Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

Desarrollo de una aplicación en Python para el pandeo global de pórticos planos

Autor:

Martín Cabeza, Jose Manuel

Tutores:

Lorenzana Ibán, Antolín

Construcciones Arquitectónicas, Ingeniería del Terreno y Mecánica de los Medios Continuos y Teoría de Estructuras

Magdaleno González, Álvaro

Construcciones Arquitectónicas, Ingeniería del Terreno y Mecánica de los Medios Continuos y Teoría de Estructuras

Valladolid, Julio de 2021

Resumen

Python es un lenguaje de programación de alto nivel, sencillo de interpretar y de código libre. En este trabajo, se ha realizado una adaptación de un programa ya existente en código Python que realizaba cálculos de casos estáticos en 2D mediante el método directo de rigidez (semejante en ciertos aspectos al método de los elementos finitos). Se ha obtenido un programa que, aprovechando el cálculo estático del programa mencionado anteriormente, realiza el cálculo de pandeo global de la estructura 2D mediante una aproximación matricial y agrupa todo en un único código.

Palabras clave

Pandeo global, Cálculo estructural, Python, SAP2000, Método Directo de Rigidez.

Abstract

Python is a high-level, easy-to-interpret and open-source programming language. In this work, an adaptation of an existing program has been carried out in Python code that performed calculations of static cases in 2D using the direct stiffness method (similar in certain aspects to the finite element method). A program has been obtained that, taking advantage of the static calculation issued from the initial program, performs the global buckling calculation of the 2D structure using a matrix approach and groups everything in a single code.

Keywords

Global buckling, Structural Analysis, Python, SAP2000, Direct Stiffness Method.

Índice general

Índice de figuras

Índice de tablas

Índice de ecuaciones

1 Introducción y objetivos

La búsqueda de la optimización del diseño de estructuras hace necesario el cálculo de estas de manera que se garantice que no se superen los estados límite de servicio y los estados límite últimos. Estas deben resistir las cargas que se prevén en su vida útil en servicio, de la forma más económica y eficiente posible. Entre los cálculos a realizar por los ingenieros de estructuras para conseguir este objetivo, el cálculo de pandeo es un cálculo habitual en el proceso de diseño de las mismas debido a la importancia de este.

Este método es especialmente efectivo en sistemas estructurales formados por barras, donde las simplificaciones del modelo de barras se hacen más exactas cuanto mayor es la relación L/h , donde L es el largo de la barra y h la dimensión transversal de la misma, tendiendo este error a cero cuanto mayor es este coeficiente. Por este motivo, este modelo solo se aplicará para el cálculo con barras.

El pandeo es un fenómeno de inestabilidad indeseable en cualquier elemento estructural. Cuando una barra se encuentra sometida a una carga a compresión $\boldsymbol{P}$ y esta llega a la carga crítica de pandeo $\boldsymbol{P_{critica}}$, la barra flecta súbitamente, produciéndose grandes desplazamientos transversales de la misma para incrementos muy pequeños de carga [1]. Por ello es de gran interés realizar el cálculo a pandeo global de pórticos planos, para conocer a que carga se produce el pandeo de la estructura con la intención de no superarla.

En el ámbito profesional, lo normal es el uso de software comercial para realizar estos cálculos. Ejemplos de este software son CYPE® [2], SAP2000® [3], ANSYS® [4], Dlubal [5] y Tekla® Structures [6]. Todos estos programas son de los más utilizados en el diseño estructural y todos ellos son de uso por licencia previo pago de la misma. Es por ello por lo que se vuelve interesante también que exista la posibilidad de cálculo mediante programas gratuitos al alcance de todos los usuarios, especialmente para el ámbito educativo, donde los alumnos pueden ver aplicados los métodos que aprenden en clase de manera teórica.

La realización de un programa para su cálculo mediante código libre es interesante para hacer accesible este tipo de análisis a un mayor número de usuarios. Como ventaja adicional, el software libre permite además la realización de programas a medida del usuario y su posterior modificación de manera sencilla para que cada persona pueda modificarlo a su gusto o mejorarlo para otros usuarios, haciendo que estos no dependan de software comercial.

1.1 Antecedentes

Para poder realizar este análisis, se va a proceder a la implementación de los métodos de cálculo matricial estructural en el software del programa. Estos métodos de cálculo son conocidos en el ámbito de la ingeniería debido a que son los utilizados para la resolución de sistemas mecánicos en el software comercial.

El Método de Directo de Rigidez, al que se mencionará de ahora en adelante MDR, consiste en el cálculo estructural de esfuerzos considerando un sólido y discretizando este para los casos de análisis no lineales (pandeo, plásticos, etc.) hasta que los valores tienen una convergencia en la que se considera que el error de cálculo debido a la linealización es suficientemente pequeño. A partir de un cierto umbral de discretización la resolución del problema tiende asintóticamente a la resolución real del sólido si se hubiesen considerado todos los términos que relacionan las cargas entre sí. Esto no ocurre en los casos analizados lineales y estáticos cuyo resultado si es independiente del mallado.

1.2 Marco teórico

Para calcular el pandeo de una barra cualquiera se realiza el equilibrio en la configuración deformada de una barra como la que se puede ver en la Figura 1.

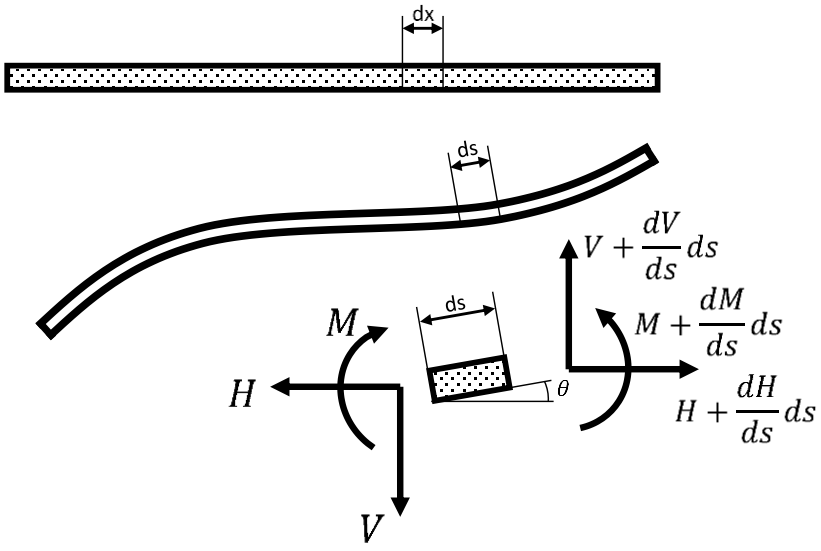


Figura 1. Problema de equilibrio en configuración deformada de una barra.

Considerando la hipótesis de pequeños desplazamientos y después de aplicar las condiciones de contorno sobre este problema obtenemos el siguiente sistema matricial de ecuaciones (Ecuación 1).

$$\begin{pmatrix} N_a \\ V_a \\ M_a \\ N_b \\ V_b \\ M_b \end{pmatrix} = \begin{pmatrix} EA/L & & & & & \\ 0 & C_4(N)EI/L^3 & & & SIM & \\ 0 & C_1(N)EI/L^2 & C_2(N)EI/L & & & \\ -EA/L & 0 & 0 & EA/L & & \\ 0 & -C_4(N)EI/L^3 & -C_1(N)EI/L^2 & 0 & C_4(N)EI/L^3 & \\ 0 & C_1(N)EI/L^2 & C_3(N)EI/L & 0 & -C_1(N)EI/L^2 & C_2(N)\mathrm{EI}/L \end{pmatrix} \cdot \begin{pmatrix} u_a \\ v_a \\ \theta_a \\ u_b \\ v_b \\ \theta_b \end{pmatrix}$$

Ecuación 1. Sistema matricial de ecuaciones de pandeo global.

Donde los coeficientes $C_1(N)$, $C_2(N)$, $C_3(N)$ y $C_4(N)$ son las funciones de estabilidad y vienen dadas por las expresiones que se ven en la Ecuación 2.

$$C_1(N) = \frac{a^2 \cdot L^2 \cdot \big(1 - cos(a \cdot L)\big)}{\Delta} \;\; ; \;\; C_2(N) = \frac{a \cdot L \cdot sen(a \cdot L) - a^2 \cdot L^2 \cdot cos(a \cdot L)}{\Delta}$$

$$C_3(N) = \frac{a^2 \cdot L^2 - a \cdot L \cdot sen(a \cdot L)}{\Delta} \;\; ; \;\; C_4(N) = \frac{a^3 \cdot L^3 \cdot sen(a \cdot L)}{\Delta}$$

$$\Delta = 2 \cdot \big(1 - cos(a \cdot L)\big) - a \cdot L \cdot sen(a \cdot L)$$

Ecuación 2. Funciones de estabilidad del problema de pandeo global.

Debido a la complejidad de estas funciones para su implementación en los programas informáticos, estas funciones de estabilidad se aproximan linealmente mediante un polinomio de Taylor de la forma que se puede ver en la Figura 2. Los errores que se pueden cometer por hacer esta aproximación se mitigan mallando de la estructura.

$$C_1(N) \approx 6 - \frac{L^2}{10 \cdot EI} \cdot N, \qquad C_2(N) \approx 4 - \frac{2 \cdot L^2}{15 \cdot EI} \cdot N$$

$$C_3(N) \approx 2 + \frac{L^2}{30 \cdot EI} \cdot N, \qquad C_4(N) \approx 12 - \frac{6 \cdot L^2}{5 \cdot EI} \cdot N$$

Figura 2. Funciones de estabilidad aproximadas.

Esto permite que el problema de pandeo global pase a ser un problema aproximado al problema teórico, donde $\underline{\underline{K}}(N) \approx \underline{\underline{K}} - N \cdot \underline{\underline{K_G}}$, cuyo valor final tenderá a la solución exacta cuanto mayor sea el número de elementos en los que se discretice el problema. La resolución del problema de pandeo global busca que los desplazamientos, tras aplicar las condiciones de contorno, no estén determinados y por tanto se debe cumplir la Ecuación 3.

$$\left|\underline{\underline{K}} - \lambda \cdot \underline{\underline{K}}_G\right|_{est} = 0$$

Ecuación 3. Problema de autovalores para pandeo global de la estructura.

Esto equivale a resolver el problema de autovalores usando las matrices $\underline{\underline{K}}$ y $\underline{\underline{K}}_G$, que son las matrices de rigidez y de rigidez geométrica respectivamente (las cuales están definidas para una sola barra en coordenadas locales en la Ecuación 4 y la Ecuación 5) y los autovalores λ serán los coeficientes del factor de carga del problema. El factor de carga λ es el factor por el que se multiplican todas las cargas de la estructura, el cual, tras alcanzarse, provoca un cierto fenómeno en la estructura, en este caso el pandeo de la misma.

Cada autovalor lleva su autovector asociado. Estos autovectores indican la solución en desplazamientos (de amplitud indeterminada) que se puede relacionar con la deformada debida al fenómeno de pandeo. Por tanto, como significado físico se tiene que el autovalor es el factor multiplicador del estado de cargas que actúan sobre la estructura que provoca el pandeo según el autovector. Si el menor de esos multiplicadores es mayor que la unidad significa que, para las cargas actuantes, no aparecerán fenómenos de inestabilidad.

$$\underline{\underline{K}} = \begin{pmatrix} EA/L & & & & & \\ 0 & 12EI/L^3 & & & SIM & \\ 0 & 6EI/L^2 & 4EI/L & & & \\ -EA/L & 0 & 0 & EA/L & & \\ 0 & -12EI/L^3 & -6EI/L^2 & 0 & 12EI/L^3 & \\ 0 & 6EI/L^2 & 2EI/L & 0 & -6EI/L^2 & 4\mathrm{EI}/L \end{pmatrix}$$

Ecuación 4. Matriz de rigidez de una barra en coordenadas locales.

$$\underline{\underline{K}}_G = \begin{pmatrix} 0 & & & & & \\ 0 & {}^{6}/_{5L} & & & SIM & \\ 0 & {}^{1}/_{10} & {}^{2L}/_{15} & & & \\ 0 & 0 & 0 & 0 & & \\ 0 & {}^{-6}/_{5L} & {}^{-1}/_{10} & 0 & {}^{6}/_{5L} & \\ 0 & {}^{1}/_{10} & {}^{-L}/_{10} & 0 & {}^{-1}/_{10} & {}^{2L}/_{15} \end{pmatrix}$$

Ecuación 5. Matriz de rigidez geométrica de una barra en coordenadas locales.

Hay que indicar de nuevo que estas matrices están definidas en coordenadas locales para una sola barra (en los ejes longitudinal y transversal de la misma) y que el problema completo de la estructura requiere el ensamblado de las mismas en coordenadas globales (es decir, en los ejes X e Y comunes a todas las barras en las que se define todo el conjunto de barras) para poder realizar correctamente la resolución del problema de autovalores.

Este será, por tanto, el modelo matemático que se implementará en el programa en código Python [7], de manera similar a la realizada por los programas comerciales de cálculo estructural indicados en la página 13 de este documento.

1.3 Conceptos básicos

Aplicando el método exacto descrito con anterioridad al caso de una barra aislada se obtienen la conocida fórmula de Euler para pandeo de barras aisladas. La siguiente fórmula (Ecuación 6) tiene como resultado la carga crítica de pandeo P_{cr} y es función de las condiciones de contorno que se pueden dar en los diferentes problemas, obteniéndose los valores del parámetro β para cada caso.

$$P_{cr} = N_{cr} = \frac{\pi^2 EI}{(\beta L)^2}$$

Ecuación 6. Fórmula de Euler para pandeo de barras aisladas.

