UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UNA APLICACIÓN SOFTWARE PARA EL APOYO EN LA ENSEÑANZA DEL MÉTODO DE LOS ELEMENTOS FINITOS

TRABAJO FINAL DEL

Máster U. en Diseño y Fabricación Integrada Asistidos por Computador

REALIZADO POR

Pablo Torrejón Cabello

TUTORIZADO POR

Juan Fayos Sancho

CURSO ACADÉMICO: 2019/2020

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Máster Universitario en Diseño y Fabricación Integrada asistidos por Computador

(2018-2020)

Desarrollo de una aplicación software para el apoyo en la enseñanza del método de los elementos finitos

Autor: Pablo Torrejón Cabello

Tutor: Juan Fayos Sancho

Agradecimientos

A Venus. A mi padre, Alfonso. A mi madre, Lorenza. A mi tía, Teresa. A mi suegra, Petra. A la Chupipandi y a la peña J&B. A todos ellos, por ser mi apoyo todos estos años.

A Vicente Colomer Romero, por su ayuda en el uso de NX.

Resumen

El método de elementos finitos es una de las herramientas de las que disponen hoy en día los ingenieros a la hora de probar sus diseños. Desde el diseño de grandes estructuras al diseño de juguetes para bebés, la ingeniería recurre a esta herramienta matemática para analizar cuál va a ser el comportamiento de sus diseños.

Hacer uso de esta herramienta de forma efectiva es una habilidad que puede aprenderse como parte de este máster, pero que puede costar muchos años perfeccionar. La mayoría de los estudiantes de ingeniería toman contacto con ella durante los primeros cursos, como parte de las asignaturas básicas de matemáticas.

Este trabajo final de máster pretende desarrollar una herramienta de cálculo por elementos finitos sencilla de usar y que permita a los estudiantes aprender acerca de los fundamentos matemáticos del método de elementos finitos.

Palabras clave: MEF, Método de elementos finitos, C#, WPF, Educativo

Resum

El mètode d'elements finits és una de les eines de les quals disposen hui dia els enginyers a l'hora de provar els seus dissenys. Des del disseny de grans estructures al disseny de joguets per a bebés, l'enginyeria recorre a aquesta eina matemàtica per a analitzar com serà el comportament dels seus dissenys.

Fer ús d'aquesta eina de manera efectiva és una habilitat que pot aprendre's com a part d'aquest màster, però que pot costar molts anys perfeccionar. La majoria dels estudiants d'enginyeria prenen contacte amb ella durant els primers cursos, com a part de les assignatures bàsiques de matemàtiques.

Aquest treball final de màster pretén desenvolupar una eina de càlcul per elements finits senzilla d'usar i que permeta als estudiants aprendre sobre els fonaments matemàtics del mètode d'elements finits.

Paraules clau: MEF, Mètode d'elements finits, C#, WPF, Educatiu

Abstract

Finite element method is one of the tools available to engineers to test their designs nowadays. From designing bit structures to design of toys for babies, engineering uses this mathematical tool to analyze what is going to be the behavior of its designs.

Using this tool in an effective way is a skill that can be learnt as part of this master, but that can take many years to perfect. Most engineering students' first contact with it during their early courses, as part of their basic mathematics courses.

This project pretends to develop an easy to use finite element method calculation tool, that allows students to learn about the mathematical fundamentals of the finite element method

Keywords: FEM, Finite element method, C#, WPF, Educational

Contenido

Figuras

Introducción

Una de las preguntas más frecuentes que me hicieron durante estos dos años en los que he estado estudiando en la ETSID ha sido "¿Qué hace un informático en un máster como este?".

La respuesta se remonta a algunos años atrás, cuando estudié Ingeniería Técnica en Informática de Sistemas en la Escuela Técnica Superior de Informática Aplicada. Durante el tercer curso elegí la intensificación de Informática Industrial. Me interesaron enormemente asignaturas sobre sistemas de tiempo real o robótica, pero la más interesante fue "CAD / CAM". En esta asignatura, desde una perspectiva orientada a la informática, se nos explicaron los fundamentos del diseño y de la fabricación, llegando incluso a programar un mecanizado simple y a ejecutarlo en una fresadora de tres ejes.

Cuando terminé, continué estudiando. Comencé a estudiar Ingeniería Informática, esta vez en la Facultad de Informática, mientras me lanzaba al mercado laboral. Aunque quise introducirme en el mundo de la informática industrial, las oportunidades que se me presentaron hicieron que me decantara por hacer carrera en el desarrollo de software. Durante los siguientes años trabajé como ingeniero de software, programador y desarrollador, mientras terminaba mis estudios. En este caso, también escogí la intensificación de Informática Industrial, especialmente por una asignatura: "Fabricación Asistida por Computador". La asignatura profundizaba más acerca de la maquinaria, su configuración y su uso, aunque seguía estando orientada a la informática, a fin de dar contexto sobre la fabricación asistida por computador a futuros desarrolladores de software.

Tras terminar mis estudios, quise profundizar por mi cuenta en el diseño y fabricación, pero me resultó complicado. Conseguí aprender acerca de impresión 3D, sabía utilizar paquetes de diseño gratuitos para diseñar piezas que necesitaba e incluso invertí un verano en convertir a control numérico una micro fresadora. Sin embargo, los avances que conseguía eran modestos, seguía necesitando una base sólida a partir de la cual pudiera seguir aprendiendo por mi cuenta.

La oportunidad se presentó unos años después, cuando descubrí el Máster Universitario en Diseño y Fabricación Integrada Asistidos por Computador. Tras esperar un año para inscribirme, pude comenzar los estudios sin sacrificar mi empleo. Creí que las asignaturas que más me interesarían serían "Fabricación Asistida por Computador" o "Aplicaciones Industriales", pero "Ingeniería Asistida por Computador" acabó llamando mi atención. El método de elementos finitos parecía tener aplicaciones enormes, como el modelado de la estructura de edificios o la de objetos más pequeños como un juguete. Por otra parte, se presentaba como un problema computacionalmente interesante, algebra lineal a una escala que no había visto nunca. Sin embargo, se presentaba un problema, yo carecía de la formación previa necesaria para la asignatura.

La mayoría de los compañeros y compañeras del máster eran Ingenieros en Diseño, Ingenieros Industriales o Ingenieros Mecánicos. Todos ellos habían tenido asignaturas de matemáticas en las que se enseñaba el método de elementos finitos, lo que les daba una base para comprender las herramientas de Ingeniería Asistida por computador".

*Por otra parte, en Julio de 2019 cambié de empleo como Ingeniero de Software, pasando a formar parte de la plantilla de **Innoveo AG**. Pasé de desarrollar aplicaciones web de procesamiento de datos a aplicaciones de escritorio. Pese a que la mayoría de la tecnología*

implicada era la misma, hubo un cambio en algunas de las tecnologías empleadas en mi día a día para mi trabajo. Se hacía necesario familiarizarse con el framework de desarrollo de aplicaciones de escritorio ***Windows Presentation Foundation****.*

Esos dos hechos fueron el germen de este proyecto. ¿Y si pudiera desarrollar una aplicación de cálculo por elementos finitos? Podría aplicar mis habilidades como ingeniero de software, familiarizarme con WPF, explorar un problema interesante y al mismo tiempo cubrir un déficit de formación que tenía cuando entré en el máster.

Objetivos de este proyecto

El desarrollo de esta aplicación es un medio para cumplir varios objetivos:

- El primero de ellos es cumplir los requisitos para obtener el título de Máster en Diseño y Fabricación Integrada Asistidos por Computador, que incluye la realización de un Trabajo Final de Máster relacionado con las asignaturas impartidas y con una complejidad suficiente.
- El segundo objetivo es familiarizarse con las tecnologías utilizadas en *Innoveo AG* para el desarrollo de software, especialmente el framework *Windows Presentation Foundation,* en un proyecto real de forma que se puedan adquirir conocimientos útiles para el entorno laboral.
- El tercer objetivo consiste en cubrir el déficit de formación existente en cuanto al método de elementos finitos a través de la investigación necesaria para la implementación de la aplicación.

Alcance y exclusiones

El objetivo de este proyecto es desarrollar una aplicación de cálculo por elementos finitos. Esta aplicación se diseña para ser utilizada en un entorno educativo, de forma que los alumnos puedan explorar la base del método de los elementos finitos. Por ello el problema debe ser lo suficientemente simple como para que los alumnos puedan explorar la base del método mediante ejemplos sencillos sin que se vean desbordados por la complejidad del problema:

- La aplicación se centrará en la resolución del problema estructural elástico lineal.
- El aspecto formativo de la aplicación se centrará en el ensamblado y resolución del sistema de ecuaciones algebraico, resultado de la aplicación del método de los elementos Finitos a la ecuación
- Tres dimensiones: La aplicación resolverá y permitirá problemas en tres dimensiones, dado que es el escenario real de cálculo al que los alumnos se enfrentarán en el futuro.
- Visualización en tres dimensiones: La aplicación mostrará problemas en tres dimensiones, de forma que el usuario pueda definir y ver el problema como se vería en una aplicación CAD/CAE comercial, a fin de que le resulte más intuitivo su uso.
- Complejidad: La aplicación permitirá definir problemas utilizando nodos con tres grados de libertad, siendo estos los desplazamientos en los ejes X, Y, Z. Por tanto, la rotación en los ejes X, Y, Z no se considera en este problema.
- Elementos: Los elementos considerados en esta aplicación son única y exclusivamente barras. Quedan por tanto excluidos elementos tipo placa o sólidos.
- Condiciones de contorno: sólo se considera la aplicación de la condición de contorno de desplazamiento nulo. Queda fuera del alcance la imposición de desplazamientos no nulos.
- Validez: La aplicación debe ofrecer soluciones matemáticamente válidas.
- La aplicación debe funcionar sobre sistema operativo Windows 10, por lo que no se considera el funcionamiento en sistemas Linux o Mac. El soporte en otras versiones de Windows (7, 8.1, Vista o XP) no se considera, aunque la aplicación pueda funcionar.

Marco teórico: El método de elementos finitos

El método de elementos finitos es un método matemático que permite resolver problemas gobernados por ecuaciones diferenciales en derivadas parciales, como análisis estructurales, transferencia de calor o dinámica de fluidos. En este TFM se ha elegido el análisis estructural como objetivo de la aplicación. El objetivo de un análisis estructural es descubrir cómo se deforma un objeto al ser sometido a unas fuerzas y restricciones.

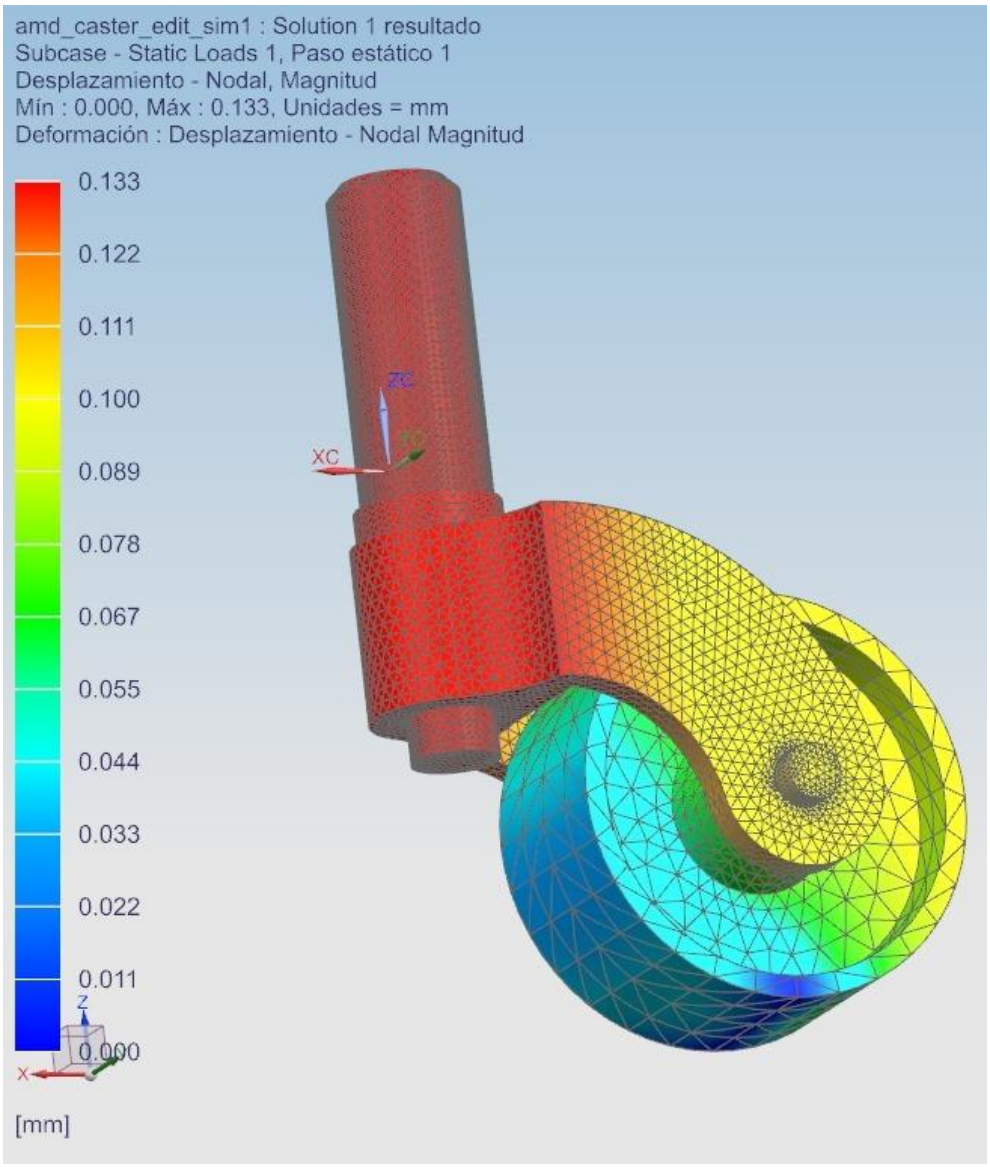

Figura 1 : Ejemplo de análisis estructural con NX

Modelar y resolver un problema de deformación a través de la resolución analítica de las ecuaciones diferenciales es enormemente complejo, algo al alcance de solo unas pocas personas incluso en problemas relativamente sencillos. Discretizar el problema, en cambio, permite que pueda ser resuelto de forma más sencilla utilizando un computador.

Por otra parte, discretizar el problema tiene un coste y es que pasamos de tener solución en todos y cada uno de los puntos que lo componen a solo tener en una serie de puntos, denominados nodos.

Fuera de estos puntos, el resultado del problema es desconocido, por lo que es necesario interpolarlo. Los nodos se agrupan en elementos. Los elementos proporcionan una función de interpolación, que permite conocer el resultado en aquellos puntos intermedios que quedan dentro de él.

Esta función de interpolación permite construir un sistema de ecuaciones, que relaciona los desplazamientos de los distintos nodos del elemento con las fuerzas aplicadas en ellos.

$$\{f\} = [k] \cdot \{d\}$$

Este sistema de ecuaciones por sí mismo no es útil, ya que no se conocen las distintas fuerzas y las deformaciones de los nodos. Para ello, los distintos sistemas de ecuaciones se ensamblan en un sistema de ecuaciones global que abarca todo el problema, que relaciona todas las fuerzas que intervienen en el problema con las deformaciones en cada uno de los nodos:

$$\{F\} = [K] \cdot \{D\}$$

Sin embargo, este sistema de ecuaciones todavía no es resoluble. Se hacen necesarias ciertas condiciones de contorno para poder resolverlo. Las condiciones de contorno pueden ser fuerzas (se conoce el valor de la fuerza que está siendo aplicada sobre uno o más nodos) o bien desplazamientos (se sabe cuál es el valor del desplazamiento de ciertos nodos). Una vez se conocen suficientes condiciones de contorno, el problema se puede resolver.

