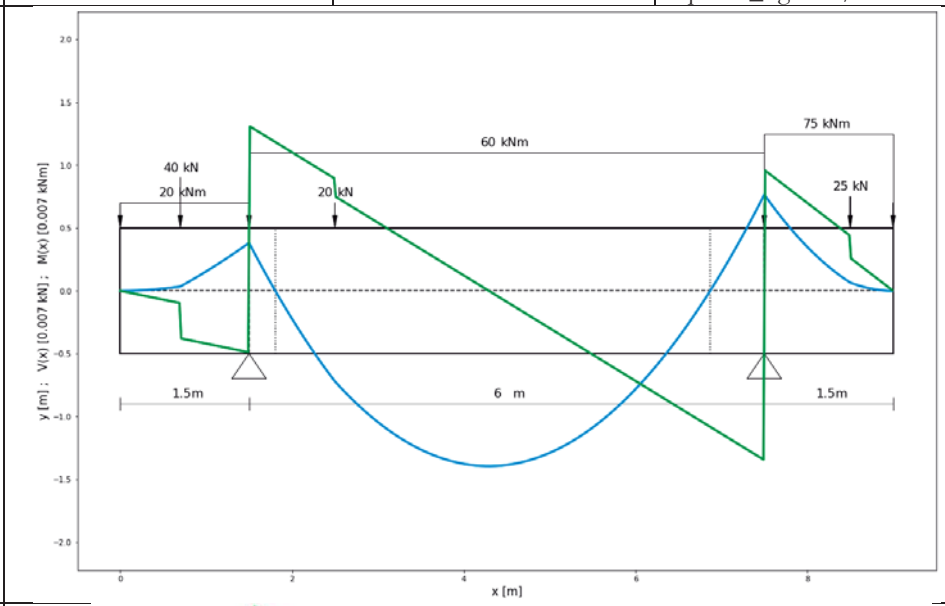
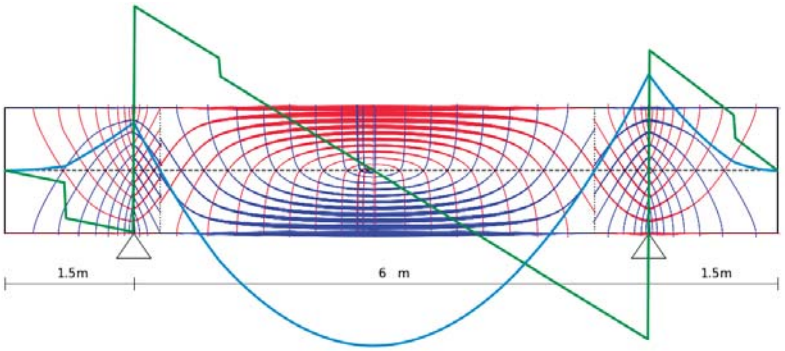
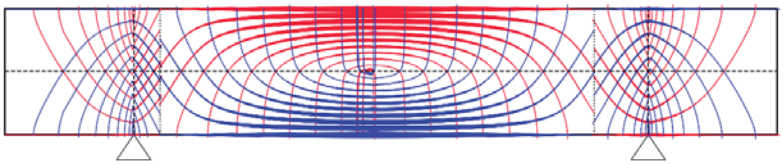
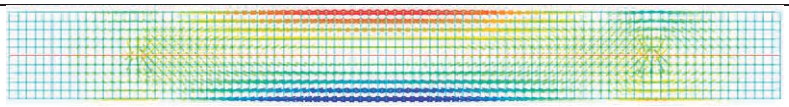
CASO	03	TIPO	Viga con dos voladizos grandes	
Descripción	Viga con dos voladizos con carga repartida uniforme			
Input	b=0.3 h=1 L1=4, L2=6, L3=4		q1=60, q2=60, q3=60 P1=0, P2=0, P3=0 d1=0, d2=0, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/1 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	Este es un caso totalmente simétrico en el cual es interesante ver como una parte de las líneas isostáticas continua su recorrido casi en horizontal, y el resto de representaciones tensionales termina muriendo en las caras superior e inferior de la viga. Puede dar la sensación que debido a su geometría pueda parecer a el caso de viga biapoyada, pero con los colores invertidos.			