A continuación, a modo de recuerdo de los casos más usuales, se pueden ver algunos de los valores del parámetro β en función de las condiciones de contorno (Tabla 1). El caso de viga biempotrada con parámetro β = 0,5 se utilizará para realizar las verificaciones preliminares del código desarrollado en el apartado 5.2; Análisis de una barra aislada con valor teórico exacto.

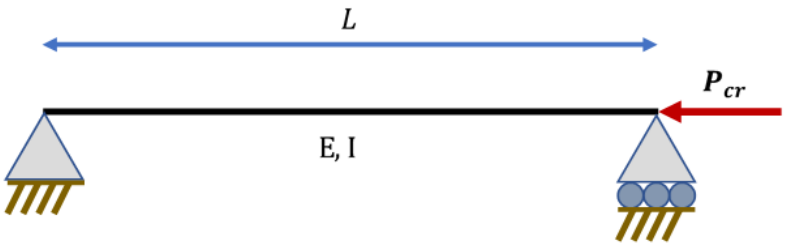
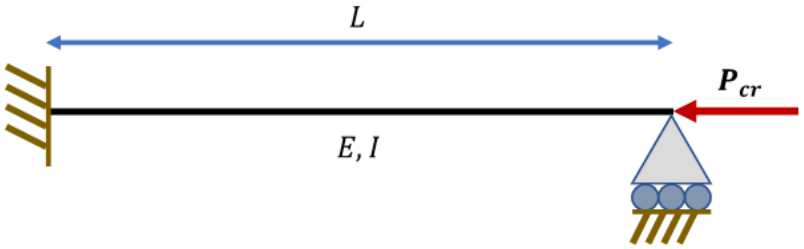
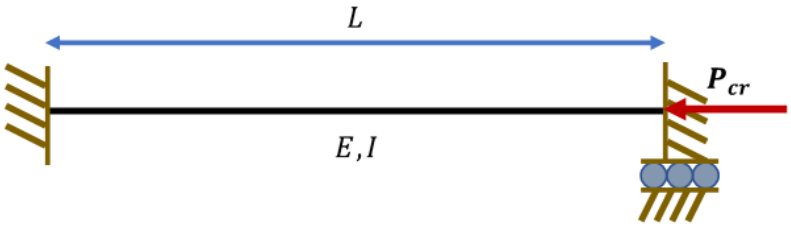
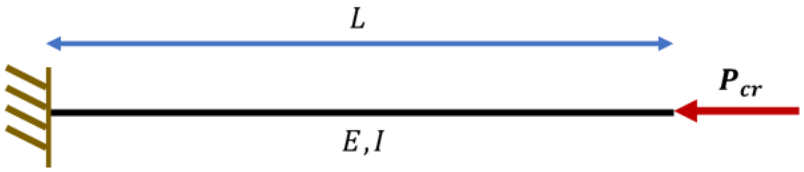
Ejemplo	β
	1
	≈ 0,7
	0,5
	2

Tabla 1. Ejemplos del parámetro β en la fórmula de Euler.

1.4 Objetivos

Mediante el aprovechamiento de dos códigos base programados en Python se plantea la consecución de los objetivos que se mencionaran a continuación. Uno de los códigos contiene una implementación del método directo de rigidez para la realización de un cálculo de estructuras planas con cargas estáticas. El otro código realiza el cálculo de pandeo de vigas continuas.

1.4.1 Objetivos técnicos

- **Programación de un software para la realización de cálculos de pandeo global de estructuras planas.**

Por una parte, la obtención de un programa sencillo para la realización de cálculos de pandeo global estructuras planas. Este programa se realizará enteramente en Python 3.7 mediante el IDE Spyder (v.4.1.2) [8] , incluido en la *suite* de programas Anaconda [9], todo ello software de código libre. Un IDE es un entorno de desarrollo integrado el cual facilita a un programador la realización del programa, recordando al usuario, por ejemplo, las variables que tiene definidas, las clases o posibles erratas en el código (Figura 3).

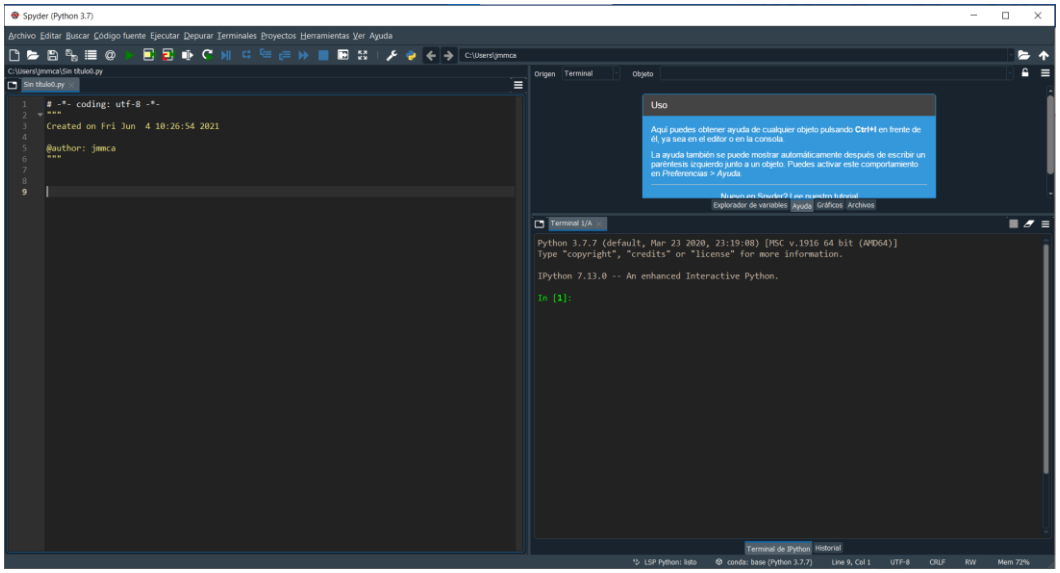


Figura 3. Entorno de desarrollo gráfico de Spyder.

- **Integración del programa para la obtención de los factores de pandeo global con el programa ya existente de cálculo de estructuras para cargas estáticas.**

Por otro lado, se busca la cohesión de este código con el código disponible de cálculo a cargas estáticas de estructuras planas. También se busca la obtención de software de código libre que permita su uso por los diferentes usuarios sin la necesidad de la utilización de software comercial el cual, además de tener costes de uso por licencia, requiere una disponibilidad de coste computacional mayor, al estar estos programas cargados con mayor número de opciones.

- **Obtención de la deformada de los diferentes modos de fallo en el caso de pandeo global y visualización del modo de fallo la estructura.**

Tener la posibilidad de visualizar los diferentes modos de fallo y las deformadas que se crean en la estructura cuando esta pandea es de gran interés de estudio para evitar este tipo de fallos, que se pueden evitar mediante la introducción de elementos rigidizadores (como los arriostramientos) y revisar que no hay interferencia con ningún elemento que pueda suponer un peligro adicional superior al ya de por si peligroso fallo por pandeo.

- **Obtención de la deformada estructural para cargas estáticas sin pandeo de la estructura y visualización de la misma.**

Aprovechar las funciones creadas para la obtención de las deformadas de los casos de pandeo global 2D, implementándolas en el código previo de StruPy2D, permitiendo obtener adicionalmente una visualización gráfica de los resultados de desplazamientos de las barras cuando estas sufren cargas estáticas, sin evaluar el pandeo de la estructura.

1.4.2 Objetivos generales

Así mismo, como objetivos adicionales de este TFG a los indicados en el apartado 1.4.1. Objetivos técnicos, se pueden considerar los siguientes objetivos.

- **Profundizar en aspectos computacionales adquiridos en la asignatura de Estructuras y Construcciones Industriales.**

- **Aplicar y ampliar conocimientos de Python y cálculo mediante ordenador a un problema concreto de interés en ingeniería.**

- **Documentar el código desarrollado para su futura consulta por otros alumnos y la posible ampliación de las capacidades del mismo, en un proceso de mejora continua.**

- **Mejora de las capacidades de razonamiento crítico, analisis, síntesis y trabajo autónomo.**

1.5 Metodología

En lo que a programación respecta, se seguirán las recomendaciones del paradigma de programación orientada a objetos (POO), a imagen de un código base de partida. Esta programación se realizará bajo el principio de clasificación de las variables en clases, cada una con sus métodos y sus atributos correspondientes. En POO, las clases que se definen en los programas sirven para representar elementos de la vida cotidiana, pudiendo ser estos físicos o no.

Por ejemplo, se puede considerar una clase para las barras y otra para un estudio a pandeo global. A los métodos de una clase se les puede considerar, para facilitar su comprensión, como las acciones que pueden tener lugar en una clase, por ejemplo, a una barra se le puede añadir o quitar una carga, esta acción se considerará un método. En ocasiones los métodos pueden ser denominados funciones. Los atributos por su parte son las propiedades que pueden tener estas clases. Siguiendo con el ejemplo de una barra, un atributo puede ser la longitud, el material, la sección o los nodos inicial y final de la misma.

Siguiendo estos principios se realizará la programación correspondiente a la clase Buckling que se definirá para los casos de cargas a pandeo, con sus atributos y métodos correspondientes y las modificaciones necesarias en el resto del código base para cumplir con los objetivos marcados en el apartado 1.4 Objetivos.

En cuanto a los contenidos técnicos que tendrán lugar, se siguen las formulaciones vistas en las asignaturas de Matemáticas III en lo referente a cálculo mediante computación, Fundamentos de la Informática en el campo de la programación, y de las asignaturas de Resistencia de Materiales y Estructuras y Construcciones Industriales para la evaluación crítica de los resultados obtenidos mediante los conocimientos adquiridos en el campo de los materiales y de los elementos estructurales. Todas estas asignaturas forman parte del plan de estudios del grado en Ingeniería Mecánica, impartido por la escuela de Ingenierías Industriales de la Universidad de Valladolid. Este trabajo de fin de grado es la culminación de ese mismo grado.

Para asegurar la corrección numérica de los ejemplos de aplicación, se hará una comprobación de los mismos comparando los resultados obtenidos mediante la aplicación creada en código Python con el software comercial SAP2000, en el cual se reproducirán de nuevo los problemas y se observarán las diferencias entre ambos, en busca de solventar posibles erratas en el código generado. SAP2000 es un programa de cálculo estructural reconocido en el mundo de la ingeniería estructural, el cual es utilizado en numerosos

proyectos a nivel global, por lo que se puede considerar sin lugar a duda que si los problemas están bien formulados en este software y los resultados coinciden, se podrá confirmar que el cálculo obtenido mediante Python es correcto, procediendo a su validación.

2 Contenido de partida

Para la realización de este programa se partió de dos códigos base realizados por el profesor Álvaro Magdaleno González. Un programa completo llamado StruPy2D, programado mediante clases y métodos (también llamados funciones) en diferentes *scripts*, que contiene todos los elementos necesarios para la obtención de esfuerzos en los extremos de las barras de una estructura ante cargas estáticas. El segundo programa, llamado viga_multi.py, contiene un código de programación para el cálculo a pandeo en barras rectas formadas por múltiples vanos y con apoyos definidos por el usuario en los extremos de los vanos, pero con todos los elementos situados en una única dimensión. Este programa no seguía la programación mediante métodos y clases, como si ocurre con el programa StruPy2D. Ambos códigos están programados en Python 3.6.

El término *script* se utiliza para designar a programas de relativa sencillez, formados por un fichero de texto que contiene una serie de líneas de código que se procesan secuencialmente. Los *scripts* suelen formar parte de un programa mayor, de esta forma se ayuda a la segmentación del programa, facilitando su comprensión por parte del programador y la reutilización o implementación de posibles mejoras en el código.

Dentro de los *scripts* se encuentran definidas las clases y los métodos. Aplicar a una variable del programa una clase es una manera de dar a esa variable las propiedades que se quiera que tenga. Así por ejemplo en el programa se puede definir una clase Barra, que dará a una variable que se defina en el programa las propiedades que queremos que tenga esa barra, pudiendo considerarse unas por defecto inicialmente y modificando estas posteriormente. Ejemplos de propiedades que se pueden definir dentro de esta clase serian, el material, la sección, la inercia, la densidad... Esto facilita ver las variables como objetos de la vida cotidiana, ayudando al usuario a comprender mejor lo que está ocurriendo en el programa y facilitando la introducción de datos correctos, ya que cuando defina una barra esta le solicitará concretamente que propiedades tiene que darle al usuario. Este paradigma de programación se llama programación orientada a objetos, o POO, y facilita la comprensión de los programas al programador y a los usuarios.

A continuación, se pasan a describir los *scripts* de mayor relevancia por los que está compuesto el programa de partida StruPy2D.

2.1 *Scripts* principales

El programa base StruPy2D de cálculo estático de estructuras planas consta de múltiples ficheros, tanto en lo referente a la realización del cálculo estructural como a la representación gráfica de los elementos al usuario del mismo. Se procede a describir los *scripts* con más relevancia del programa, haciendo mayor hincapié en los módulos relativos al cálculo y referenciando únicamente los relativos a la representación gráfica de elementos.

-Bar.py:

El archivo Bar.py sirve para definir barras, dejando almacenado dentro de estas las propiedades de las mismas. Un objeto barra definida por el usuario necesita que se introduzcan en esta las siguientes variables; nodo inicial y nodo final, material y sección de la barra.

Si las propiedades material y sección no se introducen en el programa este está predeterminado para adquirir por defecto los valores genéricos almacenados en el programa, pudiendo ser estos modificados por el usuario si lo considera oportuno en los *scripts* Material.py, Section.py

Para la introducción de los puntos inicial y final de la barra el programa comprueba que los puntos introducidos han sido considerados como nodos anteriormente, dándoles la clase Node.