Marco teórico de la aplicación

En este TFM se ha escogido un problema de análisis estructural como aplicación del método de elementos finitos. Es un problema en tres dimensiones con tres grados de libertad, en el que los elementos se unen mediante elementos tipo “barra”, de forma que un elemento conecta dos nodos y utiliza una ecuación lineal como función de interpolación.

En los nodos se considera su posición en el espacio (X, Y, Z), así como su desplazamiento por el mismo (dX, dY, dZ). También es posible fijar de forma independiente los desplazamientos de los nodos para poder introducir una condición de contorno. Al fijar uno de los ejes del nodo se asume que su desplazamiento en ese eje es cero.

En los elementos se considera la longitud de este (determinada por las coordenadas de los elementos que une), la sección transversal del mismo y el módulo de Young. Estas propiedades permiten calcular la constante k del elemento, que se utilizará más adelante.

Desarrollo matemático de la aplicación

Dado un problema válido, se puede resolver el problema de deformación siguiendo una serie de pasos.

Cálculos del elemento

En primer lugar, es necesario construir la matriz local de rigidez de cada uno de los elementos. Tenemos elementos tipo barra, en los que el origen de coordenadas local está en el primer nodo y la barra se extiende a lo largo del eje X hasta llegar al segundo elemento. Los elementos están modelados como muelles, de forma que las fuerzas y los desplazamientos se dan a lo largo del eje X. Con esa información se puede plantear un sistema de ecuaciones lineales tal que:

$$\begin{pmatrix} f1xLocal \\ f2XLocal \end{pmatrix} = K \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{pmatrix} d1xLocal \\ d2xLocal \end{pmatrix}$$

Este sistema de ecuaciones no es útil en el sistema de coordenadas local al elemento. Para poder realizar el ensamblaje de los sistemas de ecuaciones, es necesario que todos estén en el sistema global de coordenadas. Se hace necesario construir una matriz de transformación:

$$T = \begin{bmatrix} Cx & Cy & Cz & 0 & 0 & 0 \\ 0 & 0 & 0 & Cx & Cy & Cz \end{bmatrix}$$

- Cx es el coseno del ángulo formado entre el eje X local y el eje X global
- Cy es el coseno del ángulo formado entre el eje Y local y el eje Y global
- Cz es el coseno del ángulo formado entre el eje Z local y el eje Z global

Para poder aplicar la matriz de transformación, se realiza el siguiente desarrollo:

$$\{Fuerzas\ globales\} = [T]^T \cdot \{Fuerzas\ locales\}$$

$$\{Desplazamientos\ locales\} = [T] \cdot \{Desplazamientos\ globales\}$$

$$\{Fuerzas\ locales\} = [k\ local] \cdot \{Desplazamientos\ locales\}$$

$$\{Fuerzas\ locales\} = [k\ local] \cdot [T] \cdot \{Desplazamientos\ globales\}$$

$$[T]^T \cdot \{Fuerzas\ locales\} = [T]^T \cdot [k\ local] \cdot [T] \cdot \{Desplazamientos\ globales\}$$

$$\{Fuerzas\ globales\} = [T]^T \cdot [k\ local] \cdot [T] \cdot \{Desplazamientos\ globales\}$$

$$[K\ global] = [T]^T \cdot [k\ local] \cdot [T]$$

De este modo se obtiene la matriz K global del elemento. Estas son matrices de 6 x 6 (6 filas por 6 columnas), que a su vez se subdividen en 4 bloques de 3 x 3

$$\begin{bmatrix} A & A & A & | & B & B & B \\ A & A & A & | & B & B & B \\ A & A & A & | & B & B & B \\ - & - & - & + & - & - & - \\ C & C & C & | & D & D & D \\ C & C & C & | & D & D & D \\ C & C & C & | & D & D & D \end{bmatrix}$$

A partir de las distintas matrices K globales de los distintos elementos se ensambla la matriz K global del problema, colocando las secciones 3 x 3 de cada elemento en posiciones específicas. Estas posiciones vienen determinadas por la matriz de correspondencia

Ensamblaje de la matriz global

El primer paso del ensamblaje es construir la matriz de correspondencia. Es una matriz en la que filas y columnas están etiquetadas con los distintos nodos del problema y cuyas posiciones contienen el listado de elementos que están en contacto con dicho nodo.

La matriz de correspondencia es una representación de la matriz K global. Cada una de las posiciones de la matriz representa un bloque de 3 X 3. En cada una de esas posiciones se inserta una sección de 3 X 3. En caso de que en una posición de la matriz de correspondencia haya múltiples elementos, las secciones 3 X 3 correspondientes se suman.

El proceso de ensamblaje se puede comprobar con el siguiente ejemplo:

- El problema tiene 3 nodos y dos elementos

- El elemento 1 conecta el nodo 1 con el nodo 3
- El elemento 2 conecta el nodo 2 con el nodo 3

La matriz de correspondencia queda por tanto con la siguiente disposición:

$$\begin{pmatrix} & Nodo\ 1 & Nodo\ 2 & Nodo\ 3 \\ Nodo\ 1 & Elemento\ 1 & Elemento\ 1 & \dots \\ Nodo\ 2 & Elemento\ 1 & Elemento\ 1\ y\ Elemento\ 2 & Elemento\ 2 \\ Nodo\ 3 & \dots & Elemento\ 2 & Elemento\ 2 \end{pmatrix}$$

Al mismo tiempo, las matrices K globales de los elementos 1 y 2 tienen la siguiente disposición:

$$K1 = \left[\begin{array}{ccc|ccc} A & A & A & B & B & B \\ A & A & A & B & B & B \\ A & A & A & B & B & B \\ \hline C & C & C & D & D & D \\ C & C & C & D & D & D \\ C & C & C & D & D & D \end{array}\right]$$

$$K2 = \left[\begin{array}{ccc|ccc} E & E & E & F & F & F \\ E & E & E & F & F & F \\ E & E & E & F & F & F \\ \hline G & G & G & H & H & H \\ G & G & G & H & H & H \\ G & G & G & H & H & H \end{array}\right]$$

Por tanto, la matriz K global del problema se ensambla quedando del siguiente modo

$$\left[\begin{array}{ccc|ccc|ccc} A & A & A & B & B & B & 0 & 0 & 0 \\ A & A & A & B & B & B & 0 & 0 & 0 \\ A & A & A & B & B & B & 0 & 0 & 0 \\ \hline C & C & C & D+E & D+E & D+E & F & F & F \\ C & C & C & D+E & D+E & D+E & F & F & F \\ C & C & C & D+E & D+E & D+E & F & F & F \\ \hline 0 & 0 & 0 & G & G & G & H & H & H \\ 0 & 0 & 0 & G & G & G & H & H & H \\ 0 & 0 & 0 & G & G & G & H & H & H \end{array}\right]$$

Solución del sistema de ecuaciones global

Una vez se dispone de la matriz global K, se puede plantear el sistema de ecuaciones globales. Sin embargo, para poder resolverlo es necesario introducir las condiciones de contorno.

En primer lugar, las fuerzas del problema se introducen como valores numéricos en el vector de fuerzas globales. A continuación, las restricciones fijas se introducen como ceros en el vector de desplazamientos globales, de un modo similar al siguiente ejemplo:

- El eje X del nodo 1 es fijo
- El eje Y y el eje Z del nodo 2 son fijos
- Hay una fuerza aplicada en el nodo 3 con vector (1, 1, 2) N

$$\begin{bmatrix} A & A & A & | & B & B & B & | & 0 & 0 & 0 \\ A & A & A & | & B & B & B & | & 0 & 0 & 0 \\ A & A & A & | & B & B & B & | & 0 & 0 & 0 \\ - & - & - & + & - & - & - & + & - & - & - \\ C & C & C & | & D+E & D+E & D+E & | & F & F & F \\ C & C & C & | & D+E & D+E & D+E & | & F & F & F \\ C & C & C & | & D+E & D+E & D+E & | & F & F & F \\ - & - & - & + & - & - & - & + & - & - & - \\ 0 & 0 & 0 & | & G & G & G & | & H & H & H \\ 0 & 0 & 0 & | & G & G & G & | & H & H & H \\ 0 & 0 & 0 & | & G & G & G & | & H & H & H \end{bmatrix} \begin{pmatrix} 0 \\ d1y \\ d1z \\ d2x \\ 0 \\ 0 \\ d3x \\ d3y \\ d3z \end{pmatrix} = \begin{pmatrix} f1x \\ f1y \\ f1z \\ f2x \\ f2y \\ f2z \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

El siguiente paso consiste en compactar el sistema de ecuaciones. Es necesario eliminar aquellas filas y columnas que corresponden a restricciones fijas, en las que el desplazamiento se sabe que es cero. El ejemplo anterior quedaría de la siguiente forma:

$$\begin{bmatrix} A & A & | & B & | & 0 & 0 & 0 \\ A & A & | & B & | & 0 & 0 & 0 \\ - & - & + & - & + & - & - & - \\ C & C & | & D+E & | & F & F & F \\ - & - & + & - & + & - & - & - \\ 0 & 0 & | & G & | & H & H & H \\ 0 & 0 & | & G & | & H & H & H \\ 0 & 0 & | & G & | & H & H & H \end{bmatrix} \begin{pmatrix} d1y \\ d1z \\ d2x \\ d3x \\ d3y \\ d3z \end{pmatrix} = \begin{pmatrix} f1y \\ f1z \\ f2x \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

Si las restricciones del problema son correctas, el número de ecuaciones es suficiente para resolver el sistema de ecuaciones. En caso contrario, es necesario añadir restricciones. Una vez resuelto el problema, se dispone de todos los desplazamientos. Esto permite sustituirlos en el sistema de ecuaciones y obtener las fuerzas de reacción.

Marco teórico II: Sobre el desarrollo de software

La aplicación se ha desarrollado siguiendo el patrón de diseño habitual en Windows Presentation Foundation, el patrón MVVM (Model-View-ViewModel). MVVM es un patrón de diseño de software que facilita el desarrollo de la interfaz gráfica de usuario al separar el Modelo (Model) de la Vista (View) a través de una capa denominada ViewModel:

- El **Modelo** se encarga de contener la lógica de negocio de la aplicación, que sirve como representación de la realidad
- El **ViewModel** se encarga de traducir y restringir la información que se muestra del modelo a la vista. Por una parte, se encarga de la compartimentación de la información, de forma que cada vista tenga disponible únicamente la información que necesita para su trabajo. Por otra parte, se encarga de controlar la "lógica de presentación", decidiendo que debe mostrarse y que ocultarse
- La **Vista** se encarga de representar la información proporcionada por el ViewModel, habitualmente a través de un lenguaje de marcado. En el caso de Windows Presentation Foundation, este lenguaje se llama XAML (eXtensible Application Markup Language).

Una de las características de este patrón de diseño es que permite la creación de aplicaciones muy reactivas, de forma que los cambios que se introducen en una vista alteran instantáneamente el viewmodel que a su vez altera el modelo instantáneamente. Esta característica fue aprovechada en la aplicación para intentar que su uso fuera más rápido e intuitivo. En vez de tener un botón que desencadenara el botón de cálculo en la capa de viewmodel, los cambios en el modelo (el problema) provocan que el viewmodel se vea alterado y que los cambios en las matrices resultantes se vean al momento.

Esta decisión de diseño tiene ventajas, puesto que permite a los usuarios hacer pequeños cambios y observar cómo se traducen en pequeños cambios en las matrices. Del mismo modo, los usuarios evitan estar pulsando continuamente el botón de recalculo cada vez que se hace un cambio.

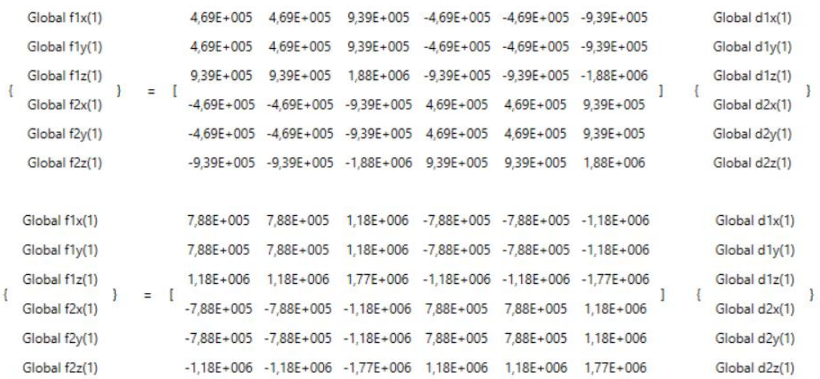

Figura 2: Matriz de un elemento tras un cambio en las coordenadas de un nodo

Por otra parte, esta decisión es inconsistente con los paquetes de cálculo comerciales, que requieren de una orden explicita para calcular la solución. Además, el recalculo constante puede suponer un problema en caso de que el problema se complique en exceso, pero ese último inconveniente es aceptable debido a que los problemas de alta complejidad no serían el objetivo de una aplicación educativa.

Desarrollo

Paso 1: Prueba de concepto

La primera etapa en el desarrollo de la aplicación ha sido una fase de prueba de concepto. El plan original consistía en desarrollar una aplicación con una limitación de dos dimensiones, X e Y. Representar el problema podía realizarse mediante un control *canvas*[13], un lienzo virtual en el que se pueden dibujar figuras geométricas y manipularlas con relativa sencillez.

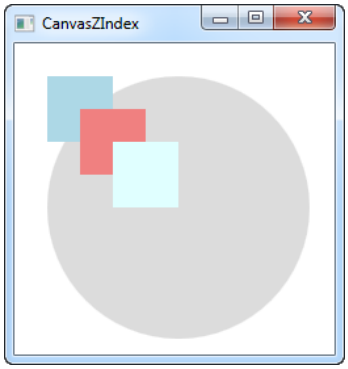

Figura 3: Ejemplo de uso de canvas[14]

Sin embargo, la recomendación inicial del director del TFM fue de no restringir la aplicación a dos dimensiones, ya que una aplicación que considerara el movimiento en los tres ejes podía resultar más práctica, sin un incremento notable en la complejidad de la implementación del cálculo, puesto que una dimensión extra solo implicaba más filas y columnas en las matrices del problema.

Pasar a la tercera dimensión presentaba un problema. Se requería un motor gráfico que permitiera representar objetos en tres dimensiones, así como desplazar y rotar la cámara que permitiría al usuario ver la escena. Afortunadamente, .net dispone del control *Viewport3D*[15]. Este control actúa como un pequeño motor 3D, permitiendo definir geometrías, fuentes de luz y una cámara.

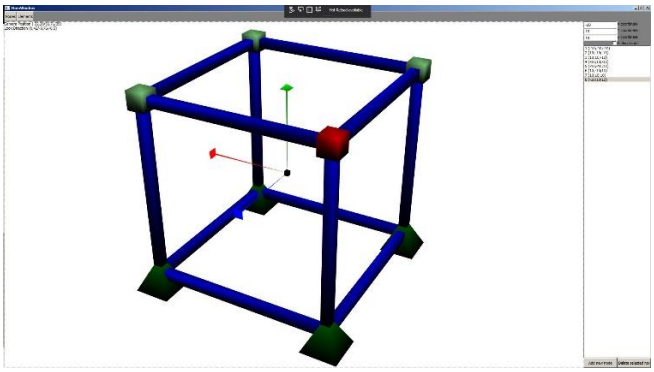

Figura 4: Captura de una versión inicial de la aplicación

A primera vista, utilizar una vista en 3D no es mucho más complicado que una en 2D, a excepción de la cámara. En una vista en dos dimensiones, no es necesario considerar una cámara puesto que nos podemos limitar a mover el control *canvas* a lo largo de los ejes X e Y, formando una ventana virtual que nos dé un efecto similar a una cámara. Sin embargo, en el control 3D es necesario disponer de una cámara con 6 grados de libertad.