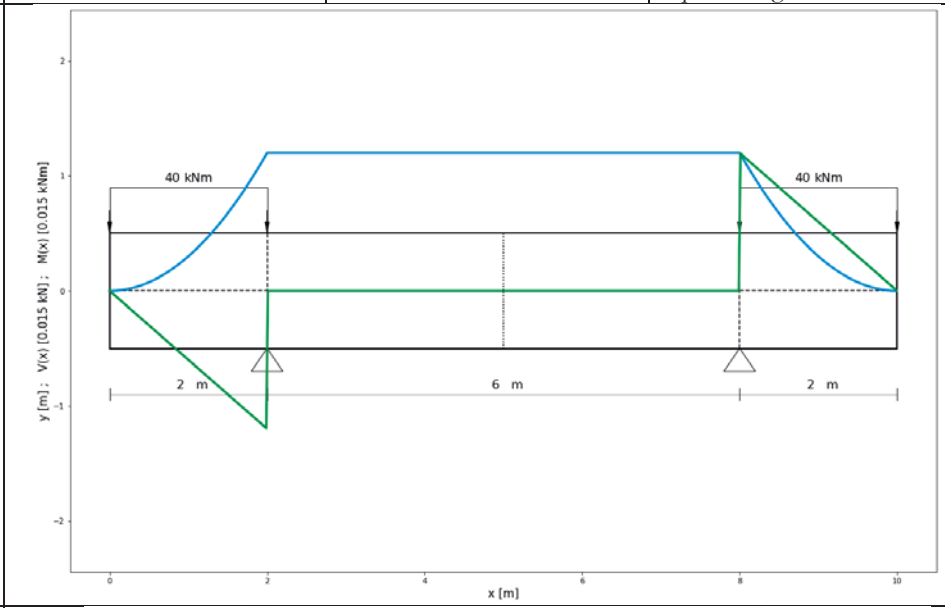
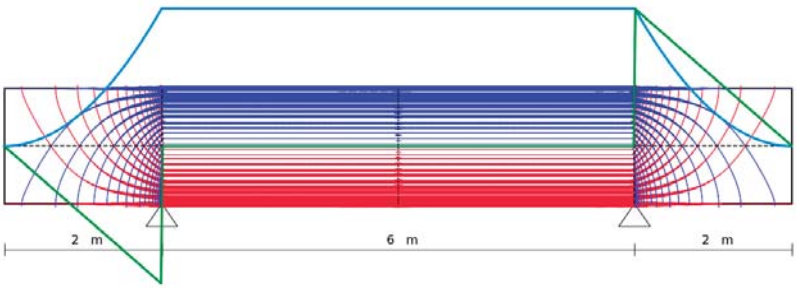
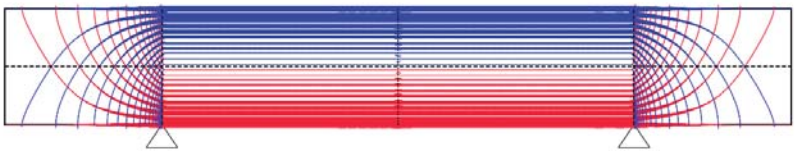
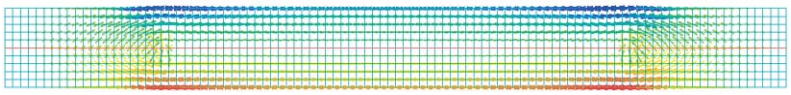
CASO	03	TIPO	Viga con dos voladizos grandes	
Descripción	Viga con dos voladizos con cargas puntuales			
Input	b=0.3 h=1 L1=4, L2=6, L3=4		q1=0, q2=0, q3=0 P1=40, P2=40, P3=40 d1=1, d2=3, d3=3	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	Este caso es exactamente igual que el anterior, pero con un pico en su representación, ya sabemos una vez más, debido al cambio den las leyes de esfuerzos. En sus extremos, al no ejercer ninguna carga se cortan las representaciones gráficas de curvas; esto se verá en más de un caso a lo largo de estas páginas.			