Dentro del *script* bar.py también se encuentran definidas dos clases más BarSol y BarList las cuales sirven para el almacenado y tratamiento de datos sobre la estructura y sobre el cálculo realizado.

-config.py:

Contiene las condiciones de contorno predefinidas por el programador, apoyo fijo y apoyo móvil tanto horizontal como vertical, así como empotramiento fijo y empotramiento móvil horizontal y vertical. También define el valor de tolerancia de los problemas, estipulado en 10^{-8}, considerando cualquier número por debajo de ese valor igual a 0.

-Load.py:

Con este programa, se define la clase Load (Fuerza), con la que se definen las cargas de los problemas planteados al programa. Permite la introducción de fuerzas y momentos, cargas térmicas, desplazamientos y giros prescritos.

Se puede distinguir entre cargas nodales (NOD), puntuales (PUN), distribuidas (DIS), deformaciones (DEF) y cargas térmicas longitudinales y transversales (TER_L y TER_T respectivamente).

En el *script* Load hay dos clases más definidas; LoadList genera una lista con las cargas del problema ejecutado y LoadGroup permite asignar determinadas las cargas generadas individualmente a un grupo para aplicar todas simultáneamente en conjunto a una o más estructuras.

-Material.py:

Para poder realizar la asignación de los materiales a la barra, esta se realiza mediante el *script* Material.py, el cual, mediante la clase Material, guarda los atributos correspondientes a cada material de la estructura; Identificación del material (ID), Módulo de Young (E), densidad (rho) y el coeficiente de dilatación térmica (al). Esta clase tiene asignado un material por defecto, 'default' con las propiedades del acero común.

De forma adicional tiene una clase con la que generar listas con diferentes materiales para que el usuario pueda, como con el caso de LoadGroup, aplicar los materiales de manera conjunta en los modelos para obtener mejor los resultados comparativos entre estos.

-Node.py:

Node.py permite la definición de los nodos de los que se va a componer la estructura. Para poder crear un nodo se deben de dar las coordenadas del mismo, "X", "Y" y "Th" (ángulo), al estar este en el plano. El ángulo Th que se puede aplicar en las variables de tipo Node sirve para poder obtener los resultados en coordenadas nodales.

El *script* dispone también de las clases NodeSol, que genera las soluciones en desplazamientos y fuerzas en los nodos en coordenadas tanto locales como globales del problema, y NodeList, la cual crea una lista con los nodos analizados en la estructura.

-Section.py:

Similar al *script* Material.py, su función es la de la identificación de las secciones utilizadas en las barras mediante la clase Section, donde se le han de introducir los atributos correspondientes a la sección, siendo estos; área (A), inercia (I), y altura de la sección en el plano de canto (h). Esta clase tiene unos valores preestablecidos por defecto si no se indica alguno de estos, los cuales pueden modificarse en el programa por el usuario.

La clase SecList, que se encuentra dentro de este *script*, permite generar listas con las secciones definidas por la clase Section o conjuntos de listas SecList anteriores para realizar los cálculos comparando las diferentes secciones entre sí.

-Static.py:

Uno de los módulos más importantes de StruPy2D es Static.py. Este módulo es el encargado de realizar el cálculo estático de la estructura definida por el usuario. Tiene definida dentro de él la clase Static, que permitirá la asignación de los resultados del análisis de la estructura a una variable indicada previamente. Para poder ejecutarse se ha de pasar una variable definida como estructura con la clase Structure para realizar su cálculo. Como propiedades tiene los vectores de desplazamientos, fuerzas y reacciones y los vectores de nodos y barras con los que se ha realizado el análisis.

El *script* tiene definida una función show() con el que el programa devuelve al usuario una representación gráfica del caso de cargas a estático analizado, mostrando por pantalla la estructura y las cargas y condiciones de contorno descritas en el modelo.

-Structure.py:

Mediante Structure.py, se realiza la definición de la clase Structure para hacer referencia a la estructura del problema, la cual se encarga de la construcción de la matriz de rigidez y de masas de toda la estructura en coordenadas globales. Guardando en su interior todos los datos que son necesarios para el ensamblado de los elementos que la componen, obtenidos de las clases definidas anteriormente.

Dentro de este programa está implementado un código que realiza la condensación estática de la matriz de rigidez, buscando la optimización de la ejecución en el cálculo. También permite la liberación de grados de libertad entre las diferentes barras de la estructura. En la clase Structure están definidos los métodos que permiten añadir y quitar nodos, barras, materiales, secciones, cargas, grupos de cargas, peso propio de la estructura, indicar la fuerza de gravedad del problema, obtener matrices nodales, fuerzas de empotramiento y obtener un gráfico de la estructura.

Cabe indicar que la dirección en la que actúa la gravedad $(\boldsymbol{g})$ en los problemas definidos por el usuario se encuentra indicada por la dirección global $-\overline{\boldsymbol{OY}} = \overline{\mathbf{g}}$

2.2 *Scripts* para gráficos

StruPy2D también incluye una carpeta Graphics, la cual contiene los *scripts* correspondientes a la realización de la representación gráfica de las estructuras y las cargas programadas por el usuario para facilitar una correcta interpretación de resultados y servir al usuario como un elemento adicional para la comprobación de que los datos introducidos al modelo son los que realmente está utilizando el programa para realizar los cálculos y no se han definido mal estos.

Esta carpeta contiene los siguientes programas; **Axes.py**, **BC.py, graphBar.py, graphLoad.py, graphNode.py y Style.py.** Estos programas se encargan de la representación gráfica de ejes, condiciones de contorno, barras, cargas, nodos y de los estilos (escala, orden…) respectivamente.

3 Modificaciones en el programa

Para la realización del programa de cálculo de pandeo global para estructuras en 2D, ha sido necesario la programación de un nuevo archivo Buckling.py, donde se han definido una nueva clase, correspondiente al cálculo a pandeo (Buckling) y los métodos correspondientes a la misma.

Por otra parte, se han realizado una serie de modificaciones para implementar este programa con el resto del código de StruPy2D. Además, en el apartado gráfico, se ha realizado la implementación de la visualización de la deformada tanto a pandeo global como para el cálculo a estático de los problemas. A continuación, se procede a describir las modificaciones realizadas en el programa.

3.1 *Script* Buckling.py para la obtención de los coeficientes de pandeo global correspondientes al problema de autovalores y su representación gráfica

En este *script* se encuentran los elementos necesarios para realizar tanto los cálculos del caso de pandeo, la determinación de los factores de carga a pandeo λ_p y los desplazamientos nodales del problema, como para realizar la visualización de los resultados del mismo para facilitar la interpretación de estos por parte del usuario.

3.1.1 Cálculos de Pandeo Global en *script* Buckling.py

En el nuevo archivo programado, Buckling.py, se han introducido las funciones necesarias para recoger los resultados de los axiles del caso estático, calcular la matriz de rigidez de la estructura, obtener el vector de condiciones de contorno del problema y resolver el problema de autovalores y autovectores correspondiente al caso.

Este *script* define la clase Buckling, a la que el usuario necesitará indicarle como dato un caso de cálculo estático, del que se obtendrán los resultados de esfuerzos, la estructura del problema y las matrices de rigidez y masas.

Mediante la función "**run()**" definida dentro de la clase, se computa la solución del cálculo a pandeo.

En la clase Buckling se han definido los siguientes métodos:

- **recalculateKKGM**(self)

Su labor es la de recoger los axiles de la estructura, generar un vector con estos axiles cambiados de signo y comunicárselos a cada una de las barras del

modelo como una propiedad de la misma para recalcular la matriz de rigidez geométrica del problema. Este método llama a la función assemblyKNKGM, que se encuentra definida en la clase Structure, que reconstruirá las matrices de la estructura completa en coordenadas globales, $\boldsymbol{K_g}$, $\boldsymbol{NKG_g}$ (matriz de rigidez geométrica de la estructura en coordenadas globales con los axiles ya aplicados en cada barra) y la matriz de masas $\boldsymbol{M_g}$, la cual no tienen ninguna utilidad en la versión actual del programa y servirá, en el futuro, para realizar cálculos de modos propios.

El método **assemblyKNKGM()** es el mismo que contenía el archivo de partida de StruPy2D salvo porque se ha añadido en este el ensamblaje de la matriz $\boldsymbol{NKG_g}$ (ver Ecuación 5, página 17).

- **cdcVectorGen**(self)

Se encarga de la construcción de un vector lógico de condiciones de contorno utilizando la propiedad BC (condiciones de contorno, en sus siglas en inglés) correspondiente a la clase Node, la cual contiene una lista de tres componentes con las condiciones de contorno $[\boldsymbol{u_x}, \boldsymbol{u_y}, \boldsymbol{u_\theta}]$ que indica los grados de libertad fijados o liberados de cada nodo de la estructura, mediante variables booleanas con valor True para los grados de libertad liberados y False si esos grados de libertad se encuentran fijados. Así pues, un apoyo móvil en la dirección horizontal generará un vector $= [\boldsymbol{True}, \boldsymbol{False}, \boldsymbol{True}]$.

- **generateSolution**(self, K, NKG)

Esta función, recibe como parámetros las matrices de rigidez $\boldsymbol{K_g}$ y $\boldsymbol{NKG_g}$ correspondientes al problema de autovalores y resuelve el problema de valores propios indicado en la Ecuación 3 (página 16). Para ello hace uso de la librería de algebra lineal scipy.linalg [10] que viene importada por defecto en el programa utilizado mediante la función **.eig**(K,NKG)**,** la cual devuelve dos matrices, una con los resultados de los autovalores, los cuales son los valores correspondientes a los factores de carga λ_p de cada modo de fallo y por otro lado devuelve una matriz los autovectores, que contienen los valores de los desplazamientos de cada uno de los modos de fallo de las libertades de la estructura.

3.1.2 Visualización de resultados de Pandeo Global en *script* Buckling.py

Se ha introducido un apartado de visualización de los resultados en el modelo de pandeo, utilizando la librería matplotlib **[11]** dentro del programa Bucking.py, para facilitar la correcta visualización por parte del usuario de los diferentes modos de fallo con sus deformada correspondiente.

Matemáticamente, es posible obtener la solución exacta de la deformada de cada una de las barras de la estructura mediante las leyes de esfuerzos de las

barras de la estructura y realizando el cálculo completo de las mismas para cada caso de pandeo. La obtención de estos resultados, aunque académicamente es interesante, en la práctica no es la llevada a cabo por ninguno de los programas de cálculo de estructuras como práctica habitual.

En los programas de elementos finitos comerciales, el método de obtención de la deformada de la estructura es mediante la interpolación de un polinomio de grado $\boldsymbol{n-1}$ donde $\boldsymbol{n}$ son los grados de libertad por dirección de cada elemento o barra. En nuestro caso, en el que se realiza una visualización en 2D, se tienen 4 grados de libertad para la componente en la dirección transversal de las barras y solo 1 grado de libertad en la componente axial de las mismas debido a la dependencia de uno de los grados de libertad en esa dirección. Esto conlleva que el grado del polinomio a interpolar para poder realizar una representación es de grado **3.**

El apartado gráfico del caso a pandeo global viene definido dentro de la clase Buckling mediante la función **show()**, la cual es necesario llamar una vez ejecutado el modelo a pandeo.

```
- def show(self, axes = None, bc_scale = 0.04, k_scale = 0.1,
f_scale = 0.075, showNodes = False, node_style = defNodeStyle,
bar_style = defBarStyle, load_style = defLoadStyle)
```

Aprovechando el método **show()** de la clase Structure, se ha replicado el elemento de visualización de la estructura indeformada del problema eliminando de este la representación de las cargas del problema, para superponer sobre esta la deformada a pandeo global para el modo de fallo seleccionado por el usuario. Desde este método se llama a la función **deformateBuckling(self)** perteneciente a la clase Buckling para calcular y sobreponer la estructura deformada.

```
-def deformateBuckling(self)
```

Esta función realiza el cálculo de la deformada de cada barra en coordenadas locales. Para ello hace una interpolación de un polinomio de grado 3 $(n = 3)$ utilizando los desplazamientos nodales de la estructura, obtenidos del cálculo de autovectores del modo de pandeo, como condiciones de contorno en los extremos. Mediante la función **solve()** de la librería scipy.linalg resuelve el polinomio a encontrar.

Es conveniente entender que, aunque la solución de la estructura deformada obtenida puede coincidir con la solución real de desplazamientos, no será siempre así, y esto dependerá de las cargas intraelementales de la barra. También es interesante indicar que, cuanto más se refine el mallado de la estructura, más se aproximará la solución obtenida por la interpolación del polinomio de cada barra a la solución real de desplazamientos de la misma.

Una vez obtenida la deformada en coordenadas locales de cada barra, esta es llevada a su posición dentro de la estructura global mediante un cambio de base para aplicar la rotación de la misma y una traslación a su coordenada inicial ya desplazada debido al pandeo estructural.

3.2 Modificación de funciones en el *script* Bar.py para la obtención de la matriz de rigidez geométrica correspondiente a una barra

3.2.1 getNKGe()

Este método definido dentro de la clase barra, sirve para calcular la matriz de rigidez en locales de la barra multiplicada por el valor de su axil cambiado de signo, considerando el axil de la barra a compresión el axil de valor positivo para el caso a pandeo, al contrario, a como ocurre en el convenio de signos para el caso estático (Figura 4). Esto viene definido por la definición del problema de autovalores de caso a pandeo indicada en la Ecuación 3 (página 16).

$$N_{pandeo} = -N_{estático}$$

Figura 4. Convenio de signos para el axil del caso de pandeo.

3.2.2 seeNKGe()

Con la función seeNKGe() el usuario puede obtener la matriz de rigidez geométrica multiplicada por su axil de la barra que considere en coordenadas locales y visualizarla en pantalla. Esta función replica las ya existentes para la visualización de las matrices $\underline{\underline{K_e}}$ y $\underline{\underline{M_e}}$ (la matriz $\underline{\underline{K_e}}$ se puede observar en la Ecuación 4).