Para poder decidir si se optaba por una aplicación en 3D se decidió realizar una prueba de concepto, consistente en implementar un motor gráfico mínimo alrededor del control *Viewport3D*. El alcance de la prueba consistía en disponer de una escena en 3D fija, pero con una cámara que pudiera ser desplazada y rotada usando el teclado y el ratón.

El mayor reto de este prototipo consistió en reaprender acerca de gráficos por computador y acerca del comportamiento de la cámara. Superado ese obstáculo, la prueba consistió en dibujar un problema de prueba y hacer que los objetos de la escena se movieran o rotaran para crear la ilusión del movimiento de la cámara, que se "movía" de acuerdo con las indicaciones del usuario a través del teclado y el ratón.

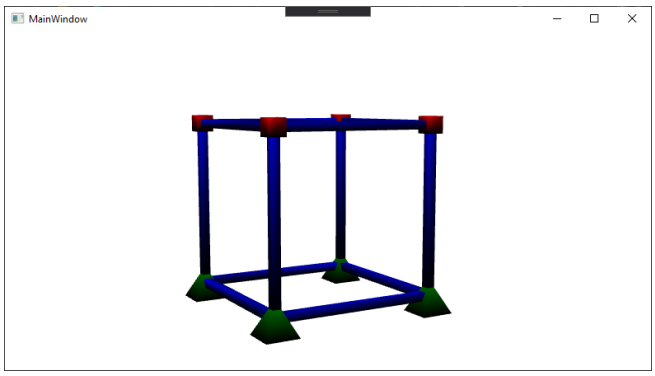

Figura 5: 3DPoc en su versión de prototipo

A esta aplicación prototipo se le denominó *3DPoC*, del inglés *3D Proof of Concept*. Dado que se consiguió implementar la cámara, parcialmente en ese momento, se decidió evolucionar ese prototipo hacia la aplicación definitiva.

Paso 2: Evolución del prototipo

El siguiente paso del proceso de desarrollo consistió en añadir la interfaz de usuario necesaria para poder definir problemas, de forma que se pudieran definir nodos, elementos y fuerzas. En este punto los materiales no se consideraban como entidades independientes, sino que el Módulo de Young se consideraba como una propiedad más de los elementos.

Tras este avance, se comienzan a definir los mecanismos de cálculo, al mismo tiempo que se definen las interfaces gráficas necesarias para mostrar los distintos pasos del proceso al usuario. Se decide optar por mantener una ventana principal con una vista en 3D, junto con una barra de menús horizontal en la parte superior de la pantalla y una zona de detalles en el lateral derecho de la pantalla.

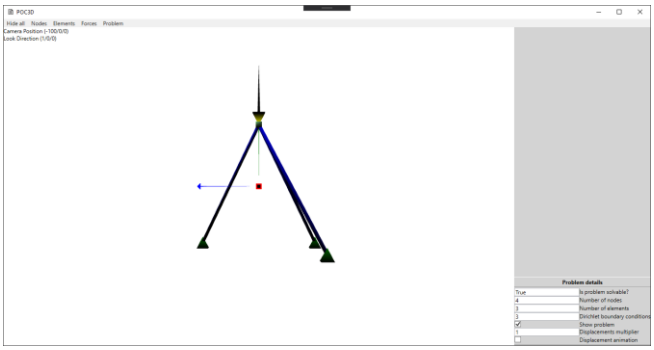

Figura 6: Prototipo con menús y capacidad para definir problemas

En cuanto a la información de los cálculos, se opta por mostrarlos en ventanas separadas. De esta forma los usuarios tienen la opción de trabajar con la aplicación en múltiples escritorios, dejando la ventana principal en uno, mientras que se observan los cambios en los cálculos en otras ventanas.

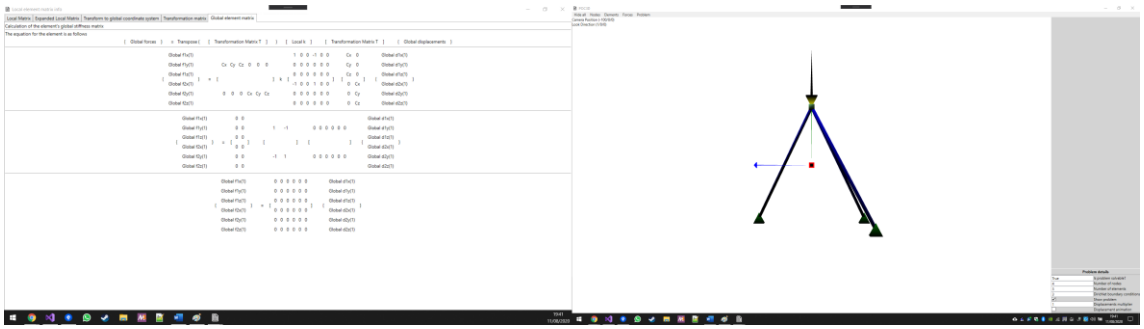

Figura 7: Ejemplo de uso de la aplicación en una configuración multi-monitor

Respecto al motor de cálculo, este sigue un desarrollo en paralelo con las propias interfaces gráficas usadas para mostrar los pasos intermedios, que a su vez van en paralelo a la investigación sobre el método de elementos finitos. Estas interfaces se desarrollan con un doble propósito:

- Por una parte, son necesarias para la enseñanza, dado que muestran pasos intermedios que los paquetes comerciales no muestran y que son útiles para la corrección de ejercicios hechos a meno.
- Por otra parte, esa misma información es necesaria para la depuración de la aplicación, especialmente en los primeros estadios de desarrollo, en los que fue necesario reescribir la lógica de cálculo en varias ocasiones debido a los errores.

Además, algunas de estas ventanas no muestran información del problema, sino que muestran explicaciones teóricas sobre cómo realizar los cálculos. Esto es de utilidad tanto al utilizar la aplicación, así como a la hora de depurarla.

La primera de las ventanas auxiliares en ser añadida es la de información de las matrices de elemento. Permite observar cada uno de los pasos desde la definición de la matriz global del

elemento hasta el resultado semi-final de la matriz del elemento en el sistema de coordenadas globales, así como los pasos necesarios para pasar del sistema de coordenadas local al global.

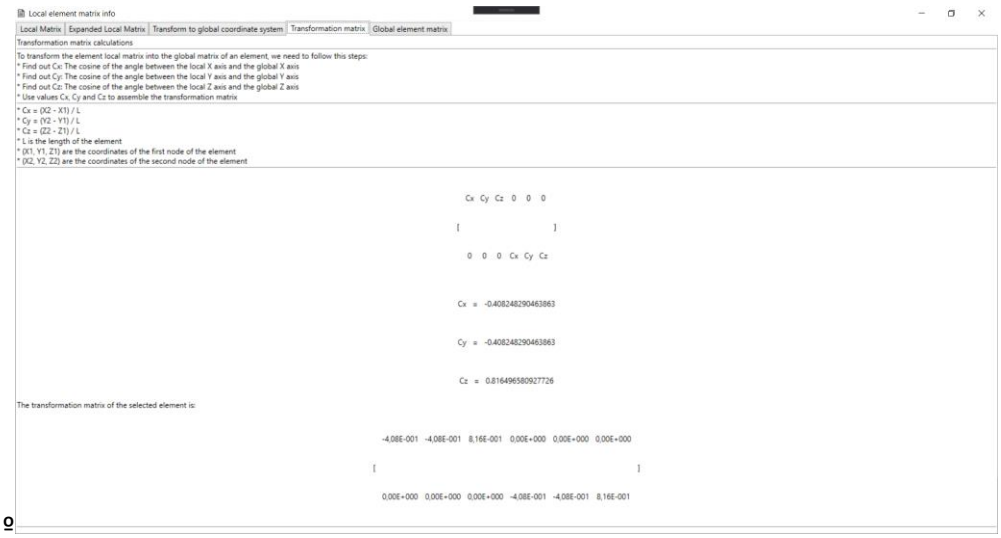

Figura 8: Detalle de la información teórica en las interfaces de usuario

Finalmente, a la aplicación se añadió una última ventana. Esta permite mostrar los resultados globales del problema, el proceso de ensamblaje de la matriz de rigidez global y su posterior resolución para obtener las deformaciones y las fuerzas resultantes en cada uno de los nodos.

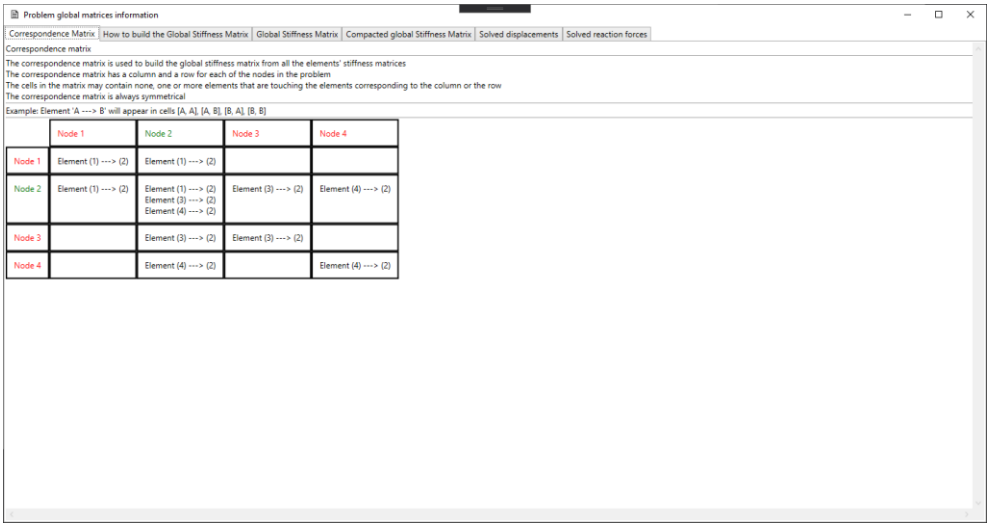

Figura 9 : Detalle de la ventana de información de las matrices globales

Otro de los puntos a resolver una vez se dispuso de los resultados de cálculo consistía en su representación. En NX o en Ansys workbench, los resultados de un cálculo son visibles de forma gráfica o animada al usuario. Para disponer de esta funcionalidad se decidió

implementar un sistema de nodos y elementos "resultado" que se mostrarían en la interfaz gráfica.

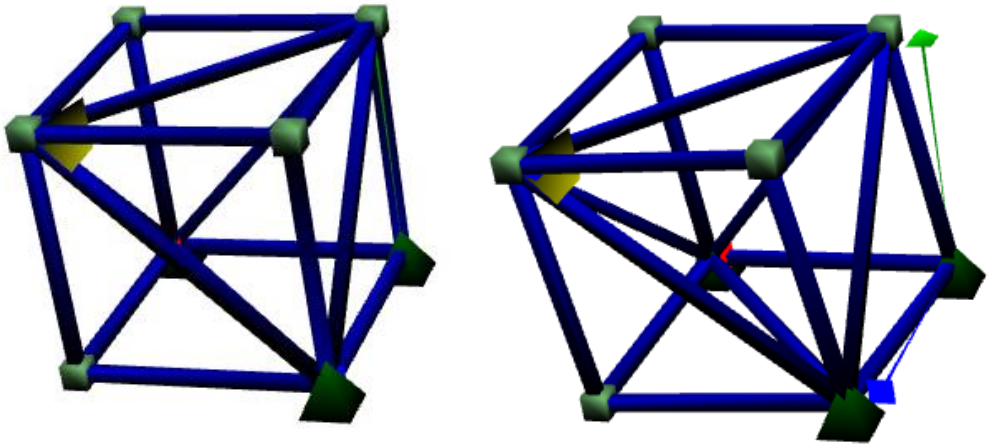

Figura 10 : Representación del modelo vs Representación del resultado exagerado

Los resultados se hicieron accesibles a través de una opción en la interfaz de usuario que permitía cambiar entre el problema y los resultados. También se agregó una opción para regular el multiplicador, de forma que se pudieran hacer evidentes deformaciones menores, junto con una opción para darle movimiento a la deformación y hacerlos más evidentes.

El resultado de este paso se liberó como la versión *Alpha-001*, la primera versión completamente funcional (desde el punto de vista del cálculo) pero que aún seguía necesitando de una serie de mejoras de usabilidad.

Paso 3: Mejoras de usabilidad

Tras la constatación de haber cumplido el objetivo principal del proyecto durante el paso 2, surge la necesidad de hacer una serie de mejoras en la aplicación:

- En primer lugar, la aplicación carece de pruebas unitarias o de integración, lo que redunda en una muy baja mantenibilidad del código.
- La aplicación está construida de forma monolítica, por lo que una división en proyectos lógicos se hace necesaria. Esto es otro factor que influye en la mantenibilidad del código.
- El motor 3D de la aplicación tiene una serie de errores y limitaciones. El movimiento de la cámara es poco fluido y presenta saltos, la cámara es incapaz de hacer un movimiento de alabeo (solo dispone de 5 grados de libertad) y los controles son excesivamente diferentes a los que cabría esperar en una aplicación de cálculo. Esto influye en la usabilidad de la aplicación.
- No se pueden crear o guardar problemas. Este inconveniente influye enormemente en la usabilidad de la aplicación.

- Existen pequeñas inconsistencias en la interfaz de usuario. Esto presenta algunos problemas de usabilidad de la aplicación, aunque en menor medida.

Estos inconvenientes fueron solucionados o mitigados en las distintas versiones comprendidas entre *Alpha-002 y Alpha-008*.

La solución a los dos primeros puntos se llevó a cabo en paralelo, creando proyectos para contener las distintas capas de modelo, viewmodel y vista, al tiempo que se iban generando pruebas unitarias para las distintas clases. Estas pruebas, además, sirvieron para realizar mejoras no-funcionales al código, a fin de mejorar la mantenibilidad de este.

La mejora del motor 3D se llevó a cabo en la versión *Alpha-003*. En este caso la mejora consistió en la reescritura y modularización del código relativo a la interacción con el teclado y ratón, así como la reescritura de parte de la lógica de la cámara, que impedía el movimiento de rotación que faltaba en el proyecto original.

Se añadió la lógica necesaria para cargar y guardar problemas, algo que requería de la creación de una capa de acceso a datos por debajo del modelo, de forma que se pudiera serializar y deserializar una representación del modelo sin perder información. Basada en esta lógica, también se añadió la funcionalidad de crear un nuevo problema, borrando el problema existente en memoria.

En cuanto a las inconsistencias en la interfaz, estas fueron solucionadas principalmente en la versión *Alpha-005*, aunque algunas de ellas se solucionaron a la par que se hacían otras mejoras.

Paso 4: Verificación

El último paso de la aplicación consistió en verificar que los resultados de cálculo eran fiables. Se trabajó junto a Juan Fayos Sancho, el director de este TFM, para definir un problema que pudiera ser resuelto por la aplicación y por un paquete CAE comercial.