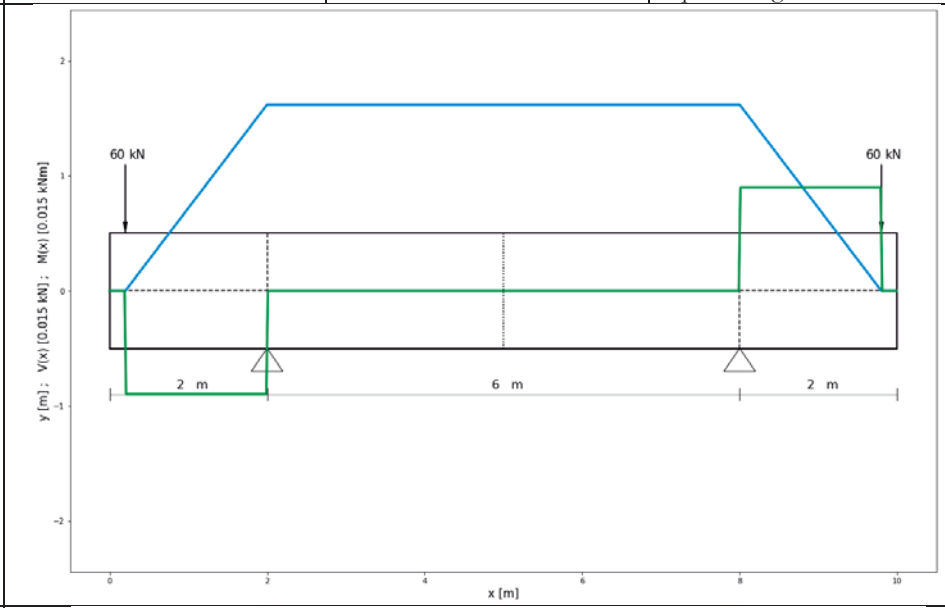
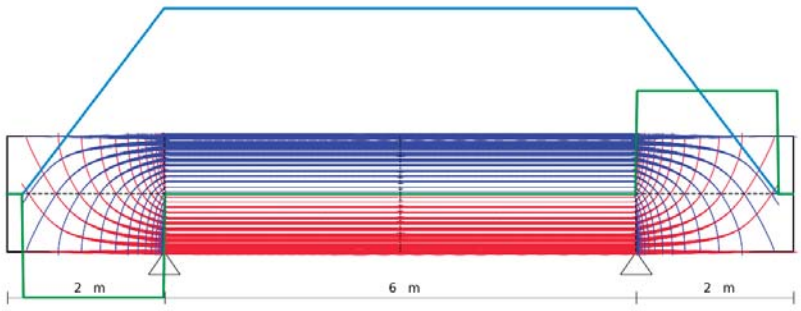
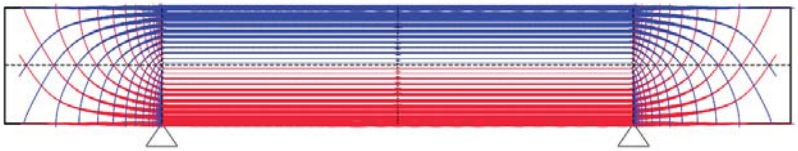
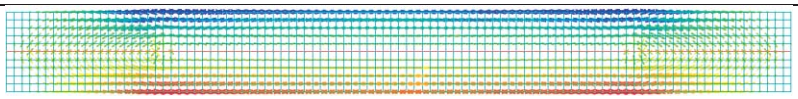
CASO	03	TIPO	Viga con dos voladizos grandes
Descripción	Viga con dos voladizos con cargas repartidas y puntuales desiguales		
Input	b=0.3 h=1 L1=4, L2=6, L3=4	q1=30, q2=55, q3=25 P1=65, P2=30, P3=50 d1=3, d2=4, d3=1	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/5 equidist_right=h/20
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	Se evidencia que hay una parte muy pequeña de momento que corta al eje y por tanto no se puede hacer una continuación entre vanos de líneas. Quedan partes blancas en el interior del boceto, pero no podemos hacer mucho más, pues si no ambos apoyos se saturarían demasiado cosa que para la representación no es bueno, pues deja de entenderse.		

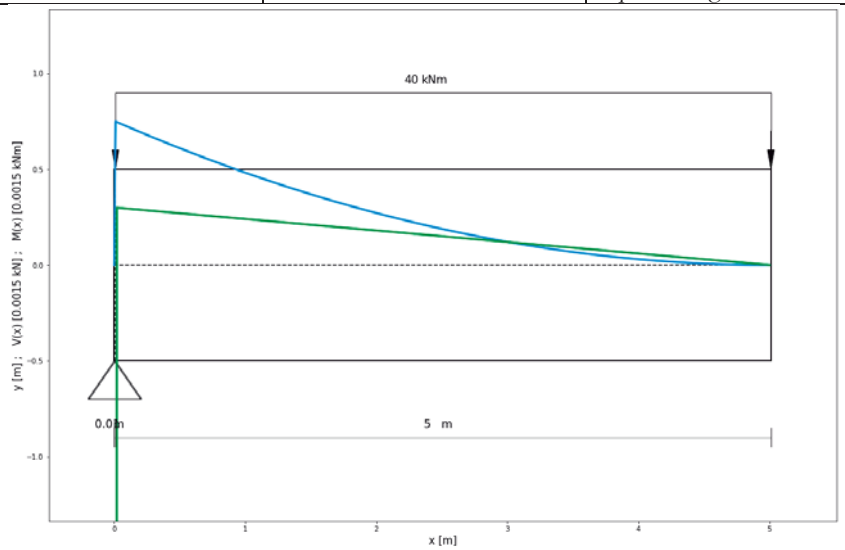
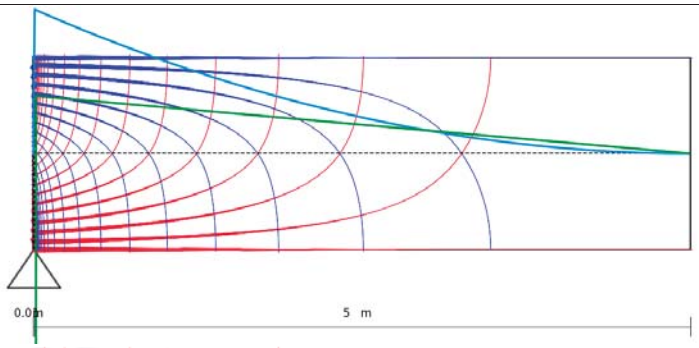
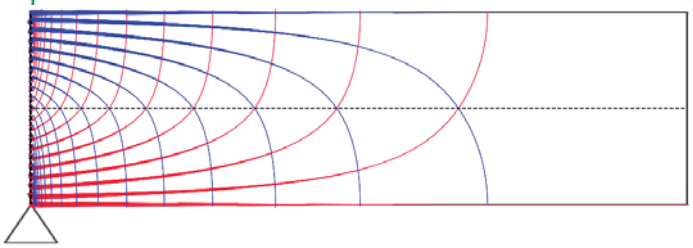
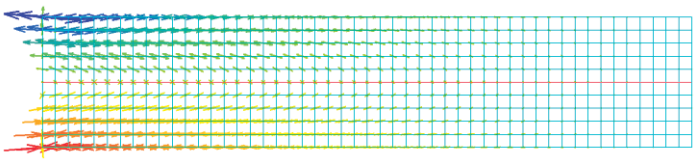
CASO	04	TIPO	Viga con dos voladizos pequeños	
Descripción	Viga con dos voladizos con carga repartida uniforme			
Input	b=0.3 h=1 L1=1.5, L2=6, L3=1.5		q1=60, q2=60, q3=60 P1=0, P2=0, P3=0 d1=0, d2=0, d3=0	distortion_Y=1.5 equidist_left=h/10 equidist_mid=h/20 equidist_right=h/10
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	En este caso se ve claramente en ambas secciones que se pueden llegar a enlazar correctamente, y que en los extremos las líneas son muy delgadas precisamente porque el momento negativo en esas zonas es mucho menor.			