3.2.3 modificación en getGlobal()

En el caso de la función getGlobal(), ya definida en la clase Bar anteriormente se introdujeron las modificaciones necesarias para que el programa procese la matriz de rigidez de la barra con el axil ya aplicado NKG_e (Ecuación 5) a la vez que las matrices K_e y M_e de la barra, para la obtención de los valores de estas matrices en coordenadas globales K_g, NKG_g, y Mg.

3.3 Modificación de variables self en el *script* Bar.py

3.3.1 self.nBu

Necesaria su introducción para el cálculo de $NKGe$, se definió una propiedad nueva nBu en la clase Bar donde almacenar el axil del problema de pandeo para el cálculo de la matriz de rigidez geométrica de cada barra en coordenadas locales, para su posterior cambio de base y ensamblaje. Por defecto este dato viene fijado con valor nBu=1 si no se cambia por el caso estático del problema, permitiendo que el programa realice el cálculo de la matriz antes de tener resuelto un caso estático si se considera que el axil de cada barra es $N = 1$. Esta variable almacena la media del valor del axil en los extremos de la barra.

3.3.2 self.NKGe

Este atributo de las variables tipo barra (Bar), almacena la matriz de rigidez de la barra multiplicada por su axil para el caso de pandeo en coordenadas locales para su posterior uso o consulta.

3.3.3 self.NKGg

De forma idéntica a $NKGe$, almacena como propiedad de una barra la matriz de rigidez de esta multiplicada por su axil a pandeo, pero en este caso en coordenadas globales.

3.4 Modificación de funciones en el *script* Structure.py para el correcto ensamblado de las matrices de rigidez de las diferentes barras

3.4.1 assembleKNKGM() en lugar de assembleKM()

La función **assembleKM()** definida en el programa base inicial ha tenido que ser ampliada para procesar el ensamblado de la matriz de rigidez geométrica. Debido a esto se le cambió el nombre a **assembleKNKGM()** para hacer referencia a este cambio en el contenido y funcionalidades de la misma.

Construye las matrices de rigidez, rigidez geométrica y matriz de masas de toda la estructura analizada ya ensambladas en coordenadas globales.

3.4.2 modificación en getNodalMatrices()

En **getNodalMatrices()** fue necesaria la inclusión de las líneas de código correspondientes a la obtención de la matriz de rigidez geométrica en coordenadas nodales. Esta función ya realizaba esta labor con la matriz de rigidez y la matriz de masas y solo requirió de una adaptación para introducir la matriz de rigidez geométrica.

3.5 Modificación de variables en el *script* Structure.py

3.5.1 Self.NKGg()

En la clase Structure fue necesario añadir una propiedad a la clase para almacenar la matriz de rigidez geométrica de la estructura completa, **self.NKGg**. Las matrices de rigidez y de masas de la estructura ya estaban definidas como las propiedades **self.Kg** y **self.Mg** respectivamente.

3.6 Modificación de variables en el *script* Node.py

3.6.1 Self.ubug()

Este atributo se ha introducido en la clase NodeSol para poder almacenar los desplazamientos en coordenadas globales de los nodos cuando se da el pandeo global de la estructura para cada uno de los modos.

3.7 Modificación del *script* Static.py

3.7.1 deformateStatic(escaleDef)

De forma similar a como se ha obtenido la deformada de los casos a pandeo de la estructura, se ha implementado código para graficar la deformada según el caso de carga estático de la misma. Esta funcionalidad no estaba disponible en el programa inicial de StruPy2D y se ha considerado aprovechar su implementación en el caso de pandeo para introducirla en los modelos de cargas estáticos.

Igual que pasa con los modelos a pandeo, matemáticamente se pueden obtener los desplazamientos de las barras en la estructura de forma exacta, pero a nivel de software estos cálculos son realizados mediante interpolación de un polinomio de grado tres $(\boldsymbol{n = 3})$ para cada una de las barras. Es conveniente indicar que, aunque en determinadas distribuciones de cargas concretas se pueda dar que la deformada interpolada sea la solución real no es lo habitual. Por las características del modelo implementado, cuanto más fino sea el mallado de la estructura más se aproximará la deformada reflejada por el programa a la deformada real.

El apartado gráfico del caso a carga estático viene definido dentro de la clase Static mediante dos funciones:

- La función show(), la cual es necesario llamar una vez ejecutado el modelo a pandeo.

```
def show(self, axes = None, bc_scale = 0.04, k_scale = 0.1, f_scale
= 0.075, showNodes = False, node_style = defNodeStyle, bar_style =
defBarStyle, load_style = defLoadStyle, escaleDef=30)
```

Esta función show refleja de nuevo la estructura del problema antes de la deformación de esta y llama a la función deformateStatic(escaleDef). Por defecto existe un valor de escala, pero este puede ser cambiado por el usuario cuando se llama a la función show(escaleDef=XX) si los resultados que devuelve el programa no son lo suficientemente representativos. Ha sido necesaria la introducción del factor de escala debido a que los desplazamientos habituales en las estructuras son mucho menores que las dimensiones características de las mismas (de acuerdo con la hipótesis de pequeños desplazamientos). Por lo que si no son amplificados los desplazamientos son inapreciables por parte de los usuarios en la mayoría de los modelos. Esto no ocurre en los modelos de pandeo global debido a que la estructura sufre grandes desplazamientos transversales en estos casos.

- deformateStatic(self,escale=30)

Se encarga de realizar la interpolación de los polinomios de desplazamientos en cada barra y su correcto escalado y replanteamiento en coordenadas globales. Recibe el valor de escala desde la función show().

4 Utilización del programa actual

Después de las modificaciones introducidas en el programa StruPy2D, el programa puede realizar las siguientes funciones; crear, modificar, mover y dar propiedades a nodos, barras y cargas (nodales, intraelementales, térmicas, de desplazamiento, etc.), aplicar las condiciones de contorno, diseño de diferentes estructuras, cálculo de estructuras para cargas estáticas, cálculo de la estructura para pandeo global y visualización de resultados, tanto de pandeo global como de cálculo estático.

A continuación, se procede a indicar un proceso tipo para la generación de un modelo por parte de un usuario. Si después se quiere modificar el modelo se puede realizar de forma sencilla mediante los métodos definidos en los *scripts* del programa. Así pues, se puede crear el modelo y mover un solo nodo después mediante la función **move()** específica de los nodos. Este método realizará una actualización del resto de los componentes de la estructura, como barras, cargas y fuerzas equivalentes y permitirá realizar los cálculos de nuevo. Los pasos a seguir por un usuario serían los siguientes.

4.1 Llamada a StruPy2D

En el *script* donde se quiera diseñar y calcular la estructura, lo primero que se tiene que hacer una llamada al programa StruPy2D para tener disponibles todas las funcionalidades de este. Esta llamada puede realizarse por ejemplo de la siguiente forma:

```
import StruPy2D as sp2
```

Nótese que al indicar import * as ** se está haciendo que las llamadas al programa StruPy2D puedan ser realizadas mediante la abreviatura sp2, si no fuese así cada vez que el usuario tuviese que llamar al programa tendría que indicar el texto “StruPy2D” completo.

4.2 Crear materiales y secciones

Una vez llamado a StruPy2D, se tienen que definir las variables correspondientes a los materiales y las secciones del problema. En el caso de los materiales se deben definir el nombre de estos, el módulo de Young, la densidad del material y coeficiente de dilatación térmica y en el caso de las secciones será necesario que se indiquen tanto el área de la sección, así como la inercia y altura de la misma.

Un ejemplo de cómo definir estas variables sería el siguiente:

```
materialA = sp2.Material('acero', 2.1e11 , 7850, 1.2e-5)
seccionA = sp2.Section('IPN100', A=10.6e-4, I=171e-8, h=0.1)
```

4.3 Crear los nodos

En el caso de los nodos, se pueden llegar a indicar 3 valores, **"x", "y"** y **"th"**. Si alguno de estos no está indicado por el usuario, el programa tomará como valor de la variable el valor 0. El ángulo que se indica en los nodos sirve al usuario para la obtención de los resultados en coordenadas nodales, lo cual es interesante en ciertos tipos de problemas.

Para nombrar nodos se puede realizar de la siguiente forma:

```
nodo1 = sp2.Node(0,0)
nodo2 = sp2.Node(2.5,0)
nodo3 = sp2.Node(x=5,y=0,th=0)
```

4.4 Crear las barras

Generar una barra por parte del usuario requiere un proceso similar al que se realizaría si de una barra real se tratase, para ello, se tiene que definir una variable barra, en la que se indique el nombre de esta, de donde a donde va (posiciones inicial y final con los nodos correspondientes), el material y la sección de la misma. Si no indicase material y sección de la barra el programa almacenaría en esta los valores que tiene por defecto definidos en los programas correspondientes a las clases Material y Section respectivamente.

Un ejemplo de cómo se puede definir una barra se puede ver en las siguientes líneas.

```
barra1 = sp2.Bar('barra 1', nodo1, nodo2, materialA, seccionA)
barra2 = sp2.Bar('barra 2', nodo2, nodo3, materialA, seccionA)
```

Como se ve, en este caso se han introducido como valores los nodos, los materiales y las secciones generados en el código ejemplo de los apartados 4.2 y 4.3. Estos elementos ya mencionados en el programa por el usuario ya tienen concedidos los atributos necesarios para efectuar su función. Si las variables que se introducen en el elemento Bar no tienen coherencia de clase el programa devolverá un fallo indicando que el elemento introducido no es de la clase esperada por el programa.

4.5 Crear las cargas y aplicación de estas

En el caso de las cargas, el usuario puede definir múltiples tipos de carga; puntuales, nodales, distribuidas, térmicas, desplazamientos prescritos, etc. Una vez definida la carga es necesario que el usuario indique al programa donde va a aplicar la misma. Una manera de realizar este proceso puede ser la siguiente:

```
fuerza1 = sp2.Load('Fuerza 1', 'NOD', fx=10000, fy=10000)
nodo1.addLoad(fuerza1)
#Para aplicar la fuerza al nodo, esta fuerza debe ser tipo NOD
```

Hay que tener en cuenta que se pueden aplicar más de un tipo de carga en una variable si está lo permite. Así un Nodo puede presentar una carga NOD (Nodal) y un desplazamiento prescrito (DEF).

Es necesario indicar que para la definición de las cargas intraelementales los ejes que se consideran para su aplicación son los ejes locales de la barra donde se van a ubicar y no los ejes globales del problema, considerando el origen de estos el nodo inicial de la barra, y el eje $\overline{OX} = \overline{P1P2}$ (Figura 5).

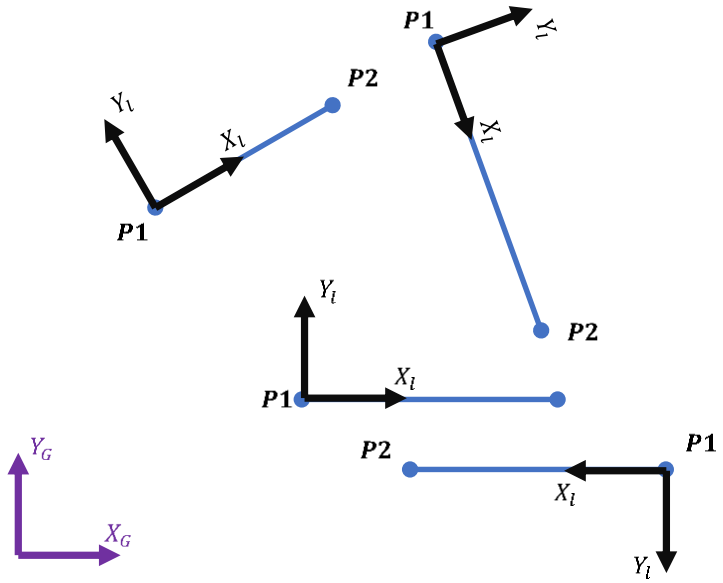


Figura 5. Ubicación de los ejes locales en StruPy2D.

4.6 Aplicación de las condiciones de contorno

Cuando un usuario aplica las condiciones de contorno del problema se aplican como atributo de los nodos ya creados. Para ello es necesario llamar al método **.setBC()** con el que el programa modifica la propiedad de condiciones de contorno, **self.BC**, de cada nodo.

```
nodo2.setBC(sp2.H_MOVING)
nodo3.setBC(sp2.CLAMPED)
```

Las condiciones de contorno se encuentran en el *script* config.py por lo que han de ser llamadas desde el *script* para poder ser utilizadas.

4.7 Crear la estructura y generar los resultados

El siguiente paso que se debe efectuar es el ensamblado de la estructura. Con ello se obtendrán las matrices globales de toda la estructura y se generará los elementos necesarios para realizar el cálculo a estático del conjunto. Dentro de los argumentos de Structure se tiene que introducir un vector con todas las barras que forman la estructura como una lista de Python ($[b_1, b_2, \dots, b_n]$ donde b_i es cada una de las barras que forman parte la estructura). Declarar una estructura puede ser algo similar a lo descrito a continuación.

```
estructura= sp2.Structure([barra1,barra2])
estructura.show() #con esta función se muestra un gráfico con la
estructura generada.
```

4.8 Crear el caso estático y activarlo

Resolver el problema estático requiere que la estructura ya haya sido declarada con sus barras, cargas y condiciones de contorno y que esta sea introducida a la clase Static como un argumento. Para definir el caso estático se declara una variable de la clase Static de la siguiente forma:

```
estatico = sp2.Static(estructura, run=True)
```

Como segundo argumento es necesario indicar que la variable run pasa a ser cierta (True), de no ser así el caso a estático estará declarado pero no se ejecutará y será necesario volver a él para activarlo.