El primer paso de esta verificación fue solucionar una limitación de la aplicación. Originalmente se diseñaron los nodos para poder ser fijos o móviles, de forma que al fijarlos se restringían sus tres grados de libertad. Esto era un inconveniente al ser poco realista, por lo que se agregó la funcionalidad necesaria para definir el estado de los tres grados de libertad de los nodos de forma más granular. A continuación, se definió el problema de la siguiente forma:

- 8 nodos, formando un cubo de un metro de arista
- 14 elementos tipo barra, uniendo los nodos formando triángulos, a fin de evitar problemas con las limitaciones de la aplicación. Las barras se definieron con un radio de 1 centímetro.
- 6 restricciones fijas en tres nodos
- Un material (acero) con características definidas (E=2.1E11, nu=0.33)
- Una fuerza de 1N en uno de los ejes de uno de los nodos

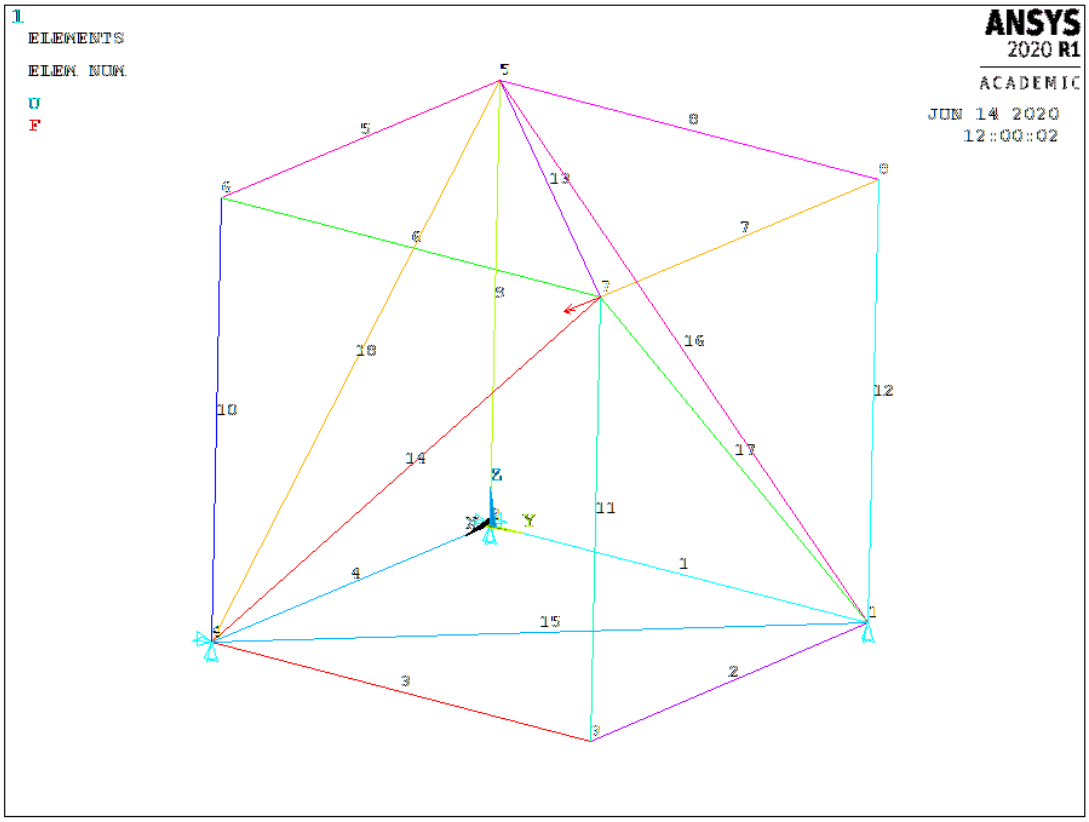

Figura 11: Representación del problema en Ansys

En primer lugar, el director del TFM proporcionó una solución al problema tal y como la había calculado con Ansys. El hecho de que esta primera validación fuera realizada por el director se debió a la falta de conocimiento acerca de cómo realizar cálculos en Ansys APDL y a la limitación de Ansys Workbench (la herramienta utilizada en la asignatura Ingeniería Asistida por Ordenador) para modelar barras.

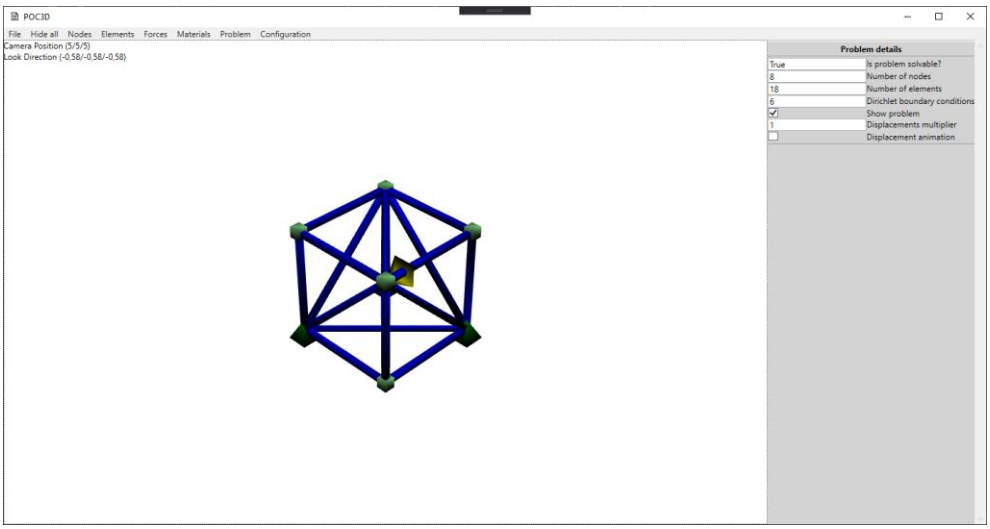

Figura 12: Representación del problema en 3DPoc

A continuación, se recogieron los resultados de desplazamiento proporcionados por la aplicación mostrados en la ventana de información de las matrices globales y se compararon con los resultados obtenidos con Ansys. Los resultados se correspondían, aunque había ciertos valores que, en lugar de cero, eran valores muy próximos a este, del orden de 10^{-10}.

El siguiente paso consistió en construir una prueba de integración que repitiera de forma automática el paso anterior, construyendo el problema tal y como se había definido, obteniendo los resultados y comparando estos con los resultados obtenidos de Ansys.

```
[Test]
0 references
public void AnsysInitialValidationTests()
{
    //Arrange
    const double MaximumAllowedDelta = 2.5E-9;

    var projectViewModel = SetupScenario();

    var expectedResults = BuildExpectedResultsFromAnsys();

    //Act
    var displacementResults = projectViewModel.ProblemCalculationViewModel.FullSolvedDisplacementsVector;

    //Assert
    foreach(var index in Enumerable.Range(0, displacementResults.Rows))
    {
        var calculated = displacementResults[index, 0];
        var expected = expectedResults[index];

        var displacementDelta = Math.Abs(calculated - expected);

        var isBelowThreshold = displacementDelta < MaximumAllowedDelta;

        Assert.That(isBelowThreshold, Is.True, $"Delta is {displacementDelta}, greater than {MaximumAllowedDelta}");
    }
}
```

Figura 13: Prueba de integración de la validación

Realizado este paso, se decidió hacer el mismo ejercicio con NX. Esta verificación fue realizada por el alumno, gracias a que se disponía de una licencia que se podía utilizar en remoto al estar cursando la asignatura de *Aplicaciones CAE*. En esta asignatura y sus tutorías se obtuvieron los conocimientos necesarios para definir y resolver el problema gracias al profesor, Vicente Colomer Romero.

- El primer paso consistió en crear un archivo de modelado vacío, con sus correspondientes archivos FEM y SIM.
- A continuación, se activó el archivo FEM y se crearon los 8 puntos correspondientes a los nodos
- Hecho esto, se dibujaron líneas que unieran los puntos que estaban conectados por nodos, usando la herramienta “Línea punto a punto”
- A partir de las líneas, se creó un mallado 1D para definir los elementos. Es importante señalar la necesidad de activar la opción “Fusionar los nodos” en esta malla, ya que tenerla desactivada provoca errores durante el cálculo.
- Definida la malla, se definió la sección y el material de los elementos

- Se definieron las cargas y las restricciones de la forma habitual
- Se resolvió el problema y se consultaron los resultados.
- Aunque los resultados eran “visibles” a través de los colores, se hacía necesario obtener los resultados de desplazamiento de los nodos, por lo que fue necesario utilizar la herramienta “Identificar los resultados” y marcar los 8 nodos del problema, a fin de obtener un fichero Excel con los resultados precisos del desplazamiento en cada uno de los ejes de cada uno de los nodos.
- Tras identificar cada uno de los nodos, se procedió a construir una prueba de integración análoga a la construida con Ansys.

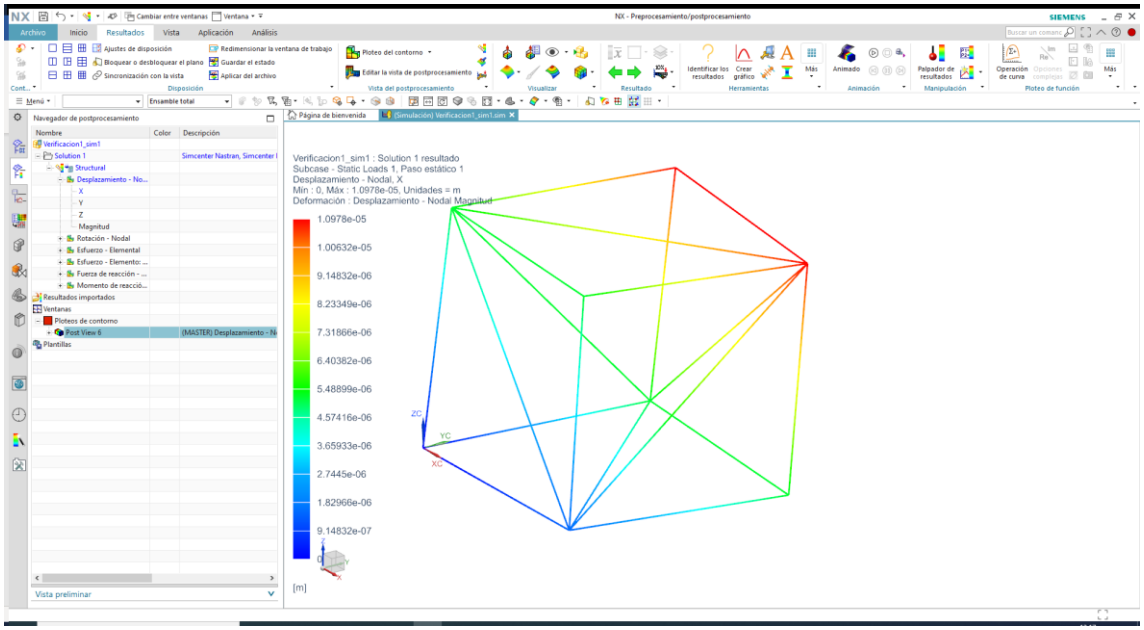

Figura 14: Resultados con NX

El resultado final fueron dos verificaciones cruzadas, que permitieron asegurar que los resultados proporcionados por la aplicación gozaban de fiabilidad suficiente.

- Para la prueba con Ansys, los resultados tuvieron un error máximo de $2{,}5 \cdot 10^{-9}$
- Para la prueba con Nx, los resultados tuvieron un error máximo de $6 \cdot 10^{-9}$

Tecnología

La tecnología, aplicaciones y herramientas utilizadas para desarrollar de la aplicación se enumeran a continuación:

- En primer lugar, el lenguaje de programación escogido ha sido **C#**, uno de los lenguajes soportados en el framework .net de Microsoft. La versión escogida de dicho framework ha sido la 4.8, al ser la última versión de este.
- El entorno de desarrollo utilizado ha sido **Microsoft Visual Studio 2019 Community Edition**. Esta es una versión gratuita, pero completamente funcional de Visual Studio, disponible para estudiantes, ONGs y su uso en desarrollo de Software Libre.
- Como herramienta adicional, se ha utilizado **ReSharper**, una extensión para Visual Studio que mejora la productividad y facilita el trabajo de los desarrolladores.
- Como sistema de control de código fuente se ha utilizado **Git**. Sobre este se ha utilizado **Atlassian Sourcetree**, una interfaz gráfica gratuita para Git.
- El código fuente se ha alojado en **GitHub**, un servicio propiedad de Microsoft en el que los usuarios pueden crear repositorios públicos o privados. En estos repositorios se pueden alojar y mantener proyectos software
- **Math.NET Numerics** es una librería de código abierto para el framework de .net de Microsoft. Permite la realización de cálculos matemáticos de distinto tipo, desde trabajo con matrices a regresiones e interpolaciones. En este caso se ha utilizado para resolver sistemas de ecuaciones lineales.
- **Windows Presentation Foundation** es una tecnología de Microsoft para el desarrollo de interfaces gráficas para aplicaciones de escritorio en entorno Windows y aplicaciones web.
- **NUnit** es una librería de testing, utilizada para definir pruebas unitarias y de integración de aplicaciones.
- **NX y Ansys** (indirectamente), para la validación de los resultados. Ansys ha sido utilizado de forma indirecta, debido a no disponer de una licencia y a no tener los conocimientos necesarios para poder plantear y resolver un problema.

La elección de tecnologías y herramientas ha venido condicionada en gran medida por el objetivo número 2 (familiarizarse con las tecnologías empleadas en **Innoveo AG**). Esto ha sido lo que ha determinado la elección de NUnit como librería de testing, Windows Presentation Foundation como tecnología de desarrollo de interfaz, así como la elección de C# como lenguaje de desarrollo.

En cuanto al uso de Visual Studio 2019 Community Edition y Resharper, es la combinación lógica para poder desarrollar aplicaciones C#. Visual Studio es el estándar de la industria a la hora de desarrollar aplicaciones .net, siendo la Community Edition una versión gratuita para estudiantes, lo que ha permitido desarrollar la aplicación sin incurrir en costes. En cuanto a Resharper, es una extensión muy popular entre los desarrolladores de aplicaciones que usan C# como lenguaje. Resharper no dispone de una versión gratuita para estudiantes. Sin embargo, se disponía de una versión propia de la extensión.

GitHub es un servicio completamente gratuito para el alojamiento de código fuente, lo que ha evitado incurrir en costes a la hora de desarrollar este proyecto. Por otra parte, permite disponer de repositorios tanto privados (con restricciones para usuarios gratuitos) como públicos. En este proyecto se ha utilizado para mantener el repositorio como privado hasta que estaba lo suficientemente avanzado, para después hacer público el repositorio. Se puede acceder al repositorio a través de este enlace.

Para poder interactuar con GitHub, se ha utilizado Atlassian Sourcetree, también debido a ser una herramienta gratuita.

Durante la implementación se evaluó la relación coste/beneficio de implementar un sistema de resolución de sistemas de ecuaciones lineales. Debido a que no era algo novedoso, pero si tenía cierto grado de complejidad, se decidió buscar una librería que cubriera esta necesidad. Se eligió Math.Net Numerics puesto que ofrecía un bajo coste de integración al tiempo que resolvía el problema, con la garantía de calidad de un proyecto de código abierto.

Por último, se utilizaron NX y Ansys para resolver dos problemas idénticos, a fin de comparar los resultados con los de la aplicación.

Valoración económica

El desarrollo de esta aplicación ha sido un esfuerzo realizado por un estudiante, compaginado con un trabajo a tiempo completo, a lo largo de casi un año a un coste cero. Sin embargo, de haber sido desarrollada por una empresa, los costes hubieran sido muy diferentes.

En primer lugar, si en lugar de haber sido desarrollada en el tiempo libre de un desarrollador hubiera sido desarrollada en su jornada laboral, el tiempo de desarrollo hubiera sido inferior debido a que la calidad del tiempo tras una jornada laboral es inferior. En el gráfico inferior se pueden observar la cantidad de *commits*, paquetes de cambios en el código fuente, en cada semana del último año.

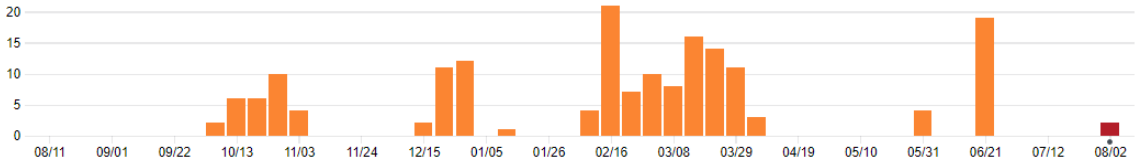

Figura 15: Histograma de commits en el último año

Dada la complejidad del código fuente se estima que el coste de desarrollo en condiciones empresariales hubiera sido de tres meses/hombre. En este tiempo se incluye el tiempo dedicado a investigación y aprendizaje, que no está reflejado en el gráfico anterior.