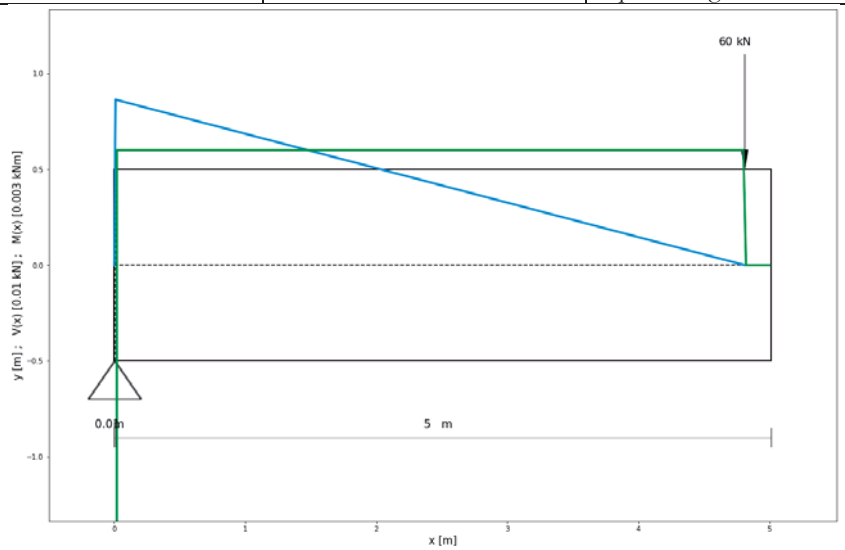
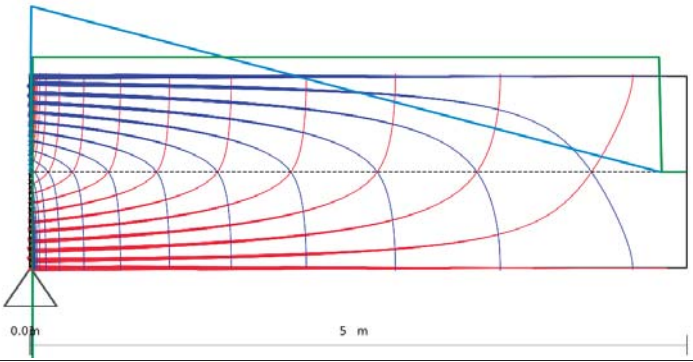
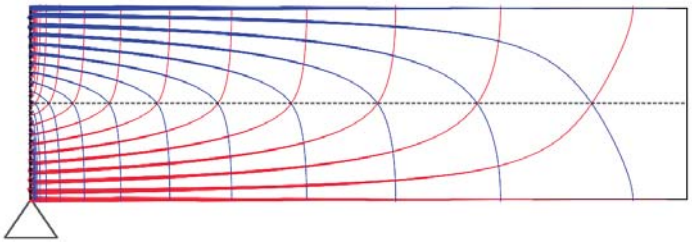
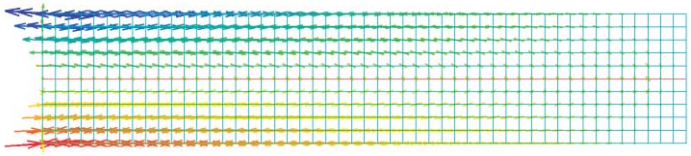
CASO	04	TIPO	Viga con dos voladizos pequeños
Descripción	Viga con dos voladizos con cargas puntuales		
Input	b=0.3 h=1 L1=1.5, L2=6, L3=1.5	q1=0, q2=0, q3=0 P1=60, P2=60, P3=60 d1=0.2, d2=3, d3=1.3	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/10 equidist_right=h/20
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	En este se ve un caso nuevamente de picos, y como bien se mencionaba anteriormente, en los extremos hay un blanco, en el cual se deja de representar pues no hay tensiones actuantes ya que no hay cargas (las leyes de esfuerzos son iguales a cero). Zona central muy pobre en representación, pero casi se llega al enlace entre extremos y parte media de la viga.		