Las variables del tipo Static tiene un gran número de datos disponibles para ser consultados por el usuario, puesto que dispone de los propios de la clase y de todos los elementos que tiene contiene dentro, permitiendo consultar los

propios del cálculo del problema de carga y los de los diferentes componentes que constituyen la estructura.

Por defecto, los cálculos realizados en los casos de cálculo estático tienen el peso propio. Si se quiere hacer que el programa no considere el peso de las barras hay que modificar la propiedad del caso estático de la siguiente forma:

```
estatico.setWeight('off')
```

Para visualizar los resultados del caso estático en lo correspondiente a la deformada de la estructura hay que llamar al método **show()**:

```
estatico.show()
```

4.9 Crear el caso de pandeo global y activarlo

Al igual que en los pasos anteriores, es necesario crear una variable con la clase Buckling a la que le pasemos un caso estático, del cual el programa extraerá los datos del caso estático necesarios para generar la matriz de rigidez geométrica global de la estructura y realizar el cálculo correspondiente. Como en las variables de tipo Static es necesario poner run = True para que el problema ejecute el cálculo.

```
pandeo = sp2.Buckling( estatico, run = True, verModo=1)
```

Por defecto si no se indica el modo de pandeo a calcular el programa retornará por pantalla el primer modo de fallo, considerado el más interesante para obtener por parte del usuario. Para visualizar la deformada de pandeo global se puede llamar a la función **show()** nuevamente.

```
pandeo.show()
```

4.10 Código completo de ejemplo

Este es un código simple de ejemplo para que el usuario pueda hacerse una idea de los datos que se pueden obtener desde el programa. Se han añadido algunas líneas en el *script* adicionales al código que se puede encontrar desde el apartado 4.1 al apartado 4.9 para obtener algunos resultados suplementarios por pantalla.

4.10.1 *Script* de entrada

```
import StruPy2D as sp2

materialA = sp2.Material('acero', 2.1e11 , 7850, 1.2e-5)
seccionA = sp2.Section('IPN100', A=10.6e-4, I=171e-8, h=0.1)

nodo1 = sp2.Node(0,0)
nodo2 = sp2.Node(2.5,0)
nodo3 = sp2.Node(x=5,y=0,th=0)

barra1 = sp2.Bar('barra 1', nodo1, nodo2, materialA, seccionA)
barra2 = sp2.Bar('barra 2', nodo2, nodo3, materialA, seccionA)



fuerza1 = sp2.Load('Fuerza 1', 'NOD', fx=10000, fy=10000)
nodo1.addLoad(fuerza1)

nodo2.setBC(sp2.H_MOVING)
nodo3.setBC(sp2.CLAMPED)

estructura= sp2.Structure([barra1,barra2])
estructura.setWeight('off')
estructura.show() #con esta función se muestra un gráfico con la
      estructura generada.


estatico = sp2.Static(estructura, run=True)
estatico.show()
for B in estatico.bars:
    print(B)

pandeo = sp2.Buckling( estatico, run = True,verModo=1)
pandeo.show()
print(pandeo)
```

4.10.2 Resultados devueltos por la consola

Lo primero que tiene lugar en el *script* descrito es la salida por pantalla del problema estático que se ha declarado, como se puede ver en la Figura 6, se trata de una viga simple con un apoyo móvil en la mitad de su longitud total y un empotramiento en el extremo final de la misma. Tiene aplicadas en su extremo libre una fuerza F_x y otra F_y de valor 10.000 N cada una. Este problema es un caso concreto con la finalidad de servir de ejemplo al usuario, habiéndose podido escoger cualquier otra estructura plana.

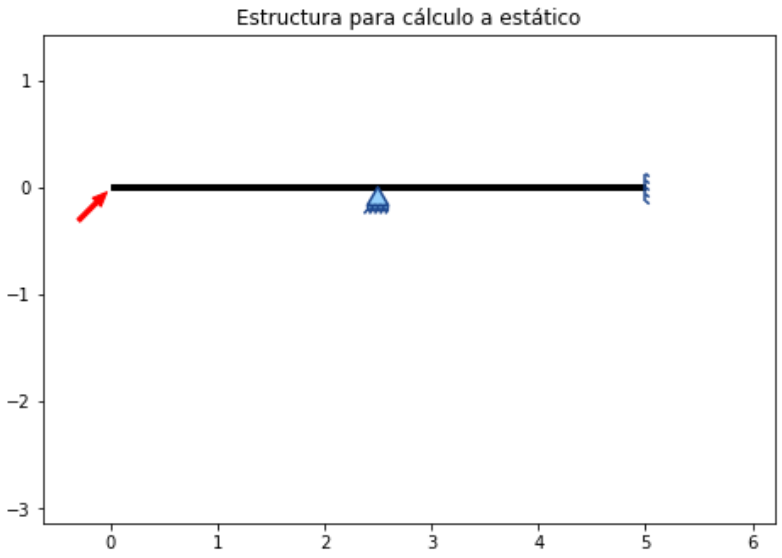


Figura 6. Caso estático del modelo de prueba.

Después de realizar este gráfico, el programa devuelve al usuario la solución por pantalla de los resultados de cada una de las barras de la estructura, tanto los desplazamientos en coordenadas locales como los esfuerzos internos de la barra para el caso estático. Esto se puede ver en la Tabla 2.

Solución asociada a la barra 1			
Desplazamientos locales:		**Esfuerzos internos:**	
u1 (mm)	0,225	**N1 (kN)**	10
v1 (mm)	253,8	**V1 (kN)**	10
th1 (rad)	-0,131	**M1 (kN.m)**	0
u2 (mm)	0,112	**N2 (kN)**	-10
v2 (mm)	0,000	**V2 (kN)**	-10
th2 (rad)	-0,044	**M2 (kN.m)**	25

Solución asociada a la barra 2			
Desplazamientos locales:		**Esfuerzos internos:**	
u1 (mm)	0,112	**N1 (kN)**	10
v1 (mm)	0,000	**V1 (kN)**	-15
th1 (rad)	-0,044	**M1 (kN.m)**	-25
u2 (mm)	0,000	**N2 (kN)**	-10
v2 (mm)	0,000	**V2 (kN)**	15
th2 (rad)	0,000	**M2 (kN.m)**	-12,5

Tabla 2. Resultados del caso estático del modelo de prueba.

A su vez también retorna el resultado de la deformada de la estructura a estático, la cual se puede ver en la Figura 7. La visualización de los resultados muestra en color negro la indeformada de la estructura y en color rojo la deformada obtenida para cada caso.

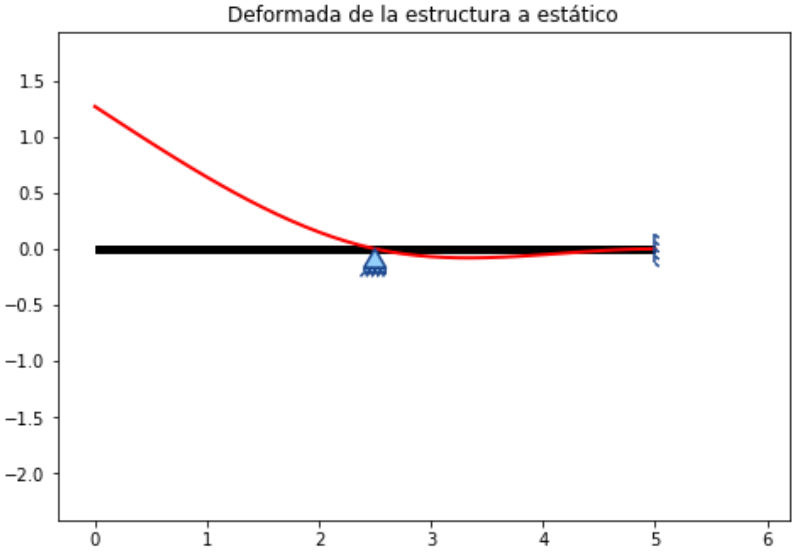


Figura 7. Deformada de la estructura para el caso estático.

Una vez obtenidos los resultados del cálculo estático de la estructura el programa ya puede resolver el problema de autovalores correspondiente al pandeo global de la estructura. El programa devuelve, en un primer lugar, una imagen con el modo de pandeo elegido por el usuario (el modo por defecto es 1), representando por un lado la indeformada de la estructura y por otro la deformada del modo de fallo en cuestión, indicando en el título de la gráfica el modo mostrado y el coeficiente de pandeo λ_p correspondiente Figura 8.

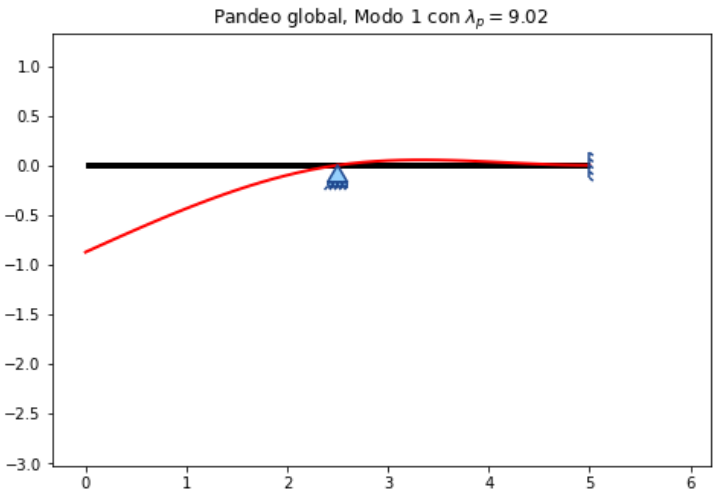


Figura 8. Deformada por pandeo global en el modelo de prueba.

A modo informativo y de forma similar al caso estático, el programa devuelve por pantalla el resto de los modos de fallo con sus correspondientes factores de carga como se puede ver en la Tabla 3.

Solución de pandeo global	
$\lambda_{p,1}$	9,02
$\lambda_{p,2}$	96,55
$\lambda_{p,3}$	285,14

Tabla 3. Resultados de pandeo global en el modelo de prueba.

El resto de los modos de fallo de la estructura se pueden visualizar por pantalla de una forma sencilla cambiando el parámetro “verModo=i” donde “i” es el modo de fallo (Figura 9).

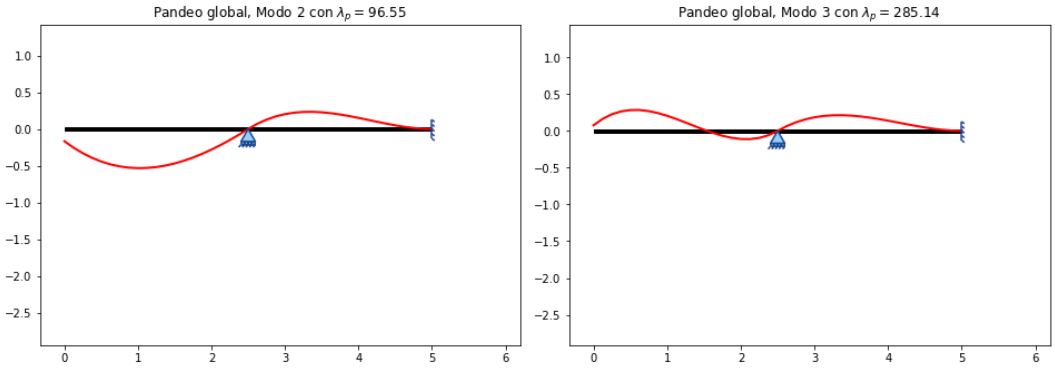


Figura 9. Diferentes modos de fallo del modelo de prueba.

5 Comparación entre StruPy2D y software comercial

Para poder comprobar los resultados que se obtienen mediante el programa realizado en Python, se ha utilizado el software comercial de cálculo estructural SAP2000, propiedad de Computers and Structures, INC., con licencia de estudiante en la versión v.22. Han sido establecidos seis problemas base para su validación; un análisis de esfuerzos en viga continua, pandeo de una barra aislada, el cálculo de pandeo global de una viga continua con cuatro casos de mallado diferentes, el cálculo de un pescante mediante dos mallados diferentes, un modelo pórtico a dos aguas con cargas distribuidas y el cálculo de un edificio residencial de 3 plantas con cubierta plana.

Con estos problemas se busca comprobar, por un lado, la tendencia del factor de pandeo frente al refinamiento de la malla, por otro, que los resultados son independientes de la posición del eje $\overline{OX}_{local}$ de cada barra y que los gráficos representados no dependen de la posición de las barras y por último que los resultados obtenidos con StruPy2D son los mismos a los que se obtienen mediante SAP2000.

En el caso de pandeo global hay que recordar que, como se mencionó anteriormente en la página 16, el método que se está utilizando para la resolución del cálculo de pandeo global converge a la solución real cuanto mayor es el número de elementos en los que se discretizan las barras de la estructura. Como este parámetro no ha sido introducido de manera que el software lo realice de forma automática, el usuario debe realizar a mano la discretización de la misma introduciendo nodos y barras más pequeños para construir el mallado.

En la realización de la mayor parte de los modelos se va a utilizar el mismo tipo de barra (salvo en el modelo de barra aislada y en el caso de edificio residencial). Esta barra será de acero S275 con las propiedades que se detallan en la Tabla 4. No se considerará el peso propio de la barra en los diferentes análisis.

Propiedades de las barras	
Material	Acero S275
Módulo de Young (Pa)	$2{,}1\text{x}10^{11}$
Densidad (kg/m³)	7850
Área (m²)	$20\text{x}10^{-3}$
Inercia (m⁴)	$220\text{x}10^{-6}$
Canto (m)	0,4

Tabla 4. Propiedades de las barras utilizadas para el análisis de los modelos.