En cuanto a las herramientas, algunas de ellas no pueden ser empleadas por empresas con cierto tamaño o facturación. Microsoft solo permite el uso de Visual Studio Community Edition a aquellas empresas de menos de 250 equipos y menos de 1 millón de dólares de facturación. En este caso, se asumirá que la compañía tiene una facturación superior y no puede acogerse a esta versión de Visual Studio. Por ello, sería necesario comprar una licencia o subscripción de Visual Studio Professional. En este caso el precio sería de $45 al mes.

Por otra parte, ReSharper tiene un coste de subscripción de 299€ el primer año, 239€ el segundo y 179€ el tercero y siguientes. En este caso se asumiría que la empresa ya ha renovado la subscripción durante algunos años y el coste sería de 179€

En cuanto a GitHub, dispone de un plan gratuito al que cualquier empresa puede unirse. Sin embargo, dado que se asume que la compañía tiene un cierto tamaño, tiene sentido optar por el plan Enterprise, que tiene un coste de $21 al mes por usuario. El plan Enterprise tiene ventajas con los planes inferiores Free y Teams debido a que incorpora mecanismos de revisión de pares en los flujos de trabajo de los desarrolladores.

Finalmente, para la validación, esta se puede realizar disponiendo de una licencia de NX con soporte para diseño y análisis durante un mes. En cuanto al coste de las licencias necesarias, se contactó con *Alfredo Navarro*, de *Navarro y Soler CAD-PLM Software S.L.* La licencia necesaria para el trabajo, *Simcenter 3D Structures*, está disponible con un coste de adquisición de 30890€ más un mantenimiento anual asociado de 6488€. Por otra parte, el coste de la suscripción anual es de 12972€ y el de la subscripción mensual de 2471€. En este caso se

asume que la compañía se limitaría a realizar una subscripción mensual, lo que deja el coste de la licencia de NX en 2471€.

En cuanto al coste salarial, se asume que se dispone de un Ingeniero de Software con conocimientos de CAD/CAM/CAE, con un salario bruto anual de 38000€. El coste para la empresa incluye el salario bruto más un 30% extra de gastos, que desglosados corresponden a:

- 23,60% contingencias comunes
- 5,5% tipo general de desempleo para contrato indefinido
- 0,20% FOGASA (Fondo de Garantía Salarial)
- 0,70% para formación profesional

Esto deja los costes de empresa anuales en 38000€ más 11400€, lo que deja los gastos de personal anuales en un total de 49400€.

En resumen, los costes totales de tres meses de desarrollo de la aplicación ascienden a:

- Salarios: 12350€
- Licencias para la validación: 2471€
- Herramientas de desarrollo:
    - Visual Studio 2019 Professional: $135 o 114,51€ con una tasa de cambio de 0,85 USD/EURO
    - ReSharper: 59,66€
    - GitHub: $63 o 53,44 con una tasa de cambio de 0,85 USD/EURO

El coste total de desarrollo asciende a 15048,61€

Bibliografía

[1] Zienkiewicz, O. C., Taylor R. L. y Zhu J.Z. (2013). The Finite Element Method: Its Basis and Fundamentals. Oxford: Butterworth-Heinemann

[2] Caendkoelsch <https://caendkoelsch.wordpress.com/2017/12/03/how-are-stiffness-matrices-assembled-in-fem/> [Consulta: 10 de diciembre de 2019]

[3] CE IIT, Kharagpur <https://nptel.ac.in/content/storage2/courses/105105109/pdf/m4l27.pdf> [Consulta: 5 de diciembre de 2019]

[4] Christian Mosers <https://www.wpftutorial.net/IntroductionTo3D.html> [Consulta: 10 de octubre de 2019]

[5] Henri P. Gavin, Duke University <http://people.duke.edu/~hpgavin/cee421/truss-3d.pdf> [Consulta: 3 de enero de 2020]

[6] Learn about Structures <http://www.learnaboutstructures.com/Stiffness-Method-for-One-Dimensional-Truss-Elements> [Consulta: 1 de diciembre de 2019]

[7] Mahdi Farahikia, "Introduction to finite element method (8-week course)" <https://www.youtube.com/watch?v=Y71jil9qiYY&list=PLQVMpQ7G7XvHrdHLJgH8SeZQsiy2lQUcV&index=1> [Consulta: 3 de diciembre de 2019]

[8] Purdue University <https://engineering.purdue.edu/~aprakas/CE474/CE474-Ch5-StiffnessMethod.pdf> [Consulta: 20 de diciembre de 2019]

[9]Rensselaer Polytechnic University <http://homepages.rpi.edu/~des/Trusses.pdf> [Consulta: 3 de enero de 2020]

[10] Unidad 8: Método de los elementos finitos < https://poliformat.upv.es/, Aplicaciones CAE 2018-2019> [Consulta: 10 de octubre de 2019] Unidad 9: Interpolación en elementos finitos < https://poliformat.upv.es/, Aplicaciones CAE 2018-2019> [Consulta: 10 de octubre de 2019]

[11] Universiti Malaysia PAHANG <http://ocw.ump.edu.my/course/view.php?id=47> [Consulta: 3 de enero de 2020]

[12] University of Victoria <https://www.engr.uvic.ca/~mech410/lectures/FEA_Theory.pdf> [Consulta: 8 de enero de 2020]

[13] What-when-how < http://what-when-how.com/the-finite-element-method/fem-for-two-dimensional-solids-finite-element-method-part-1/> [Consulta: 10 de enero de 2020]

[14] Microsoft <https://docs.microsoft.com/es-es/dotnet/api/system.windows.controls.canvas?view=netframework-4.8> [Consulta: 10 de agosto de 2020]

[15] WPF Tutorial <https://www.wpf-tutorial.com/es/24/paneles/el-control-canvas/> [Consulta 10 de agosto de 2020]

[16] Microsoft < https://docs.microsoft.com/es-es/dotnet/framework/wpf/graphics-multimedia/how-to-create-a-3-d-scene> [Consulta: 10 de agosto de 2020]

[17] Alfredo Navarro <https://www.linkedin.com/in/alfredo-n-8783a0a4/> [Consulta 11 de agosto de 2020]

Anexo A: Obteniendo de la aplicación

La aplicación y el código fuente de esta se pueden encontrar en su repositorio de GitHub (https://github.com/Mr-Alicates/Master_FEM). En esta dirección se puede consultar el código fuente y descargarlo.

En caso de querer obtener una versión usable de la aplicación, se puede compilar manualmente utilizando las herramientas anteriormente descritas o bien descargando una de las seis versiones ejecutables disponibles en el apartado de *Releases* (https://github.com/Mr-Alicates/Master_FEM/releases).

La versión más reciente y que representa el trabajo de este TFM es la versión *Alpha-008*, disponible en este enlace (https://github.com/Mr-Alicates/Master_FEM/releases/download/Alpha-008/Alpha-008.zip).

La aplicación está contenida en un fichero zip, por lo que utilizar una aplicación de descompresión puede ser necesario. Dentro de este fichero existen dos carpetas:

- **Debug,** que contiene la aplicación con símbolos de depuración. Esta versión solo es necesaria en caso de que sea necesario depurar algún error en la misma.
- **Release**, que contiene la aplicación lista para ser usada o evaluada por un usuario. Esta es la aplicación que debe ser usada para evaluar este TFM.

El archivo ejecutable se llama **POC3D.exe**. La pantalla inicial se muestra en la siguiente figura:

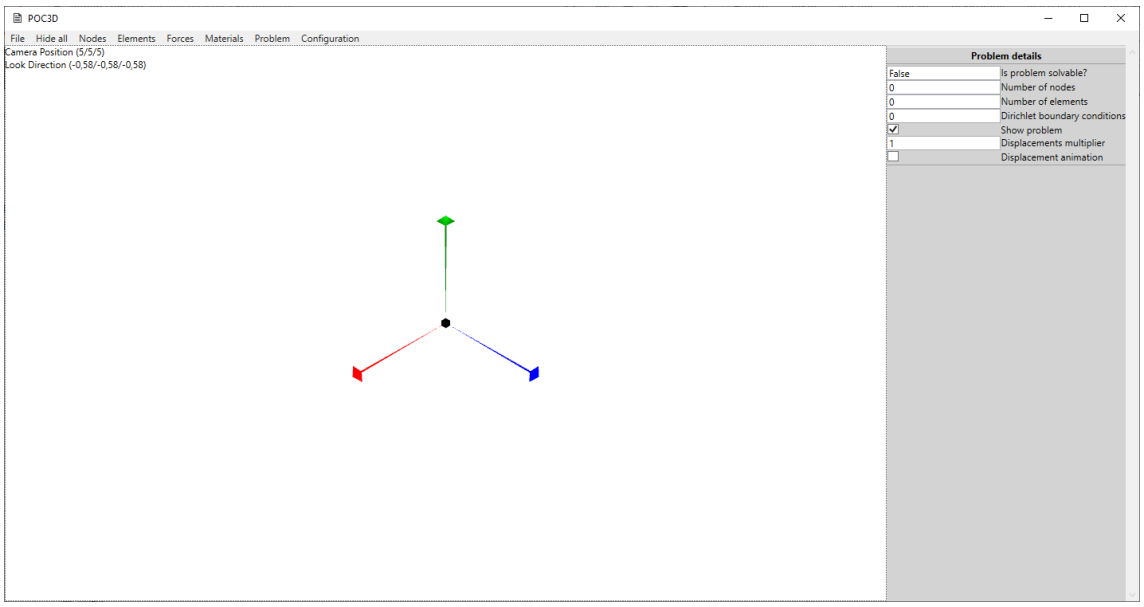

Figura 16: Ventana inicial de la aplicación

La aplicación ha sido probada en Windows 10. En sistemas Linux con *Wine* no se ha conseguido ejecutarla. No se ha probado en sistemas Mac debido a la falta de un equipo en el que realizar la comprobación,

Anexo B: Manual de uso de la aplicación

La ventana de la aplicación dispone de una serie de menús y áreas especializadas que permiten dar información o interactuar con el usuario:

1. La barra superior contiene los distintos menús desplegables que permiten interactuar con la aplicación.
2. Los indicadores permiten al usuario saber cuál es la posición de la cámara (**Camera position**) y el vector que describe la dirección en la que mira la cámara (**Look Direction**).
3. El área de trabajo muestra los componentes 3D del problema. En la figura se muestra el origen de coordenadas y los ejes.
4. El área de menús muestra los detalles del problema y de los distintos componentes de la aplicación

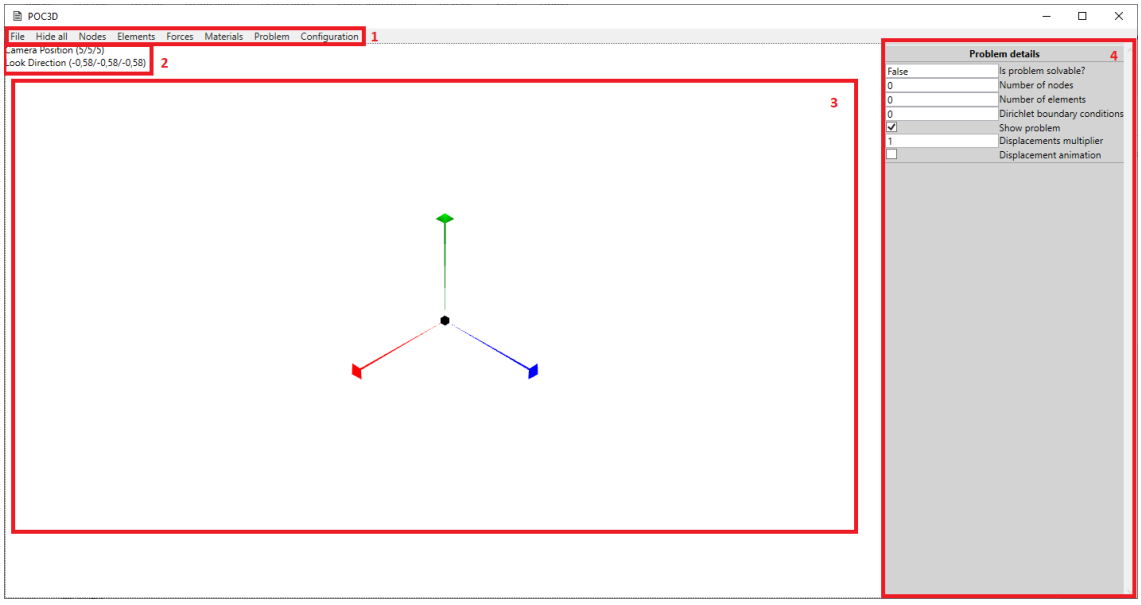

Figura 17 : Áreas de la aplicación

El área de trabajo

El área de trabajo permite observar una representación en tres dimensiones del problema que se pretende resolver.

Los nodos del problema se representan mediante cubos y pirámides de color verde. Los cubos representan nodos libres, mientras que las pirámides representan nodos que tienen alguno de sus ejes fijos.

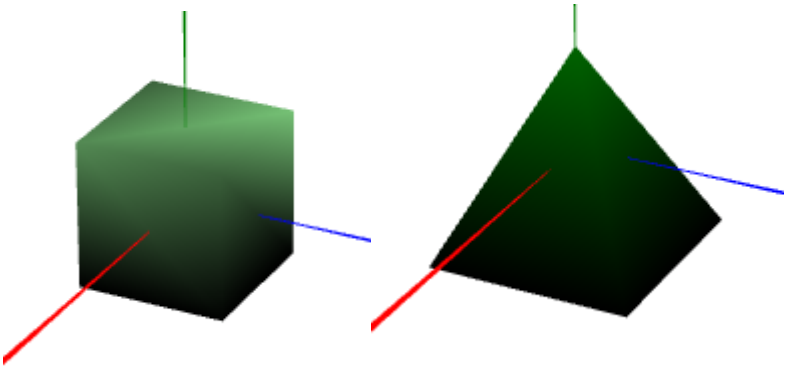

Figura 18 : Nodos libres y restringidos

Los elementos del problema se representan como barras de sección cuadrada de color azul.

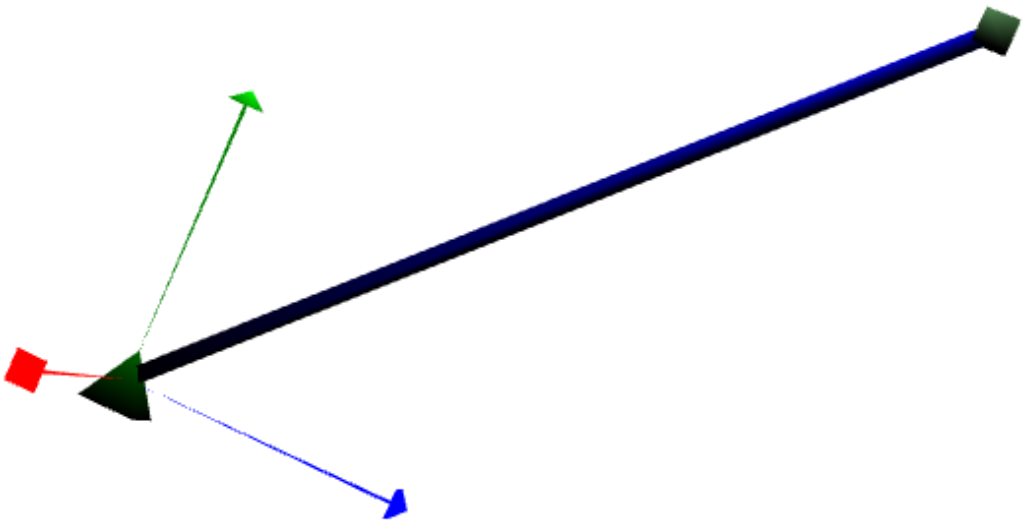

Figura 19 : Elemento tipo barra

Por otra parte, las fuerzas del problema se representan mediante flechas de color amarillo. Estas cambian su orientación en función del vector de la fuerza.