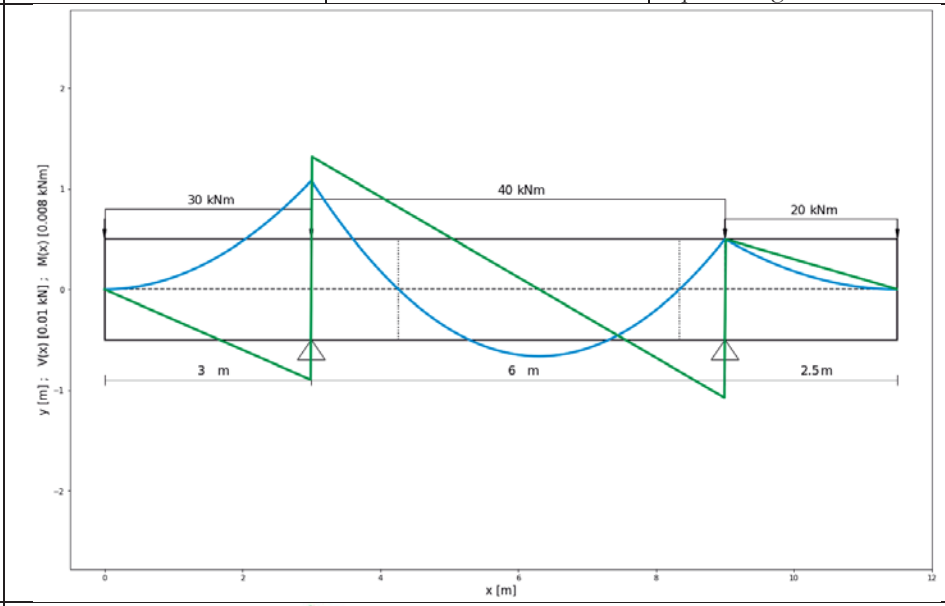
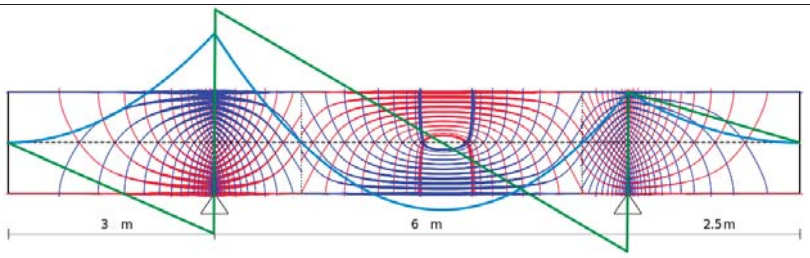
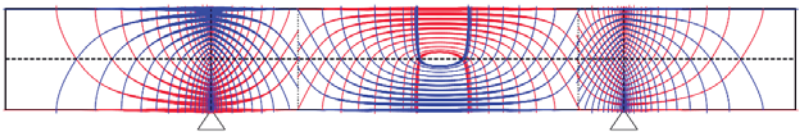
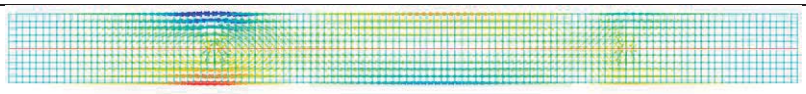
CASO	04	TIPO	Viga con dos voladizos pequeños
Descripción	Viga con dos voladizos con cargas repartidas y puntuales desiguales		
Input	b=0.3 h=1 L1=1.5, L2=6, L3=1.5	q1=20, q2=60, q3=75 P1=40, P2=20, P3=25 d1=0.7, d2=1, d3=1	distortion_Y=1.5 equidist_left=h/10 equidist_mid=h/20 equidist_right=h/10
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	Este es un caso más asimétrico en sus cortes principales, pero en líneas generales se parece mucho al anterior ejemplo.		