El análisis de los diferentes modelos ha sido realizado para comparar los resultados retornados por ambos programas para el pandeo global de las diferentes estructuras, antes de ser mallados y después de haber realizado este mallado. Se ha comparado el factor de cargar a pandeo λ_p y los resultados gráficos de la deformada por pandeo global de los tres primeros factores de carga para cada caso, salvo que el modelo disponga de menos factores de carga. La selección de estos seis ejemplos concretos se debe a que son modelos significativos para su estudio debido a sus características.

5.1 Análisis de esfuerzos en una viga continua ante cargas estáticas

Es necesario para poder realizar correctamente los cálculos de pandeo global de los diferentes casos comprobar que los esfuerzos de la estructura están bien calculados mediante el software StruPy2D para el caso estático. Los esfuerzos axiles de cada barra son imprescindibles para la correcta obtención de la matriz de rigidez geométrica y su posterior resolución. Se ha procedido a comparar los esfuerzos obtenidos en un modelo barra en StruPy2D y en SAP2000 para constatar que estos coinciden, mediante un modelo como el que se puede ver en la Figura 10, similar al modelo barra que se utilizará en los casos de pandeo analizados de barras.

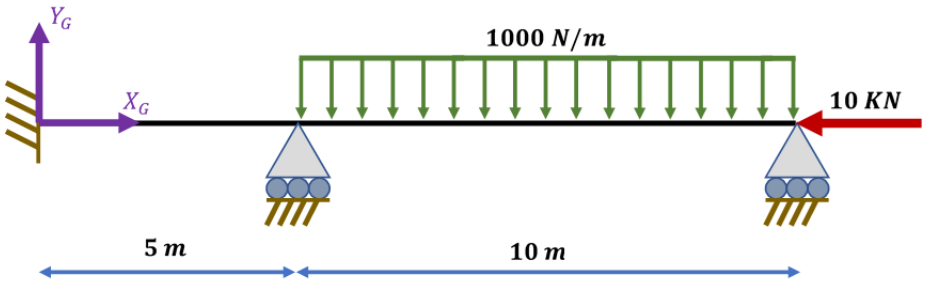


Figura 10. Modelo para la comprobación del cálculo a estático en StruPy2D.

Por las características del programa StruPy2D, no es posible calcular directamente el resultado de los esfuerzos en un punto intermedio de una barra. Para poder comparar con los resultados de SAP2000 se ha realizado una comparación de los valores en los nodos de cada problema. En la Figura 11 podemos ver la distribución de cortante axil cortante y flector a lo largo de las dos barras del problema con sus valores en los extremos obtenidos mediante el software de SAP2000.

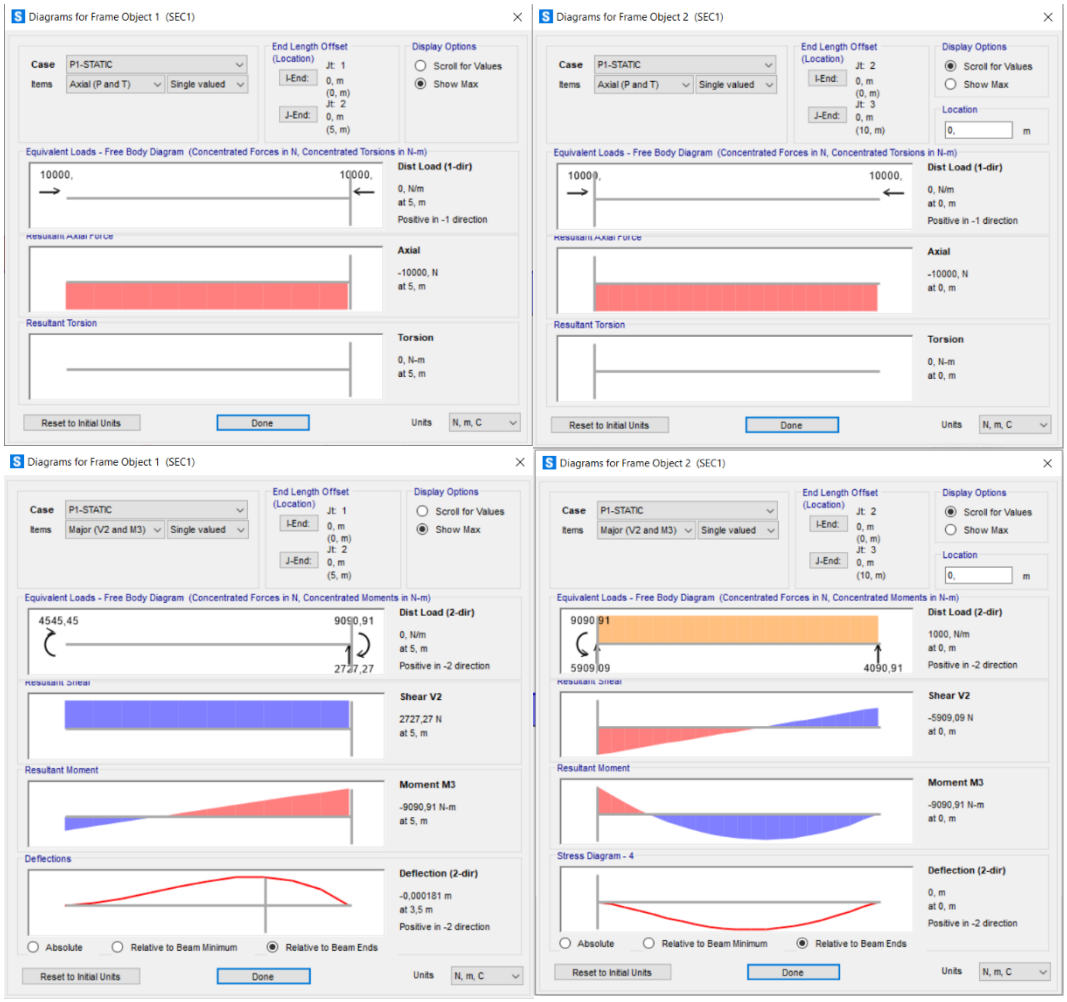


Figura 11. Resultados por cada barra para el cálculo estático en SAP2000.
A la izquierda se puede ver los resultados correspondientes a la barra con L=5m, a la derecha los correspondientes a la barra con L=10m.

Los resultados obtenidos por el programa StruPy2D para el cálculo a estático de la estructura son los que se pueden ver en la Tabla 5.

Solución asociada a la barra 1			
Desplazamientos locales:		**Esfuerzos internos:**	
u1 (mm)	0,000	**N1 (kN)**	10,0
v1 (mm)	0,000	**V1 (kN)**	-2,727
th1 (mrad)	0,000	**M1 (kN.m)**	-4,545
u2 (mm)	-0,012	**N2 (kN)**	-10,0
v2 (mm)	0,000	**V2 (kN)**	2,727
th2 (mrad)	-0,246	**M2 (kN.m)**	-9,091

Solución asociada a la barra 2			
Desplazamientos locales:		**Esfuerzos internos:**	
u1 (mm)	-0,012	**N1 (kN)**	10,0
v1 (mm)	0,000	**V1 (kN)**	5,909
th1 (mrad)	-0,246	**M1 (kN.m)**	9,091
u2 (mm)	-0,036	**N2 (kN)**	-10,0
v2 (mm)	0,000	**V2 (kN)**	4,091
th2 (mrad)	0,574	**M2 (kN.m)**	0,00

Tabla 5. Resultados del cálculo estático de una viga continua en StruPy2D.

Se puede observar que las cargas y los desplazamientos nodales en los dos problemas son iguales, no importando que programa se utilice de los dos, por lo que a partir de ahora se considerará que estos resultados son correctos por parte del programa StruPy2D, y se realizará solo la comparativa de los cálculos de pandeo entre los dos programas en cada uno de los siguientes modelos.

Debido a la implementación del cálculo de la estructura deformada en estático se va a proceder en este caso a su comparativa entre ambos programas. El resultado de estas puede verse en la Figura 12. Como puede apreciarse ambas geometrías de la deformada estructural son iguales, esto es debido a que SAP2000 también realiza la curva de desplazamientos interpolando un polinomio de grado $n = 3$.

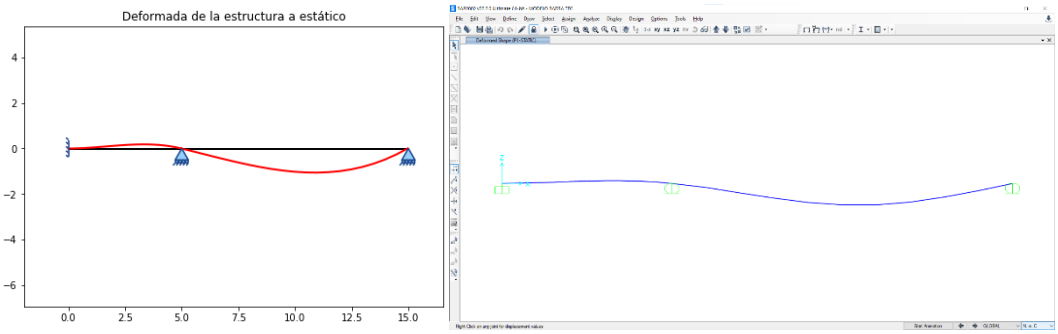


Figura 12. Deformada de una barra viga continua con carga estática.
A la izquierda la deformada correspondiente obtenida con StruPy2D, a la derecha obtenida con SAP2000.

A modo informativo, en la Tabla 6, se indican los resultados de pandeo global del modelo planteado para la comprobación del caso a estático.

Solución de pandeo global	
$\lambda_{p,1}$	1037,41
$\lambda_{p,2}$	3542,41

Tabla 6. Factores de carga en el modelo de prueba del cálculo estático.

5.2 Análisis de una barra aislada con valor teórico exacto

Se ha procedido a realizar el cálculo de una barra aislada para comprobar que el método de pandeo global aproximado funciona. Para ello se ha comprobado el resultado de fallo por pandeo de un modelo del que se conoce la solución exacta, para comprobar la convergencia del método aproximado al método exacto con el mallado. El modelo que se va a realizar es el que se puede ver en la Figura 13, el cual dispone de un valor β = 0,5 en la fórmula de Euler (Ecuación 6).

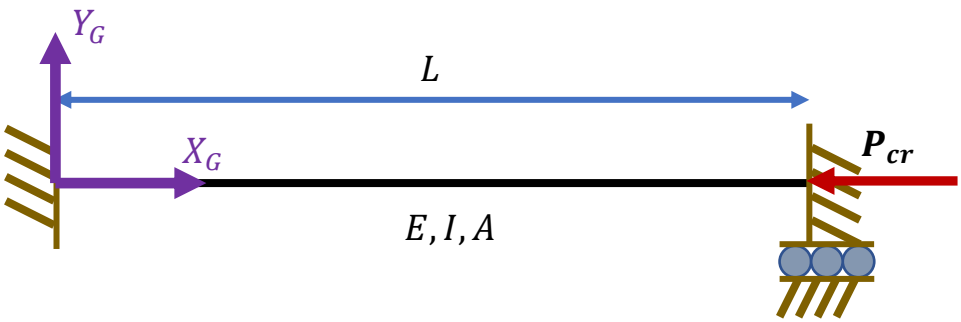


Figura 13. Análisis de una barra aislada con solución exacta de pandeo.

En el análisis se han considerado las siguientes propiedades de la barra; $A = 50x10^{-4}$ m², $I = 200x10^{-6}$ m⁴ , $E = 2{,}1x10^{11}$ Pa , L= 8 m, Con estas propiedades, según la fórmula de Euler la carga crítica teórica del problema tiene un valor de $P_{cr} = 25907711{,}55$ N.

El mallado del modelo se va a realizar con 2, 4 y 8 elementos. No se puede considerar el modelo con un único elemento en la barra, debido a que en el ejemplo considerado se anulan todos los grados de libertad del problema menos el desplazamiento en el eje axial de la barra, pero la matriz de rigidez geométrica en el elemento correspondiente a esa libertad queda con valor 0, quedando un sistema irresoluble para este caso. A continuación, en la Tabla 7, se muestran los resultados de carga crítica y coeficiente β obtenidos en cada uno de los casos mediante StruPy2D.

Número de elementos	P_{cr} (N)	β
2	26250000,00	0,4967
4	26102597,84	0,4981
8	25920979,94	0,4999

Tabla 7. Valores del coeficiente β para diferentes mallados.

Como se puede comprobar en la Figura 14, el valor β converge a la solución teórica real del problema analizado. Con esto se comprueba que el método utilizado funciona y que cuanto mayor sea el número de elementos en los que se malle más se acercará el resultado al valor real del factor de carga de pandeo global.