Figura 20: Fuerza

Adicionalmente, cuando un elemento sea seleccionado su color cambiará a rojo.

Controles de la cámara

La cámara se puede controlar mediante el uso del ratón:

- Manteniendo pulsado el botón izquierdo del ratón y moviéndolo se puede realizar un movimiento de **pan**, en el que se mueve la cámara sin girarla por el plano de la pantalla.
- Manteniendo pulsada la rueda del ratón y moviéndolo se puede realizar un movimiento de **giro** de la cámara, mientras se mantiene fija su posición. Es un movimiento análogo al que permite realizar el ratón en videojuegos en primera persona.
- Manteniendo pulsado el botón derecho y moviéndolo se puede realizar un movimiento de **orbita** de la cámara alrededor del origen de coordenadas.
- La rueda del ratón permite realizar un movimiento de **zoom**, que mueve la cámara hacia delante o detrás en la dirección en la que esté mirando.

La cámara también se puede mover mediante el uso del teclado:

- Las teclas **W** y **S** realizan un movimiento similar al de **zoom**, moviendo la cámara hacia delante o detrás en la dirección en la que esté mirando.
- Las teclas **A** y **D** mueven la cámara a izquierda y derecha.
- Las teclas **R** y **F** hacen ascender o descender la cámara

La cámara también se puede rotar a lo largo de los tres ejes mediante el uso del teclado:

- Manteniendo pulsada la tecla **Mayúsculas izquierda** y pulsando las teclas **W** y **S** se realiza un movimiento de rotación en el eje transversal o **cabeceo**.
- Manteniendo pulsada la tecla **Mayúsculas izquierda** y pulsando las teclas **A** y **D** se realiza un movimiento de rotación en el eje vertical o **guiñada**.
- Manteniendo pulsada la tecla **Mayúsculas izquierda** y pulsando las teclas **Q** y E se realiza un movimiento de rotación en el eje longitudinal o **alabeo**.

Por último, también se pueden seleccionar nodos, elementos o fuerzas utilizando el botón izquierdo del ratón, simplemente haciendo clic en los respectivos objetos 3D que los representan. Como se ha indicado en la sección anterior, esto hace que el color de dichos elementos cambie a rojo, además de hacer que se muestre el menú de detalle correspondiente al tipo de elemento seleccionado.

Menús desplegables

El primero de los menús es el de **File**.

- La primera opción borra el contenido del problema actual y genera un problema nuevo
- La segunda opción abre un selector de ficheros y permite cargar un problema desde un archivo con extensión **3dPoc**.
- La tercera opción abre un selector de ficheros y permite guardar un problema a un fichero con extensión **3dPoc**.

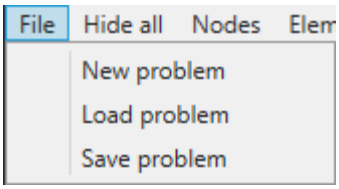

Figura 21: Menú Archivo

El siguiente menú es **Hide all**. No dispone de opciones, sino que se encarga de ocultar los distintos menús que pueda haber en el área de menús.

El menú **Nodes** se encarga de mostrar los distintos menús de detalles relacionados con los nodos del problema:

- **Show selected Node details** se encarga de mostrar el menú de detalles del nodo seleccionado en caso de que haya un nodo seleccionado y el menú se haya ocultado.
- **Add new Node** se encarga de agregar un nuevo nodo al problema. Por defecto, este aparece en el origen de coordenadas.
- **Show list of Nodes** se encarga de mostrar un listado de nodos en el problema, así como el menú de detalles del nodo seleccionado.

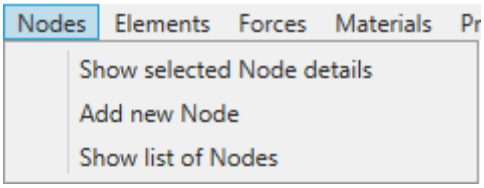

Figura 22: Menú Nodos

El menú **Elements** se encarga de mostrar menús relacionados con los elementos:

- **Show selected Element details** se encarga de mostrar el menú de detalles del elemento seleccionado en caso de que haya un elemento seleccionado y el menú se haya ocultado.

- **Add new Element** Se encarga de mostrar el menú de detalle que se encarga de agregar nuevos elementos
- **Show list of Elements** se encarga de mostrar un listado de elementos en el problema, así como el menú de detalles del elemento seleccionado.
- **Show matrices from element** abre la ventana de detalles del elemento

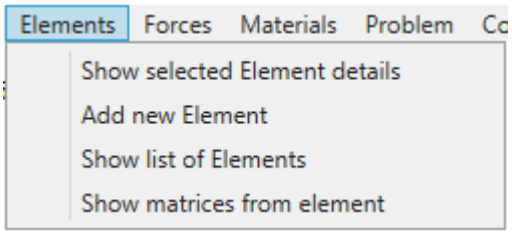

Figura 23: Menú Elementos

El menú **Forces** se encarga de mostrar menús relacionados con las fuerzas:

- **Show selected Force details** se encarga de mostrar el menú de detalles de la fuerza seleccionada en caso de que haya una fuerza seleccionada y el menú se haya ocultado.
- **Add new Force** se encarga de abrir el menú de detalles que permite añadir una fuerza al problema.
- **Show list of Forces** se encarga de mostrar un listado de fuerzas en el problema, así como el menú de detalles de la fuerza seleccionada.

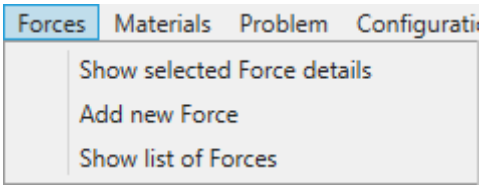

Figura 24: Menú Fuerzas

El menú **Materials** se encarga de mostrar menús relacionados con los materiales:

- **Add new Material** agrega un nuevo material al problema.
- **Show list of Materials** se encarga de mostrar un listado de materiales en el problema, así como un menú de detalles del material seleccionado.

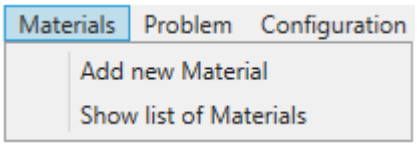

Figura 25: Menú Materiales

El menú **Problem** solo dispone del submenú **Show matrices from problem**, que se encarga de abrir la ventana que muestra las matrices globales del problema y sus detalles.

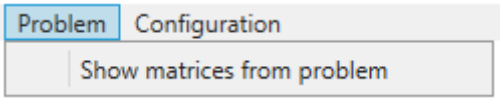

Figura 26: Menú Problema

Finalmente, el menú **Configuration** dispone del submenú **Show configuration**, que se encarga de mostrar un menú de detalle con opciones de depuración para poder cambiar el comportamiento de ciertos controles de la cámara. Estas opciones son una herramienta de desarrollo que no se ha retirado del programa final, por lo que su uso no está recomendado.

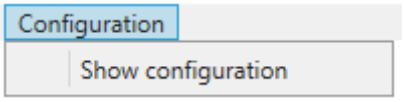

Figura 27: Menú Configuración

Menús de detalles

Menú de detalles del problema

El primer menú es el de detalles del problema y siempre está visible, independientemente de otros menús que puedan aparecer encima de el:

- **Is problema solvable?** indica si el problema tiene solución.
- **Number of nodes** indica el número de nodos en el problema
- **Number of elements** indica el número de elementos en el problema
- **Dirichelet boundary conditions** indica el número de grados de libertad que han sido restringidos.
- **Show problem** permite seleccionar si se muestra el problema o la solución. Cuando está seleccionado muestra el problema. En caso contrario muestra la solución.
- **Displacements multiplier** permite multiplicar o exagerar los resultados de deformación del problema. Solo aplica cuando el checkbox “Show problem” está seleccionado.
- **Displacement animation** indica al programa que realice una pequeña animación que representa la deformación. Realiza esto al hacer variar el valor de “Displacements multiplier” desde 0 al valor actual de “Displacements multiplier” en incrementos de una décima parte a lo largo de varios segundos.

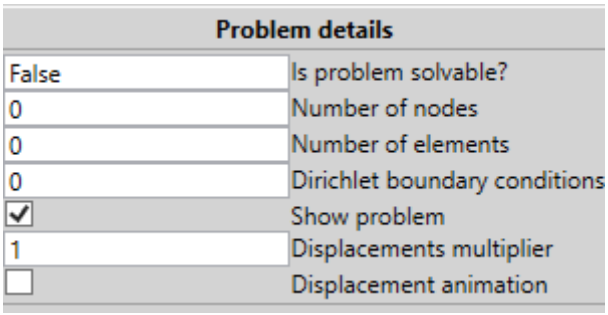

Figura 28: Menú de detalles del problema

Menú de detalles del nodo

El menú de detalles del nodo se muestra cuando hay un nodo seleccionado o cuando se muestra el listado de nodos del problema. Permite cambiar los detalles del nodo que esté actualmente seleccionado:

- **X coordinate**, **Y coordinate**, **Z coordinate** permiten cambiar la ubicación espacial del nodo
- Los checkboxes **Is X axis fixed**, **Is Y axis fixed**, **Is Z axis fixed** permiten fijar un eje de movimiento del nodo. Cuando están seleccionados fijan el eje, dejándolo libre en caso contrario.

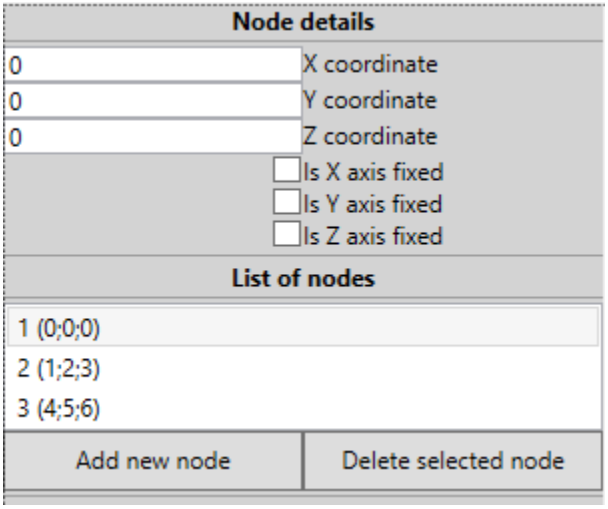

Figura 29: Menú de detalles del nodo y listado de nodos

Menú de listado de nodos

El menú de listado de nodos se encarga de enumerar los distintos nodos del problema e identificarlos mediante sus coordenadas. Los usuarios pueden usar el listado para seleccionar nodos que estén fuera de la vista y editar sus propiedades mediante el menú de detalles del nodo.

El botón **Add a new node** crea un nuevo nodo en el origen de coordenadas, del mismo modo que le menú desplegable del mismo nombre.

El botón **Delete selected node** elimina el nodo que esté actualmente seleccionado, siempre y cuando pueda ser eliminado. Un nodo que forme parte de un elemento o tenga una fuerza aplicada sobre él no puede ser eliminado.

Menú de nuevo elemento

El menú de nuevo elemento permite seleccionar los dos nodos necesarios para crear un nuevo elemento. **Origin node** permite seleccionar el nodo de origen mientras que **Destination node** permite seleccionar el nodo destino. El control no permite al usuario seleccionar el mismo nodo como origen y destino o crear un elemento sin alguno de los nodos necesarios.

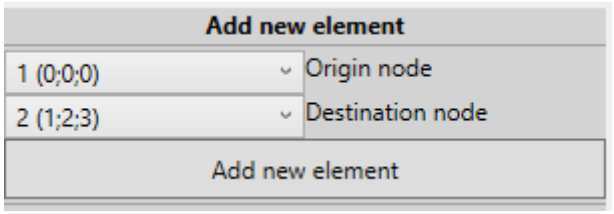

Figura 30: Menú de nuevo elemento

Menú de detalles del elemento

El menú de detalles del elemento se muestra cuando hay un elemento seleccionado o cuando se muestra el listado de elementos del problema. Permite cambiar los detalles del elemento que esté actualmente seleccionado:

- **Id** es un campo de solo lectura que indica al usuario el identificador único del elemento. Este valor permite al usuario identificar los distintos elementos dentro de la información global del problema
- **Description** es un campo de solo lectura. Muestra un identificador que permite al usuario saber entre que nodos está ubicado el elemento. Este identificador también se utiliza en la matriz de correspondencia.
- **Origin node** y **Destination node** permiten ver y cambiar los nodos entre los que está definido el elemento. Las mismas restricciones que en el menú de nuevo elemento aplican aquí.
- **Material** permite ver y cambiar el material que define las propiedades del elemento. Las propiedades de este se muestran en **Material Name y Material Young's Modulus**.
- **Cross section area** permite ver y cambiar la sección del elemento.
- **Length** indica la longitud del elemento.
- **K constant** indica la constante K que tiene el elemento en función de su longitud, su material y su sección transversal.

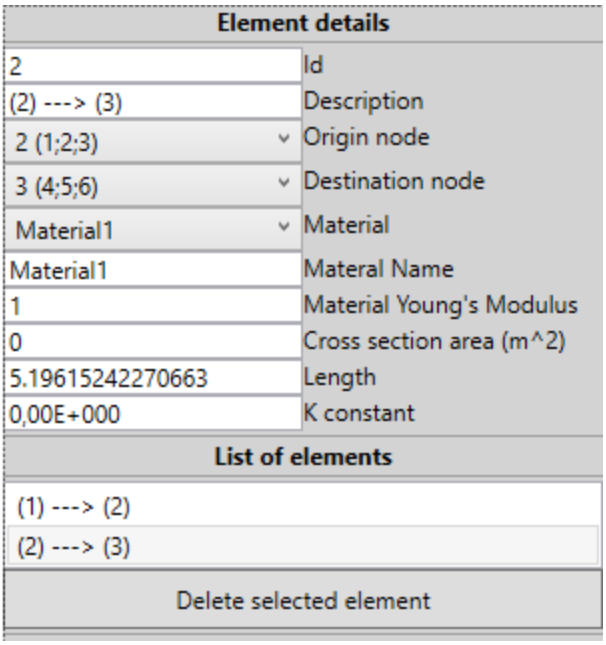

Figura 31: Menú de detalles del elemento y listado de elementos

Menú de listado de elementos

El menú de listado de elementos se encarga de enumerar los distintos elementos del problema e identificarlos mediante su descripción. Los usuarios pueden usar el listado para seleccionar elementos que estén fuera de la vista y editar sus propiedades mediante el menú de detalles del elemento.

El botón **Delete selected element** elimina el elemento que esté actualmente seleccionado.

Menú de nueva fuerza

El menú de nueva fuerza permite seleccionar un nodo en el que se va a crear una nueva fuerza a través del campo **Application node**. El nodo en el que se aplique la fuerza puede ser cualquiera sobre el que no se esté aplicando otra fuerza. Esto es debido a que la aplicación no permite la existencia de múltiples fuerzas en el mismo nodo, por una cuestión de simplicidad.

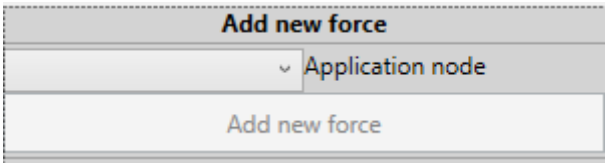

Figura 32: Menú de nueva fuerza

Menú de detalles de la fuerza

El menú de detalles de la fuerza se muestra cuando hay una fuerza seleccionada o cuando se muestra el listado de fuerzas del problema. Permite cambiar los detalles de la fuerza que esté actualmente seleccionada.

- **Application node** indica el nodo sobre el que la fuerza está siendo aplicada. Además, también permite cambiar el nodo a cualquier otro que no esté siendo usado por otra fuerza.
- **Magnitude** indica el módulo del vector de fuerza
- **Vector X coordinate**, **Vector Y coordinate**, **Vector Z coordinate** representan las componentes del vector de fuerza.