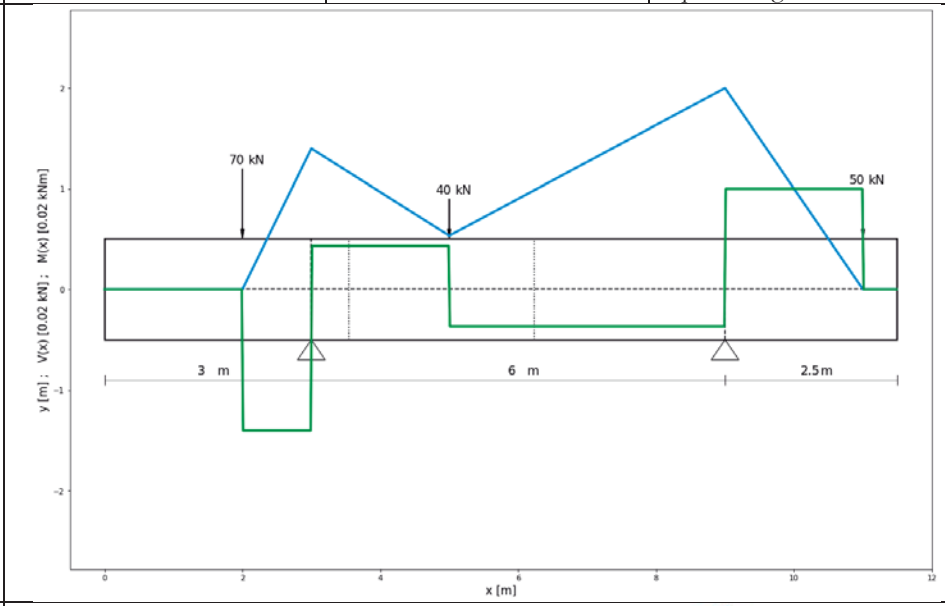
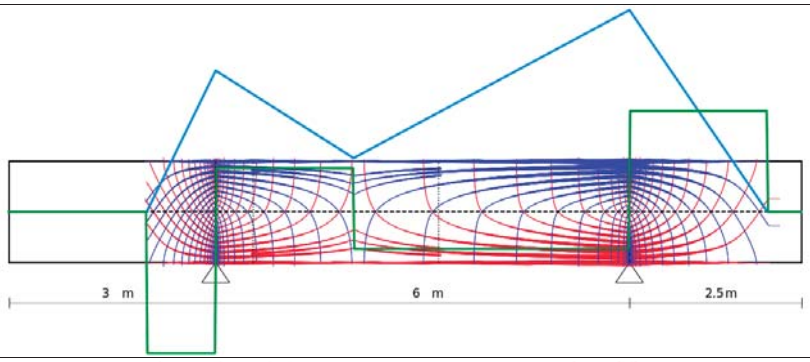
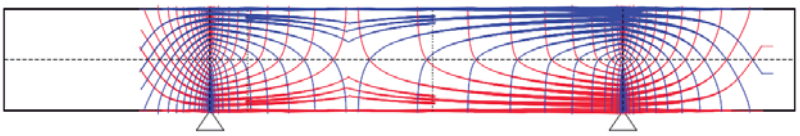
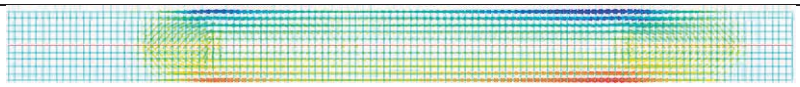
CASO	05	TIPO	Viga con carga sólo en voladizos
Descripción	Viga con dos voladizos con cargas repartidas en los extremos		
Input	b=0.3 h=1 L1=2, L2=6, L3=2	q1=40, q2=0, q3=40 P1=0, P2=0, P3=0 d1=0, d2=0, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	Tanto este caso como el siguiente son bastante interesantes debido a que en la fase central encontramos un caso de flexión pura, es decir, sería tensión máxima y horizontal.		