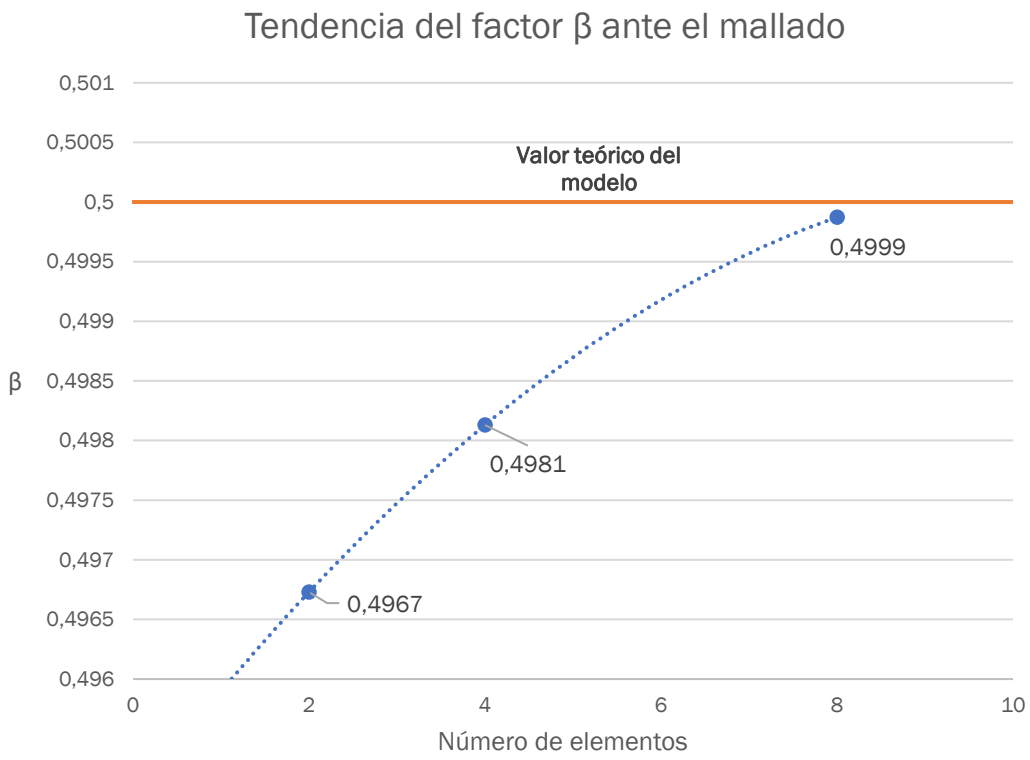


Figura 14. Convergencia del coeficiente β.

5.3 Análisis de una viga continua

El segundo caso de pandeo que se va a analizar es el correspondiente a la Figura 15. Se trata de una viga formada por dos vanos, empotrada en un extremo y con dos apoyos móviles en la dirección horizontal, con vanos de 5 m y 10 m cada uno respectivamente, la cual está cargada con 10.000 N a compresión en el extremo apoyado de la misma.

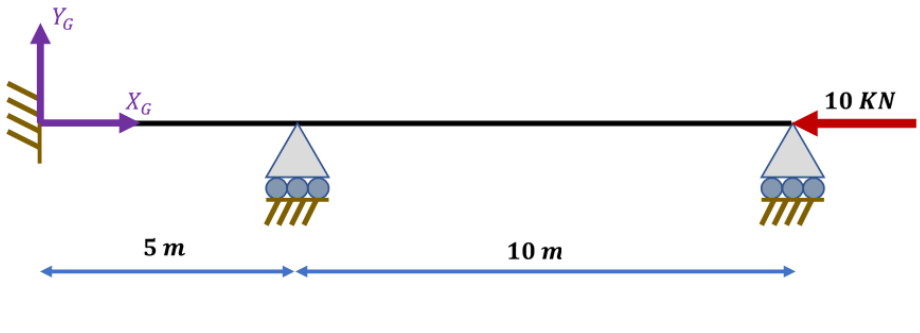


Figura 15. Modelo de viga continua para pandeo global.
A la izquierda se puede ver la figura correspondiente al programa StruPy2D, a la derecha el modelo realizado en el software SAP2000.

Es interesante la realización de este modelo de viga continua para comprobar la variación en el resultado debido al mallado de la misma. Considerando que los vanos de 5 m y 10 m son los vanos A y B respectivamente, se han realizado los siguientes ensayos dividiendo los vanos en los elementos indicados en cada caso.

5.3.1 Caso 1. A=1 B=1

En este caso se ha considerado que en el problema hay dos barras formadas cada una de ellas por un único elemento, buscando obtener un primer factor de aproximación del problema de pandeo.

Tanto SAP2000 como StruPy2D encontraron dos modos de pandeo, correspondientes a los dos grados de libertad de pandeo independientes que existen en la estructura, uno por cada giro de cada apoyo móvil. Los movimientos en horizontal de los apoyos móviles no generan ningún grado de libertad adicional a pandeo debido a las condiciones de contorno del problema. Véase que los valores obtenidos por SAP2000 son idénticos a los resultados conseguidos con StruPy2D como se puede ver en la Tabla 8.

Solución de pandeo StruPy2D		Solución de pandeo SAP2000	
$\lambda_{p,1}$	1037,414	$\lambda_{p,1}$	1037,414
$\lambda_{p,2}$	3542,412	$\lambda_{p,2}$	3542,412

Tabla 8. Resultados de pandeo global en viga continua, caso 1.

Obsérvese que, aunque el modelo de este caso no es idéntico al caso analizado en el apartado 5.1 (porque carece de la carga distribuida), el resto del problema hace que ambos modelos tengan el mismo axil y como las barras y las condiciones de contorno coinciden, el resultado de los coeficientes de pandeo λ_p sean los mismos (Tabla 6 y Tabla 8).

El software SAP2000 notificó dos avisos durante el analisis realizado; “No existen más modos de pandeo independientes por encontrar” y “Se encontraron menos modos de pandeo de los que se buscaron”. Esto se debe a que SAP2000 busca por defecto seis modos de fallo en cada modelo, por lo que arroja estos dos errores debido a que en este modelo solo encuentra dos modos de fallo, eso es de esperar que suceda al estar buscando el programa seis modos de fallo por lo que no se tendrá en cuenta este error que arroja el software. La representación gráfica de los resultados obtenida se puede observar en la Figura 16. En esta imagen se puede ver cómo las deformadas de ambos modelos se muestran con sentido contrario. Esto no supone ningún problema y es solo un cambio de signo en la resolución de la ecuación por lo que los resultados son correctos. Los resultados de la parte superior de la

imagen son los correspondientes al primer modo de fallo $\lambda_{p,1}$ y los que están en la parte inferior de la misma los correspondientes al modo 2 $\lambda_{p,2}$.

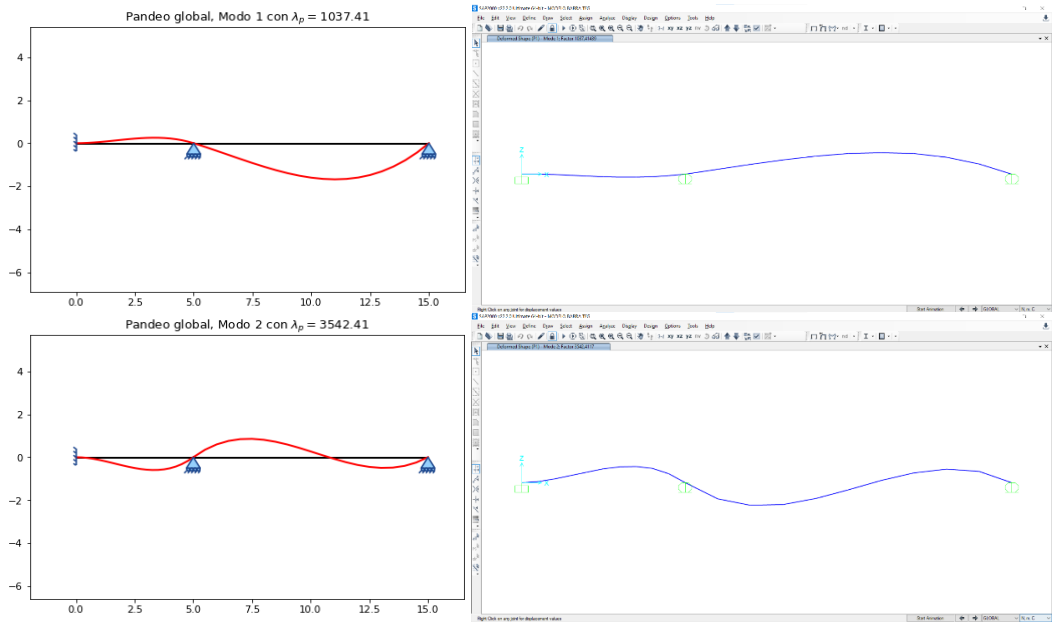


Figura 16. Deformaciones por pandeo global de la viga continua, caso 1.
A la izquierda de la imagen se pueden ver los resultados en StruPy2D, a la derecha los correspondientes al programa SAP2000.

5.3.2 Caso 2. A=1 B=2

En el caso 2 se ha realizado el mallado únicamente en la barra B, pasando a estar formada por dos elementos, haciendo que todos los elementos dispongan de la misma longitud. Esto hace que la estructura disponga de un total de cuatro modos de fallo, añadiendo dos grados de libertad adicionales en la confluencia de los dos elementos de la barra B, los correspondientes al desplazamiento vertical y al giro del nodo indicado. Los resultados obtenidos para este modelo se pueden ver en la Tabla 9.

Solución de pandeo StruPy2D		Solución de pandeo SAP2000	
$\lambda_{p,1}$	756,190	$\lambda_{p,1}$	756,190
$\lambda_{p,2}$	2664,929	$\lambda_{p,2}$	2664,929
$\lambda_{p,3}$	5832,589	$\lambda_{p,3}$	5832,589
$\lambda_{p,4}$	10140,356	$\lambda_{p,4}$	10140,356

Tabla 9. Resultados de pandeo global en viga continua, caso 2.

Solo para este caso se analizarán los 4 modos de fallo obtenidos para ver la diferencia en la formación del pandeo según el factor de carga. En el resto de los casos, cada vez estarán disponibles más valores para los factores de carga,

en función del número de elementos en los que discreticemos cada barra, debido a que el aumento del mallado añade grados de libertad para el pandeo de la estructura analizada. En la Figura 17 se pueden observar las deformaciones que adquiere el modelo barra para cada factor de carga.

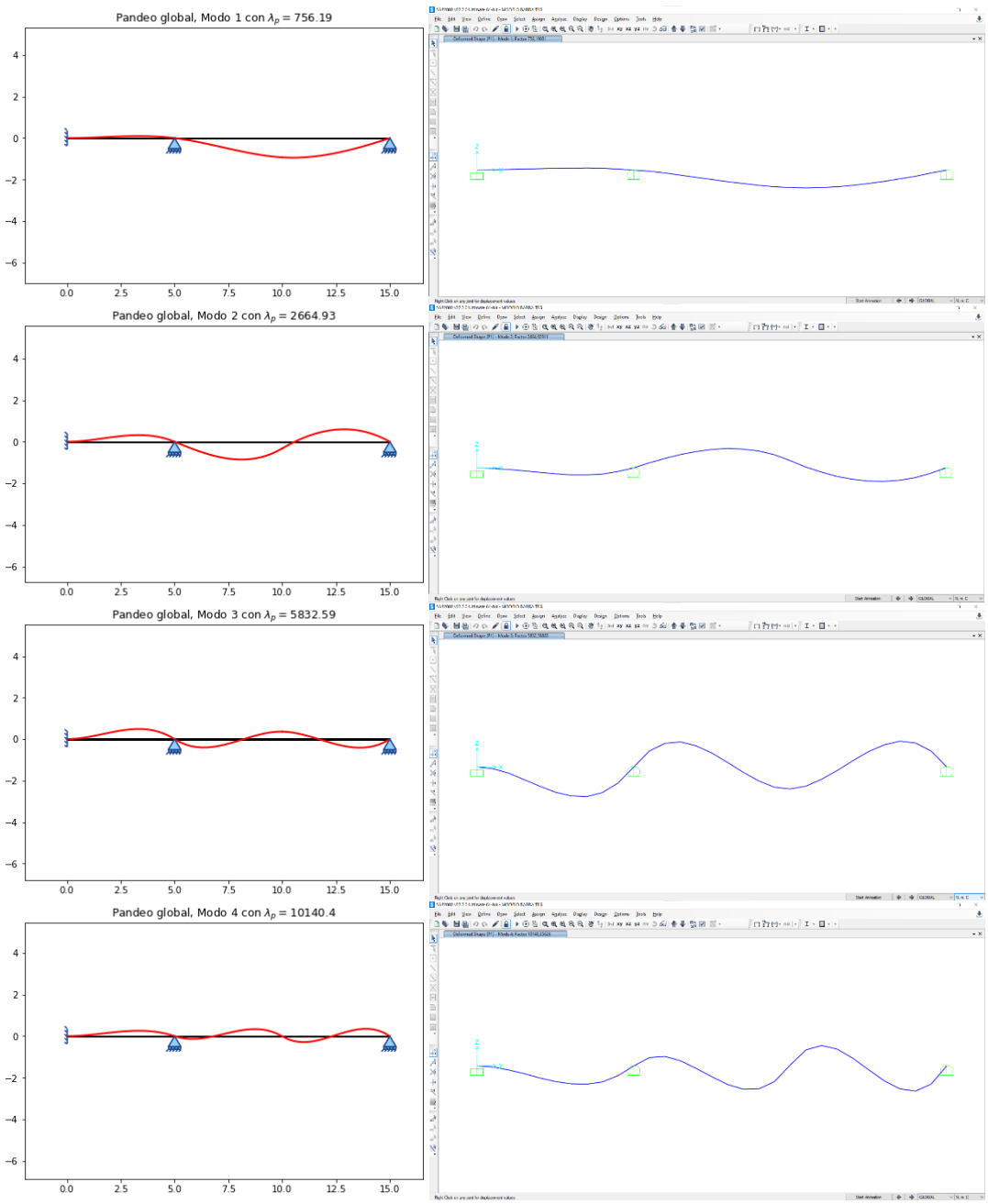


Figura 17. Modos de fallo en una viga continua, caso 2.

5.3.3 Caso 3. A=2 B=4

Al aumentar el número de elementos en los que se discretizan las barras el número de modos de fallo también aumenta de manera que, en este caso, con 6 elementos iguales, la barra dispone de 10 factores de carga diferentes. Aunque el valor más interesante es el correspondiente al primer modo de fallo se han obtenido los 3 primeros valores del factor de pandeo con el programa Strupy2D y con SAP2000 para comprobar los resultados, los cuales se pueden observar en la Tabla 10.