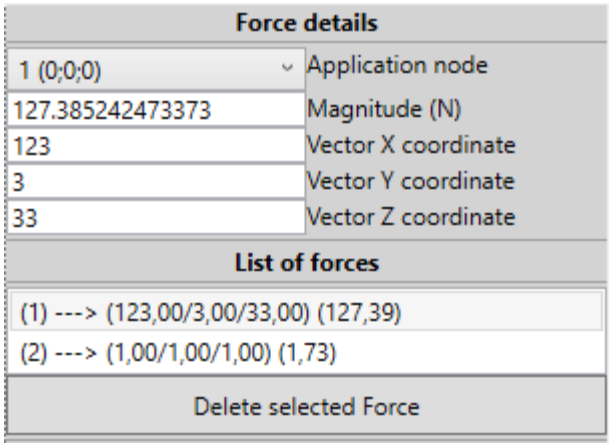

Figura 33: Menú de detalles de la fuerza y listado de fuerzas

Menú de listado de fuerzas

El menú de listado de fuerzas se encarga de enumerar las distintas fuerzas del problema e identificarlas mediante una descripción corta. La descripción indica el nodo de aplicación, las componentes del vector de fuerza y su módulo. Los usuarios pueden usar el listado para seleccionar fuerzas que estén fuera de la vista y editar sus propiedades mediante el menú de detalles de la fuerza.

El botón **Delete selected force** elimina la fuerza que esté actualmente seleccionada.

Menú de detalles del material

El menú de detalles del material se muestra cuando se muestra el listado de fuerzas del problema. Permite cambiar los detalles del material que esté actualmente seleccionado.

- **Material name** indica el nombre del material y permite editarlo.
- **Material Young's Modulus** muestra el módulo de Young del material y permite editarlo.

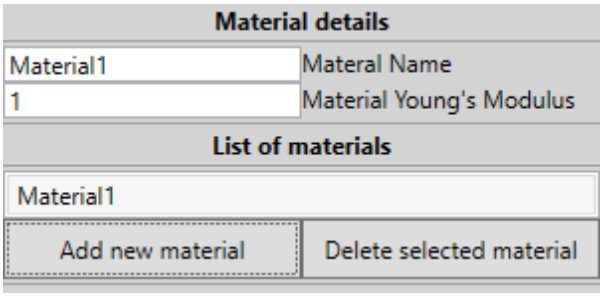

Figura 34: Menú de detalles del materia y listado de materiales

Menú de listado de materiales

El menú de listado de materiales se encarga de enumerar los distintos materiales del problema. Dado que los materiales no se representan en el área de trabajo, este control es el único capaz de seleccionar los materiales.

El botón **Add a new material** crea un nuevo material

El botón **Delete selected material** el material actualmente seleccionado, siempre y cuando no esté siendo utilizado en algún elemento.

Menú de configuración

El menú de configuración no guarda relación directa con el problema y no es necesario usarlo para poder resolver un problema. Únicamente contiene ciertas opciones gráficas para poder facilitar el uso de la aplicación por parte del usuario. Las opciones aquí representadas no se guardan al cerrar la aplicación:

- **Graphics objects size multiplier** es un multiplicador que cambia el tamaño de todos los objetos gráficos, como los cubos, pirámides, flechas y barras que representan los nodos, elementos y fuerzas.
- **Mouse rotation delta** es un multiplicador que regula cuanto gira la cámara al realizar un movimiento de rotación con el ratón.
- **Mouse pan delta** es un multiplicador que regula cuanto se mueve la cámara al realizar un movimiento de pan con el ratón.
- **Mouse Wheel delta** es un multiplicador que regula cuanto se mueve la cámara al realizar un movimiento de zoom con la rueda del ratón.
- **Keyboard rotation delta** es un multiplicador que regula cuanto gira la cámara al realizar un movimiento de giro con el teclado.
- **Mouse Wheel sensitivity** es un divisor que regula cuantas veces se mueve la cámara durante el movimiento de zoom con la rueda del ratón. Cuando el valor de este parámetro es 1, la cámara realiza 120 movimientos de avance, por lo que su valor no debería superar 120.

En cualquier caso, manipular estas opciones puede producir efectos inesperados en la aplicación.

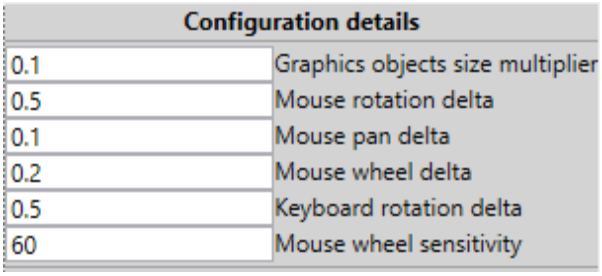

Figura 35: Menú de configuración

Ventana de información del elemento

La ventana de información del elemento es una ventana auxiliar que muestra información del elemento seleccionado. Está compuesta por distintas pestañas que muestran los detalles del proceso para pasar de la matriz del elemento en su sistema local de coordenadas hasta llegar a la matriz del elemento en el sistema global de coordenadas del problema.

Esta ventana es independiente de la ventana principal, pudiendo cerrarse si no es necesaria. También reacciona al elemento seleccionado, por lo que siempre muestra la información del elemento actualmente seleccionado o bien un esqueleto vacío en caso de que el objeto seleccionado no sea un elemento. La ventana se divide en cuatro pestañas con información teórica y práctica del proceso.

Pestaña de matriz local

La pestaña **Local Matrix** muestra al usuario el planteamiento de la matriz local de rigidez del elemento. También se muestra el valor K del elemento seleccionado.

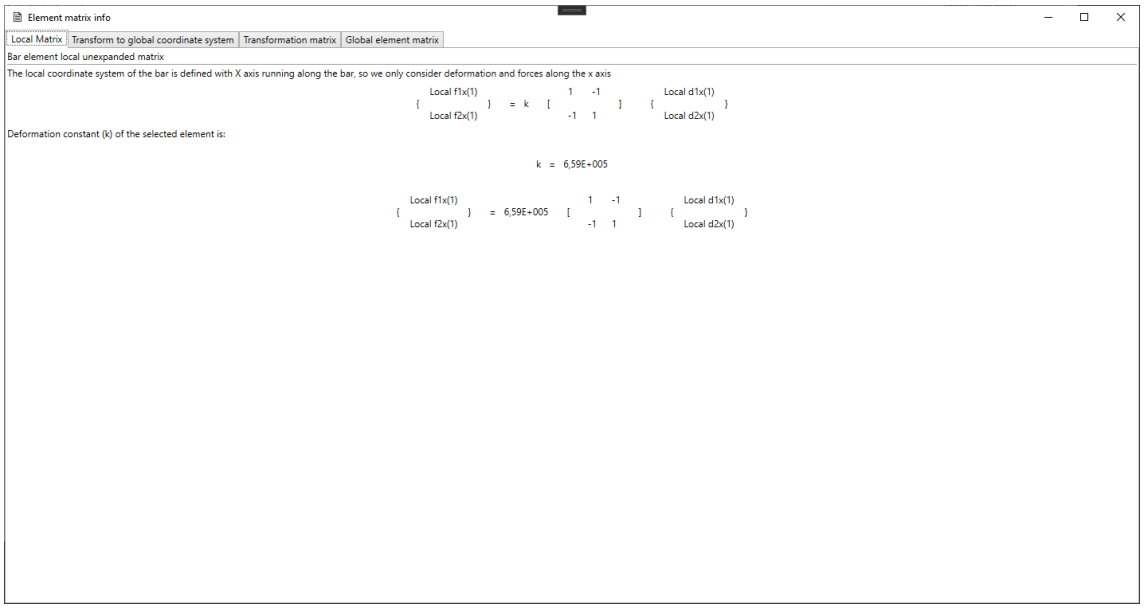

Figura 36: Pestaña de matriz local

Pestaña de transformación al sistema global de coordenadas

La pestaña **Transform to global coordinate system** muestra el desarrollo teórico necesario para pasar de la matriz local del elemento a la matriz global del elemento.

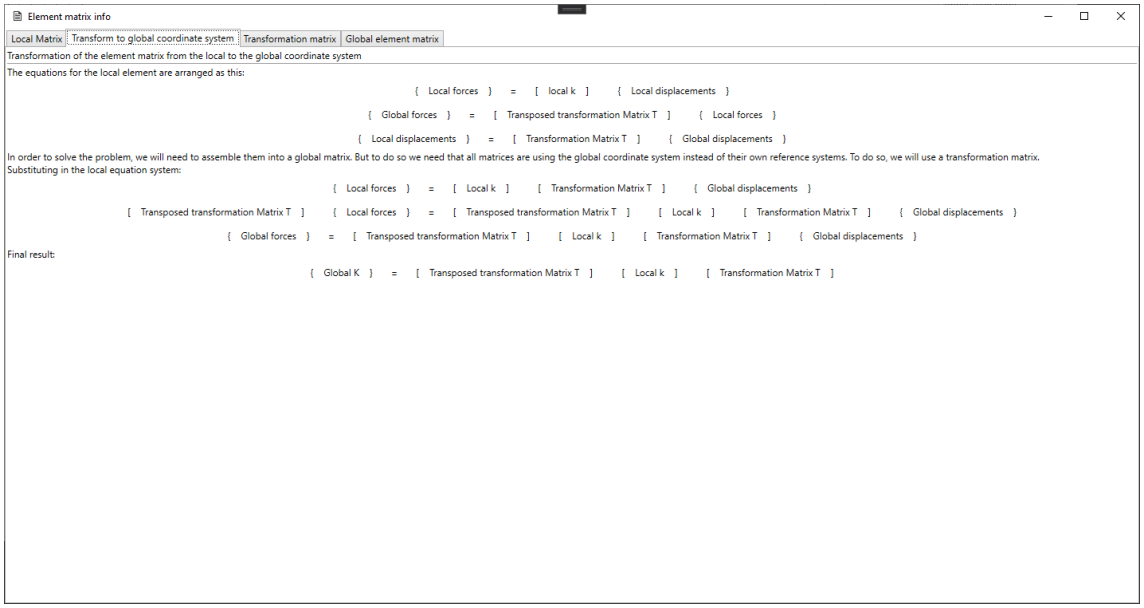

Figura 37: Pestaña de transformación al sistema global de coordenadas

Pestaña de matriz de transformación

La pestaña **Transformation matrix** muestra la base teórica y el desarrollo necesario para calcular la matriz de transformación usada para pasar de la matriz local del elemento a la matriz global del elemento.

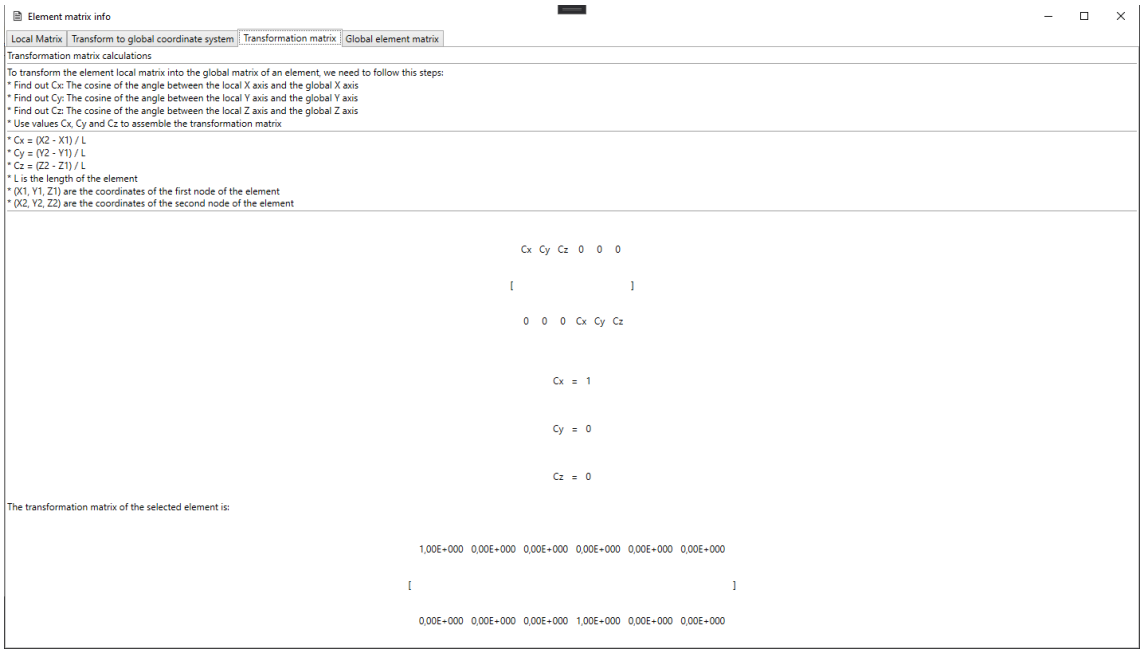

Figura 38: Pestaña de matriz de transformación

Pestaña de matriz global del elemento

En la pestaña **Global element matrix** se muestra el planteamiento de los cálculos necesarios para calcular la matriz global del elemento. Además, se muestra la matriz resultante.

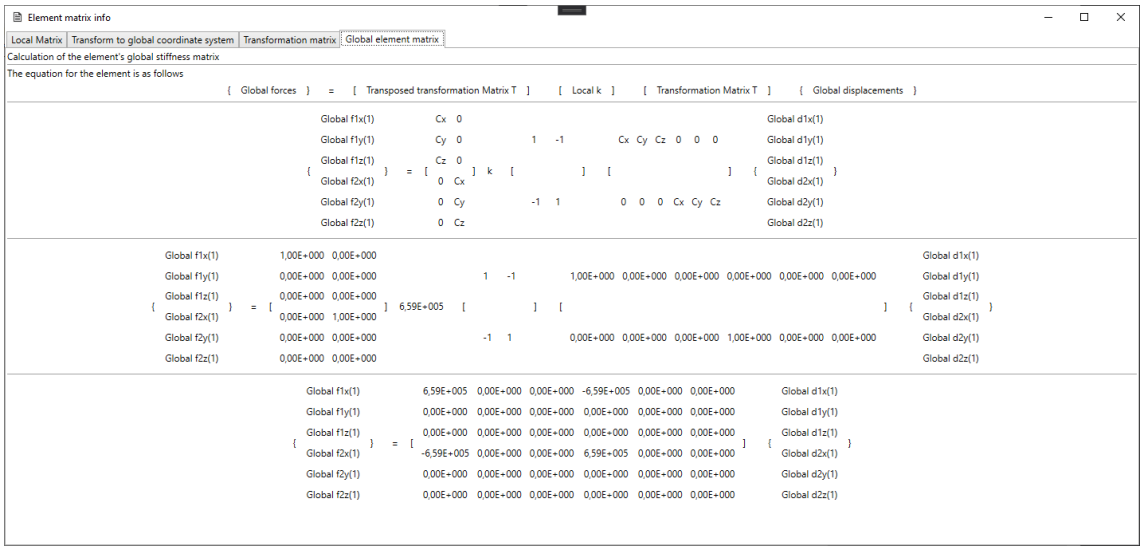

Figura 39: Pestaña de matriz global del elemento

Ventana de información del problema

La ventana de información del problema es una ventana auxiliar que muestra información sobre las fases finales de cálculo del problema. Está compuesta por distintas pestañas que muestran todo el proceso de ensamblaje y preparación de la matriz global de rigidez. También muestra el sistema de ecuaciones lineales, las deformaciones resultantes y el cálculo de las fuerzas de reacción.

Pestaña de matriz de correspondencia

La pestaña **Correspondence Matrix** comienza mostrando una pequeña explicación acerca de la representación que se hace de la matriz de correspondencia.

La ventana muestra una tabla en la que los nodos del problema están situados en las columnas y en las filas. La tabla indica que elementos están en contacto con que nodos, información que permite ensamblar la matriz global de rigidez.