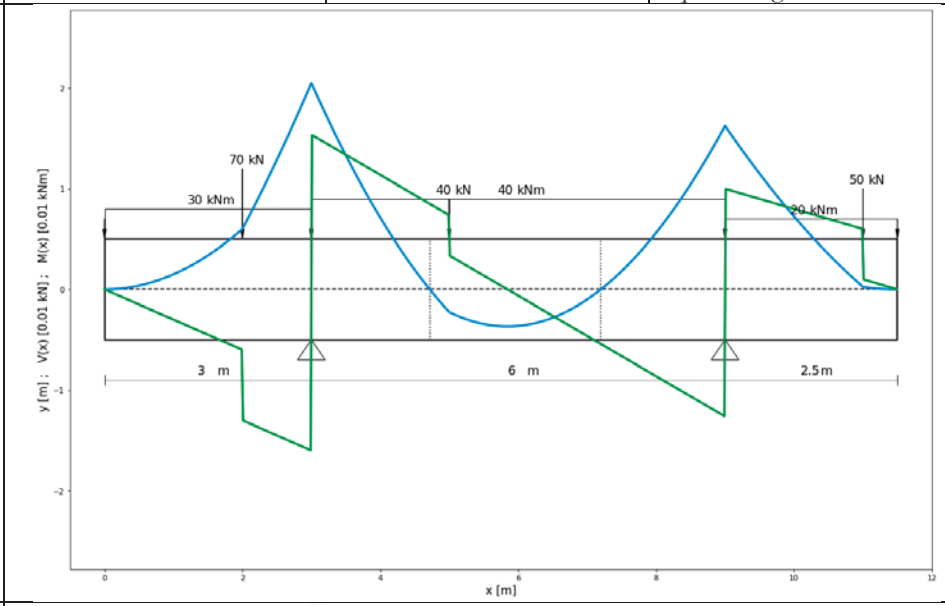
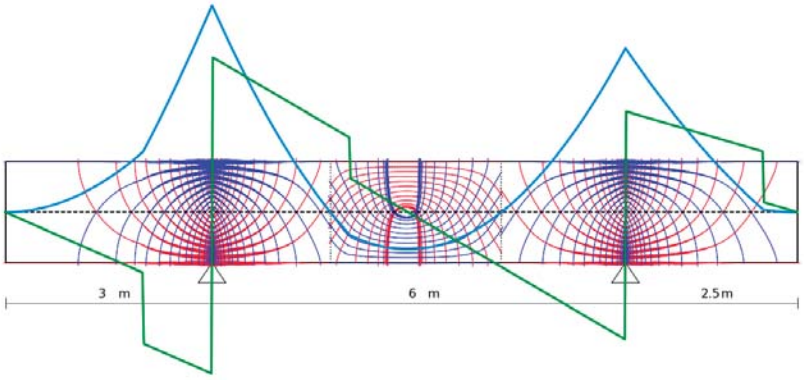
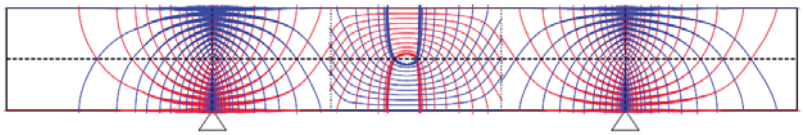
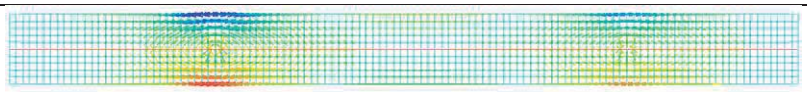
CASO	05	TIPO	Viga con carga sólo en voladizos	
Descripción	Viga con dos voladizos con cargas puntuales en los extremos			
Input	b=0.3 h=1 L1=2, L2=6, L3=2		q1=0, q2=0, q3=0 P1=60, P2=0, P3=60 d1=0.2, d2=0, d3=1.8	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	Este ejemplo es prácticamente igual que el anterior, pero la diferencia clara es que como actúan cargas puntuales, en la parte de los extremos que no actúa carga no existe representación de isostáticas.			

CASO	06	TIPO	Viga en voladizo	
Descripción	Viga en voladizo con carga repartida			
Input	b=0.3 h=1 L1=0, L2=0.01, L3=5		q1=0, q2=0, q3=40 P1=0, P2=0, P3=0 d1=0, d2=0, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	Primero hemos de señalar que este tipo de apoyo no es el correcto, este es sólo el ploteado de dos apoyos muy juntos donde aparecería una singularidad. Lo que hacemos poniendo dos apoyos juntos es hacer un par de fuerzas que es lo que simularía el empotramiento que no podemos dibujar. No sería igual que con una viga biapoyada si hiciéramos efecto espejo y le diéramos la vuelta pues la ley de momentos es parabólica, no curva perfecta. Por eso estas curvas no empiezan con una tensión cercana al cero, si no que empiezan con una pendiente un poco más pronunciada. Estas curvas mueren antes ya que la ley de momentos tiende a cero de manera exponencial.			

CASO	06	TIPO	Viga en voladizo	
Descripción	Viga en voladizo con carga puntual en extremo			
Input	b=0.3 h=1 L1=0, L2=0.01, L3=5		q1=0, q2=0, q3=0 P1=0, P2=0, P3=60 d1=0, d2=0, d3=4.8	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	Este caso, como el anterior, tiene una representación parecida, pero al ser una ley de momento constante, las líneas se mantienen más constantes en su representación. Al igual que ya se ha comentado, aquella parte donde no actúa carga no hay representación de isostáticas.			

CASO	07	TIPO	Viga asimétrica	
Descripción	Viga asimétrica con cargas repartidas			
Input	b=0.3 h=1 L1=3, L2=6, L3=2.5		q1=30, q2=40, q3=20 P1=0, P2=0, P3=0 d1=0, d2=0, d3=0	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	En este ejemplo observamos el mismo factor de error que ya se ha comentado anteriormente. Todo queda bastante homogéneo en representación, ya que las leyes de momentos son casi simétricas.			

CASO	07	TIPO	Viga asimétrica
Descripción	Viga asimétrica con cargas puntuales		
Input	b=0.3 h=1 L1=3, L2=6, L3=2.5	q1=0, q2=0, q3=0 P1=70, P2=40, P3=50 d1=2, d2=2, d3=2	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas			
Isostáticas y solicitaciones			
Isostáticas			
EF			
Comentarios	Este caso es muy variopinto pues se ven varios comentarios ya realizados en una misma gráfica. Se ven las partes blancas allí donde no actúa carga alguna, también se ve uno pequeños flecos en la parte derecha que son errores de programa, y por una cuestión de equidistancias, aparece una representación un tanto llamativa en la parte central, obteniendo así una representación diferente a todas las anteriores.		