Como detalle adicional, los nodos en los que existe alguna restricción están marcados en color rojo, mientras aquellos que están completamente libres están marcados en color verde. Esta funcionalidad es un remanente de la versión inicial de la aplicación, en la que no se podían restringir los grados de libertad de los nodos de forma individual

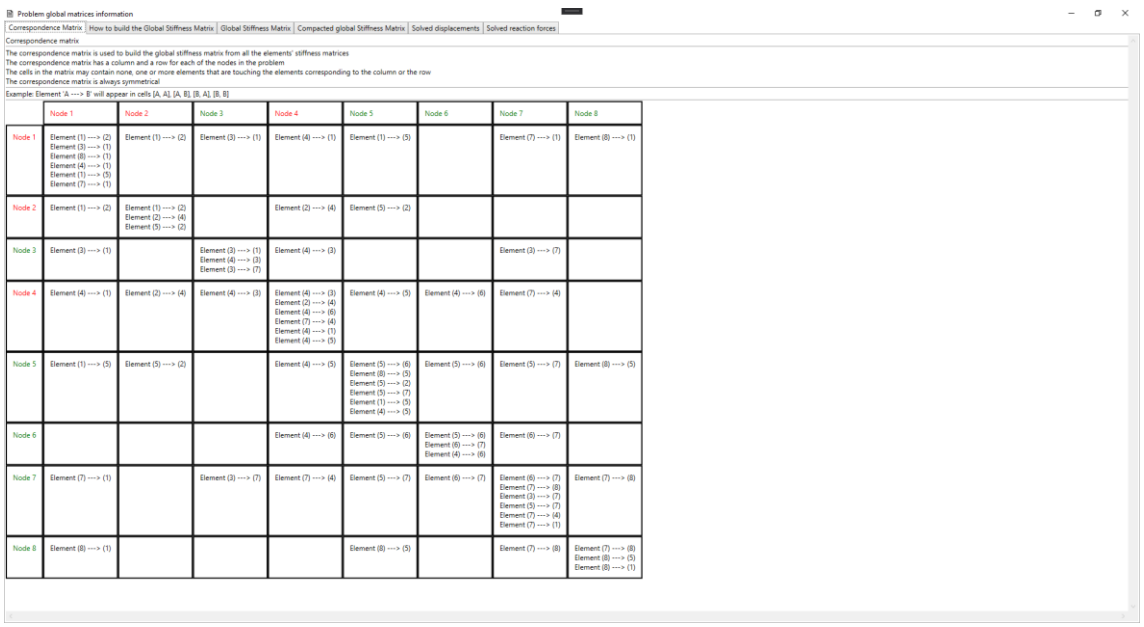

Figura 40: Pestaña de matriz de correspondencia

Pestaña de información de ensamblaje

La pestaña **How to build the Global Stiffness Matrix** contiene información teórica acerca de cómo construir la matriz global de rigidez. Detalla el procedimiento de como ensamblar la matriz global a partir de las matrices de los elementos y de la matriz de correspondencia. También ofrece un ejemplo visual de cómo realizar el ensamblaje.

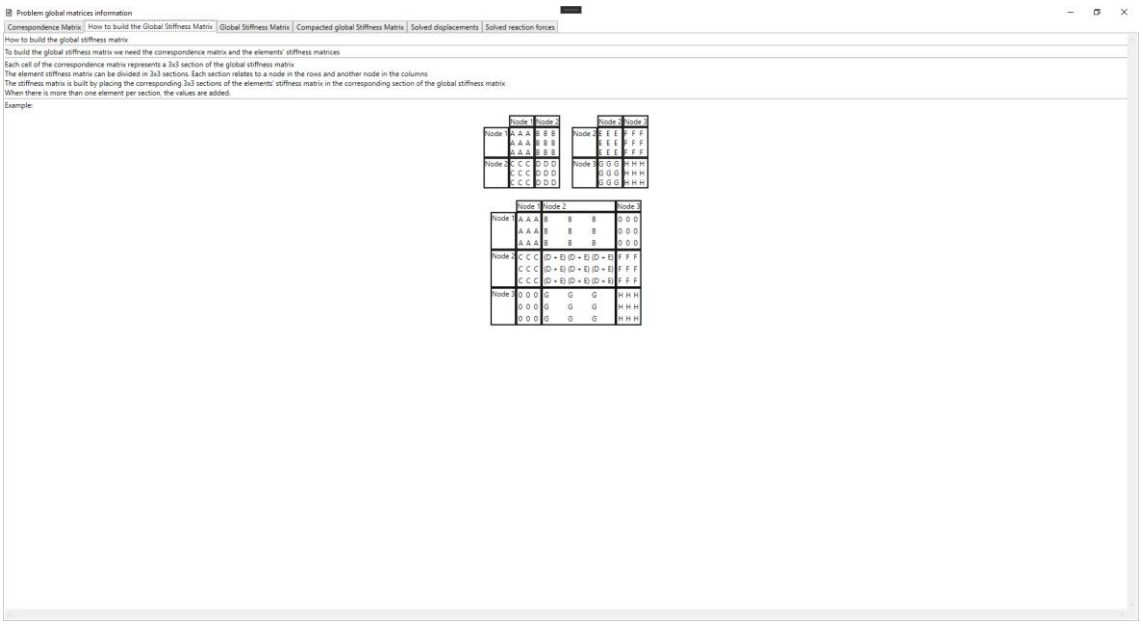

Figura 41: Pestaña de información de ensamblaje

Pestaña de la matriz global

La pestaña **Global Stiffness Matrix** muestra la matriz global de rigidez, completamente ensamblada. Las columnas y las filas de la tabla están identificadas con los nombres de los distintos grados de libertad de cada uno de los nodos.

Los grados de libertad están coloreados en rojo para aquellos que están restringidos, estando coloreados en verde en caso contrario. También se muestra el vector de desplazamientos y el vector de fuerzas.

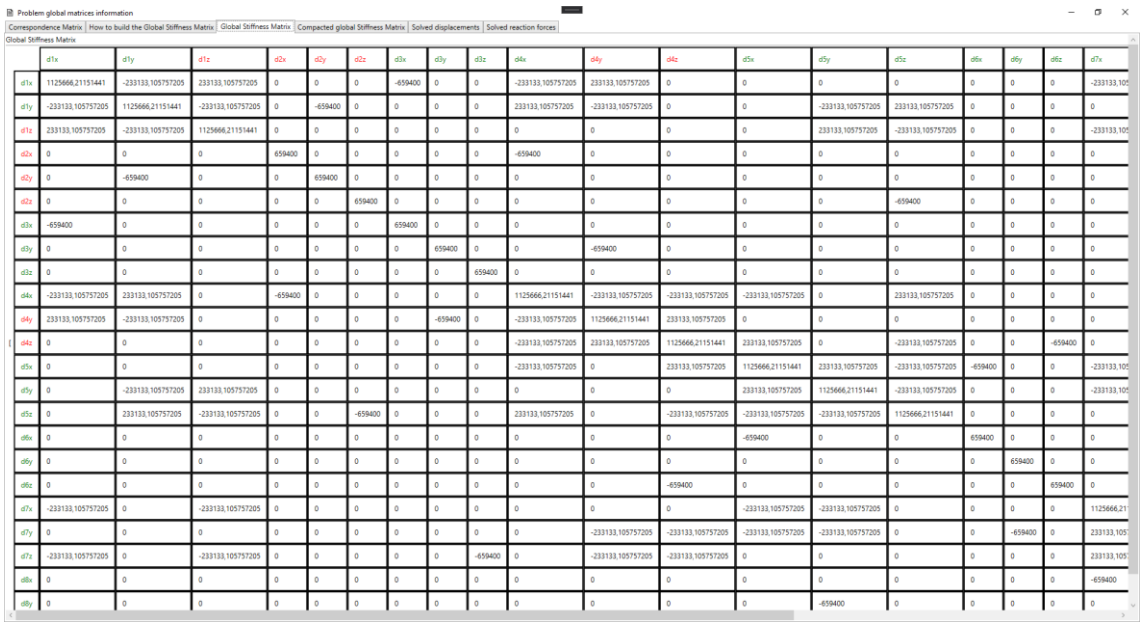

Figura 42: Pestaña de la matriz de rigidez

Pestaña de matriz global compactada

En la pestaña **Compacted global Stiffness Matrix** se muestra la versión compactada de la matriz de rigidez, a la que se han eliminado las filas y columnas correspondientes a los grados de libertad que han sido restringidos, junto con el vector de desplazamientos compactado y el vector de fuerzas compactadas.

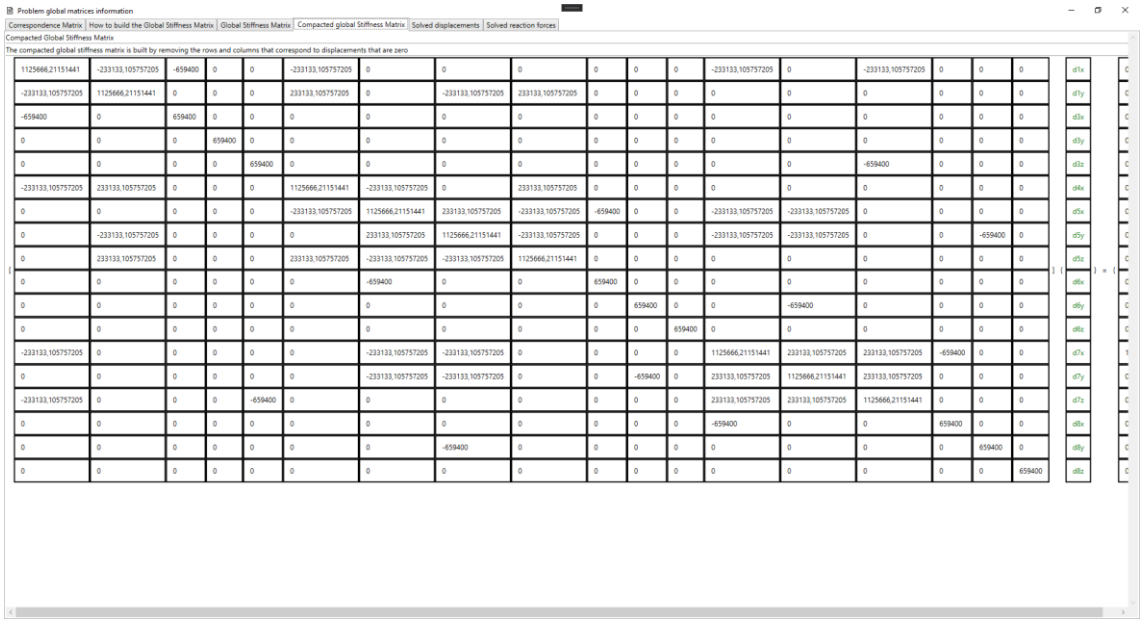

Figura 43: Pestaña de matriz global compactada

Pestaña de resultados de desplazamiento

La pestaña **Solved displacements** muestra los resultados de resolver el sistema de ecuaciones lineales que se muestra en la pestaña de la matriz global compactada.

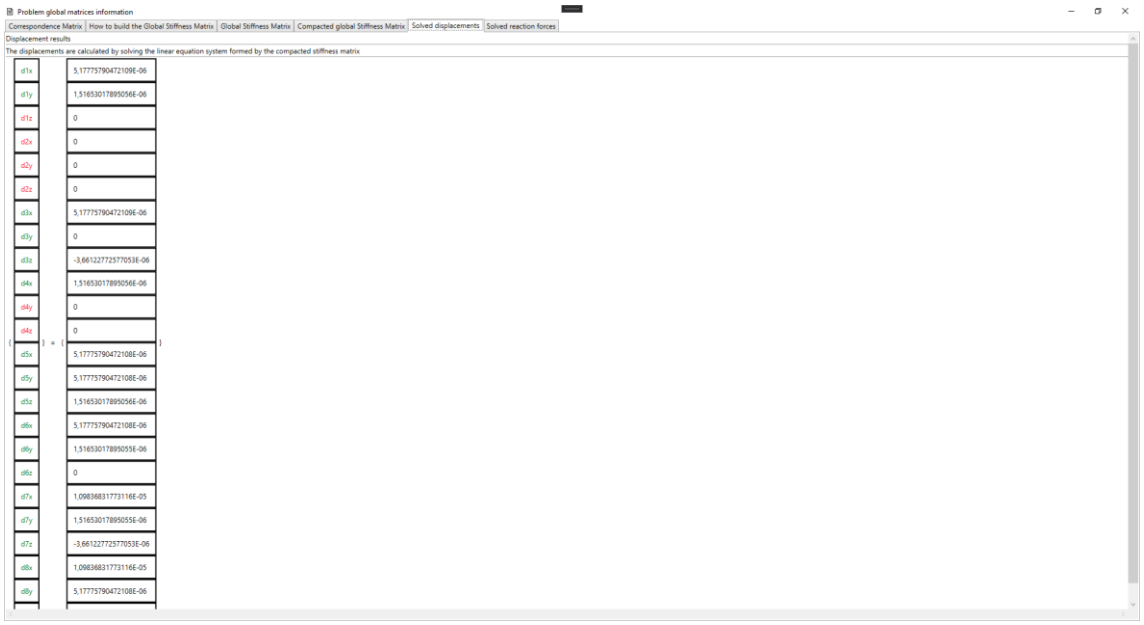

Figura 44: Pestaña de resultados de desplazamiento

Pestaña de fuerzas resultantes

La pestaña de fuerzas resultantes muestra el resultado de aplicar los resultados de desplazamiento a la matriz global compactada.

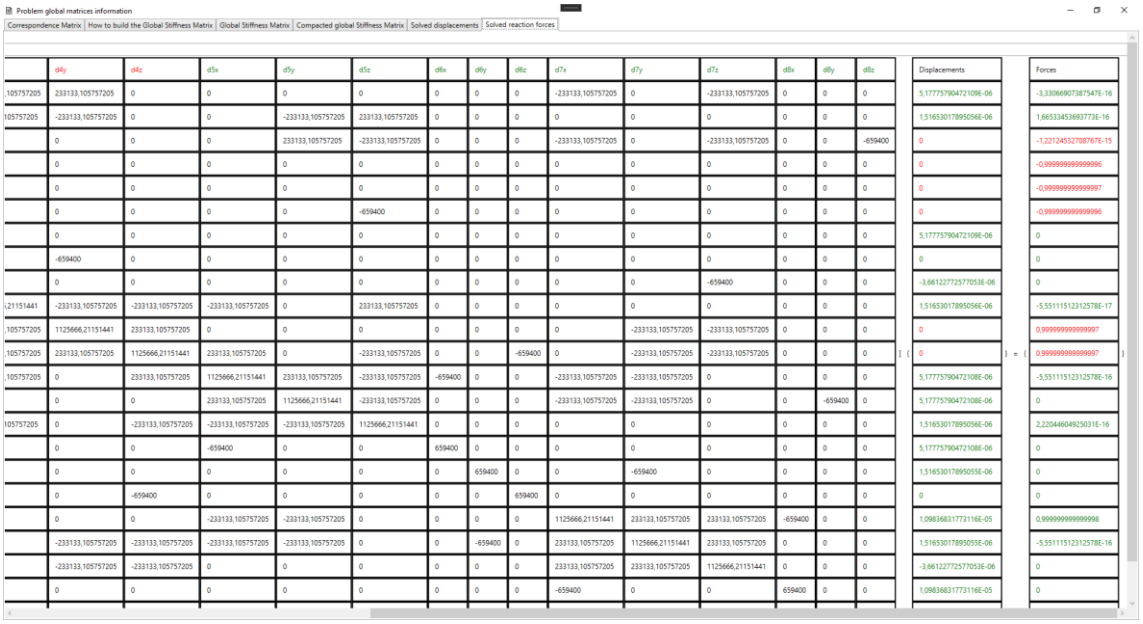

Figura 45: Pestaña de fuerzas resultantes