CASO	07	TIPO	Viga asimétrica	
Descripción	Viga asimétrica con cargas repartidas y puntuales			
Input	b=0.3 h=1 L1=3, L2=6, L3=2.5		q1=30, q2=40, q3=20 P1=70, P2=40, P3=50 d1=2, d2=2, d3=2	distortion_Y=1.5 equidist_left=h/20 equidist_mid=h/20 equidist_right=h/20
Geometría y cargas				
Isostáticas y solicitaciones				
Isostáticas				
EF				
Comentarios	Estas tres zonas del ejemplo parecen casi idénticas, pero en la parte central hay débil representación, no siendo tan densa como en los apoyos. Igualmente, vuelve a remitir el error de cálculo en derivada del momento. En los saltos de las leyes de cortante se ve en las isostáticas unos intentos de pico como hemos visto en otros casos.			

CONCLUSIONES

A lo largo de este Trabajo Fin de Grado se ha conseguido elaborar a través de una herramienta gráfica la representación clara y fiable del camino interno que recorren las líneas isostáticas en diferentes ejemplos de vigas biapoyadas. Esta representación lineal de una manera gradual y escalada se ha resuelto a través de un software que se ha programado para la obtención de dichos resultados.

El fin siempre didáctico de este Trabajo Fin de Grado, ha estado muy presente en la elaboración de la herramienta, logrando que todo pueda llegar a ser muy práctico y funcional para cualquier tipo de usuario con conocimientos básicos en estructuras arquitectónicas. A través de los ejemplos podemos entender mejor cómo funciona el interior de una viga sometida a cargas de diferente índole y, por ende, descifrar y dar a entender mejor estos mapas conformes isostáticos.

Al haber sido un trabajo sobre elementos unidimensionales, se diferenciaba de aquellos en los que nos habíamos basado precisamente para empezar a indagar y profundizar en él. Estos trabajos eran estudios sobre elementos bidimensionales los cuales tenían como fin principal la optimización en material de elementos estructurales.

Habiendo fijado la atención en la representación de estas tensiones internas, y en los propios problemas que les podían haber surgido, se tomó nota de todo ello desde un principio y se adaptó a una variante unidimensional.

Como bien se mencionaba al principio de este estudio, no existe nada que se haya podido comprobar que se le parezca sobre elementos unidimensionales aplicados a la enseñanza, y lo que existe se centra sobre todo en elementos de dos dimensiones. Puede que sea eso lo que nos haya parecido tan interesante a la hora de realizar esta investigación.

Una vez acotado el caso que vamos a estudiar disponemos la programación de la expresión de dichas líneas isostáticas por el método más eficiente posible: el método discreto. Y así es como llegamos a obtener los diferentes resultados de este estudio.

Teníamos desde un principio este objetivo de representar las líneas isostáticas internas en una viga y sus diferentes casos, y hemos comprobado que tras haber realizado todo aquello que se creía necesario el estudio funcionaba correctamente; hemos visto que el hecho de calcular el cortante como deformada nos ha permitido que las funciones a representar sean continuas, aunque da unos ciertos problemas gráficos.

Hay ocasiones en que las transiciones de unas líneas a otras son muy abruptas, así que, quede constancia clara de que siempre se podría mejorar el fin gráfico de este trabajo.

La continuación natural de este TFG sería hacer los mismos cálculos, pero con pórticos, donde habría unas regiones en las cuales encontramos los nudos de unión entre viga y pilar, y ahí no se pueden trazar realmente líneas isostáticas pues habría que enlazar dos familias de curvas. Este sería un desafío a resolver en el futuro si este trabajo continuara.

Se ha demostrado una herramienta muy útil para entender cómo funcionan las tensiones, los caminos de fuerzas e incluso podríamos decir que se ha descubierto una herramienta estética; de estos patrones podrían salir diseños estructurales de aligeramiento para la arquitectura.

B I B L I O G R A F Í A

Torroja, E. (2010). ***Razón y ser de los tipos estructurales***, Consejo Superior de Investigaciones Científicas, Madrid

Cobb, M. (2018). ***Investigating Principal Stress Lines, Optimization in Gridshell Structures***, Universidad Técnica de Delft

A. G. M. Michell, ***Limits of Economy of Material in Frame-structures,*** University College Nottingham, Inglaterra, 1904, pp. 589

Apuntes Facultad de Ingeniería UNA, ***Estado Plano de Tensiones,*** Mecánica de materiales I

Mueller, C. T., Mark, T, K-M. (2015). Stress Line Generation for Structurally Performative Architectural Design, Optimization in Gridshell Structures, Structure Economy, ETH Zurich

Barbic, J., Sorkine-Hornung O., (2017) ***Rib-reinforced Shell Structure***, volumen 37, versión 7, Pacifics Graphics, Bournemouth University, Reino Unido

Menciones:

Pérez García, A., Alonso Durá, A., Gómez Martínez, F., Alonso Ávalos, J. M., Lozano Lloret, P. (2019). Software de análisis estructural Architrave®

FIN