Solución de pandeo StruPy2D		Solución de pandeo SAP2000	
$\lambda_{p,1}$	746,481	$\lambda_{p,1}$	746,481
$\lambda_{p,2}$	2161,075	$\lambda_{p,2}$	2161,075
$\lambda_{p,3}$	4133,695	$\lambda_{p,3}$	4133,695

Tabla 10. Resultados de pandeo global en viga continua, caso 3

En este caso solo se procederá a la comprobación de la deformada del factor de carga $\lambda_{p,1}$=746,481 (Figura 18). Como era de esperar, la deformada de la estructura vuelve a coincidir como ya ocurría en los casos anteriores.

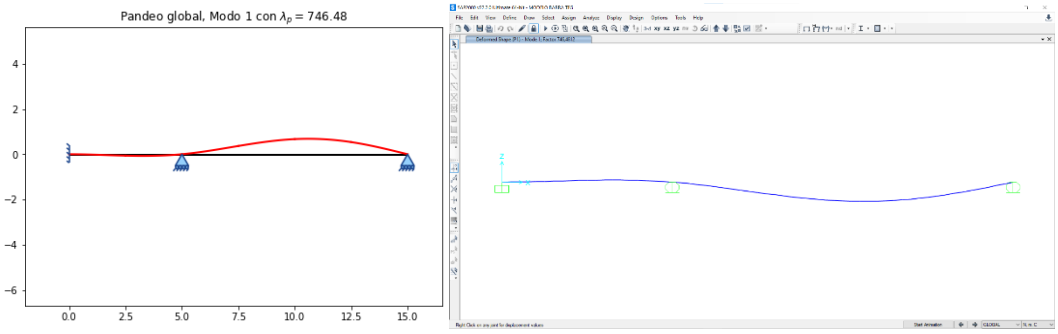


Figura 18. Modo 1 de fallo en viga continua, caso 3.

5.3.4 Caso 4. A=4 B=8

Para el último caso del modelo barra, se va a comprobar la tendencia del factor de pandeo de los 3 primeros factores de carga en función del refinado de la malla. Los resultados para el Caso 4. A=4 B=8, son los que se pueden observar en la Tabla 11.

Solución de pandeo StruPy2D		Solución de pandeo SAP2000	
$\lambda_{p,1}$	745,581	$\lambda_{p,1}$	745,581
$\lambda_{p,2}$	2141,518	$\lambda_{p,2}$	2141,518
$\lambda_{p,3}$	4017,659	$\lambda_{p,3}$	4017,659

Tabla 11. Resultados de pandeo global en viga continua, caso 4.

Con estos resultados y los obtenidos en el resto de los casos del modelo que se han realizado se ha hecho un análisis de los mismos para ver la tendencia de estos factores, el cual se puede ver en la Figura 19.

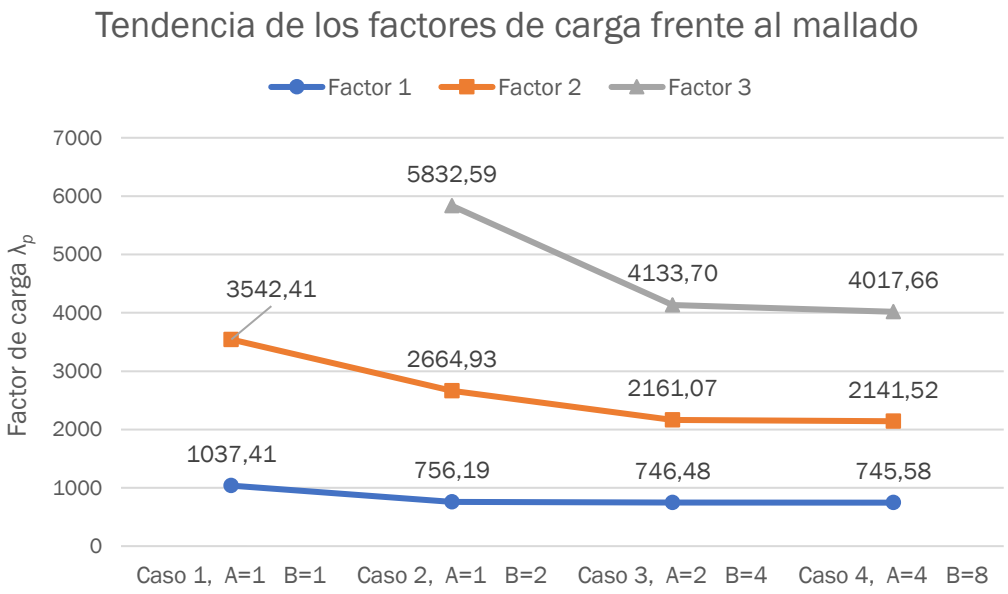


Figura 19. Factor de carga según el mallado en una viga continua.

Se puede observar que los valores de factor de carga tienden a converger con el refinamiento de la malla. Esto es especialmente llamativo en el caso 2, donde el hecho de haber dividido solo una de las barras redujo los factores de carga $\lambda_{p,1}$, $\lambda_{p,2}$ y $\lambda_{p,3}$ en un 27,11%, 24,77% y 29,12% respectivamente. En el resto de las iteraciones del mallado el valor converge de forma menos acusada, véase que entre el caso 2 y el caso 3 el factor $\lambda_{p,1}$ se reduce en un 1,28% y entre el caso 3 y el caso 4 tan solo un 0,12%. Con esto, se puede ver que no es necesario un mallado muy fino para obtener valores de cercanos al valor de convergencia.

5.4 Análisis de un pescante

Un pescante es una estructura constituida por una barra empotrada en uno de sus extremos unida con otra barra a noventa grados en el otro extremo. Este tipo de estructuras se suele utilizar para elevación de cargas en el extremo de la barra en voladizo. Un ejemplo de una estructura tipo pescante es la que se puede ver en la Figura 20. Se trata de una grúa en un semipórtico, este caso con empotramiento móvil en la base y apoyo móvil en el extremo de la viga.

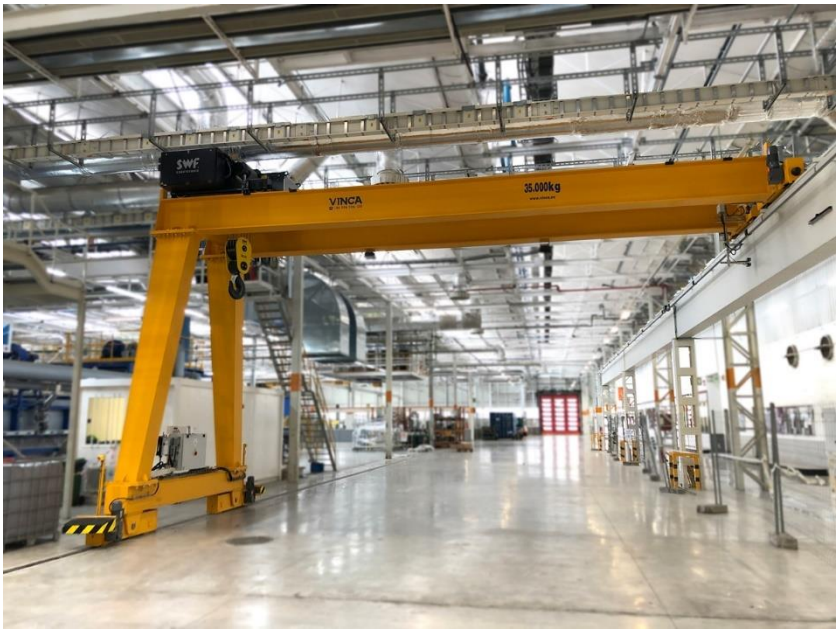


Figura 20. Ejemplo de pescante. [12]

En este caso se realizará el análisis de un pescante con empotramiento y apoyo fijo, el cual tendrá 4 metros de alto y 4 metros de ancho y estará cargado en el vértice de las dos barras con $F_x = F_y = -10000\,N$. Este modelo de pescante es el que se puede observar en la Figura 21. Hay que aclarar que, la imagen del modelo que se va a analizar no representa a la figura real del pescante que se comentó como ejemplo en la Figura 20.

Las barras del modelo tienen las características descritas en el apartado 5.3, disponibles en la Tabla 4. Propiedades de las barras utilizadas para el análisis de los modelos (página 47) y no se considerará el peso propio de las mismas.

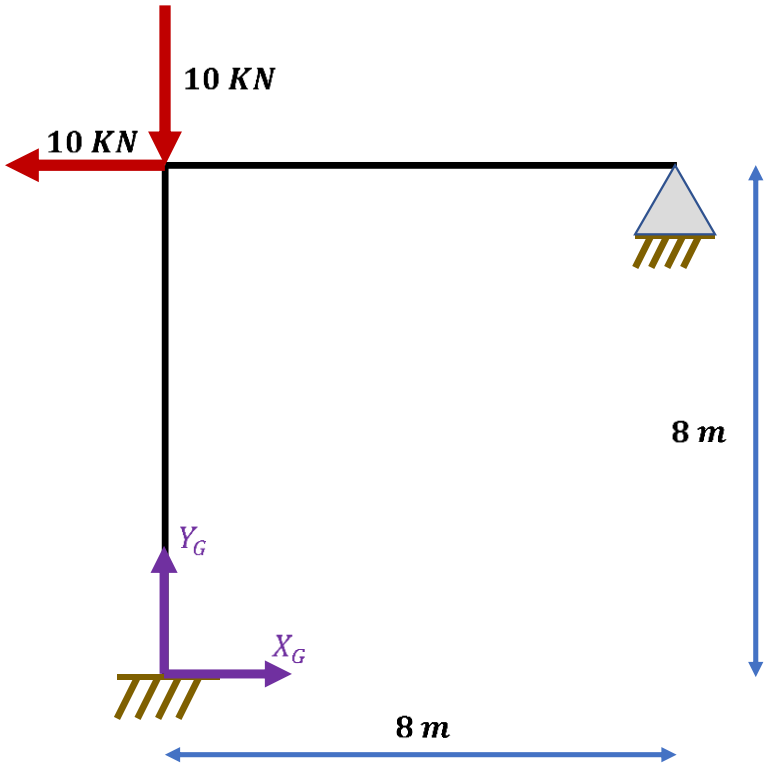


Figura 21. Modelo del pescante para el análisis de pandeo global.

Siendo la barra A la que se encuentra en posición vertical y la barra B la situada a 4 metros de altura en posición horizontal, serán analizadas mediante dos casos; el primero de ellos sin discretizar ninguna barra (A=1 B=1) y un segundo caso buscando un resultado del factor de pandeo más cercano al real mediante un mallado de 8 elementos por barra (A=8 B=8).

5.4.1 Caso 1. A=1 B=1

El análisis del pescante que se realizó devolvió los siguientes resultados (Tabla 12). Al considerarse el Caso 1. A=1 B=1, solo puede haber cuatro modos de fallo, correspondientes a los cuatro grados de libertad a pandeo de la estructura analizada, tres del nodo de confluencia de las dos barras y uno más correspondiente al giro en el nodo del apoyo fijo.

Solución de pandeo StruPy2D	
$\lambda_{p,1}$	-219610,378
$\lambda_{p,2}$	-6601,881
$\lambda_{p,3}$	146794,507
$\lambda_{p,4}$	760833,525

Solución de pandeo SAP2000	
$\lambda_{p,1}$	-219610,378
$\lambda_{p,2}$	-6601,881
$\lambda_{p,3}$	146794,507
$\lambda_{p,4}$	760833,525

Tabla 12. Resultados de pandeo global en pescante, caso 1.

Se puede observar que en este caso los programas están devolviendo al usuario valores del factor de carga negativos. Que el problema de autovalores del caso de pandeo global que se resuelve devuelva al usuario valores inferiores a cero es posible y además este hecho tiene una explicación física. Debido al estado de cargas actual una de las barras del problema analizado se encuentra a tracción (barra B) y la otra se encuentra a compresión (barra A). Si las cargas que se encuentran en el problema fuesen las mismas pero en sentido contrario, que es lo que indica la respuesta de signo negativo, la barra A pasaría a estar a tracción y la barra B sufriría compresiones haciendo posible de nuevo el pandeo de la estructura. Es conveniente darse cuenta de que esto no podría ocurrir en el modelo barra analizado en el apartado 5.3, debido a que un cambio en el sentido de la carga llevaba a que toda la viga estuviese traccionada, por lo que no podía tener lugar el pandeo de la estructura para factores de carga negativos.

A continuación, se puede ver la solución numérica de los factores de carga positivos (Tabla 13), que como se indicó anteriormente, a nivel de cálculo son los que interesan en el análisis estructural.

Solución de pandeo para StruPy2D		Solución de pandeo para SAP2000	
$\boldsymbol{\lambda_{p,1}}$	146794,507	$\boldsymbol{\lambda_{p,1}}$	146794,507
$\boldsymbol{\lambda_{p,2}}$	760833,525	$\boldsymbol{\lambda_{p,2}}$	760833,525

Tabla 13. Resultados de pandeo global en pescante con $\lambda_p > 0$, caso 1.

Aunque en algunas situaciones pueda ser conveniente conocer los factores de pandeo negativos, lo habitual es que el estado de cargas que se proporcione a la estructura sea la que va a sufrir esta en su vida en servicio y no sea necesario tener en cuenta los factores de carga negativos debido a que estos no deberían de poder darse en la estructura y por eso no han sido considerados.

En la Figura 22 se puede ver la comparación de los modos de fallo, en cuanto a deformación de la estructura se refiere con factor de carga positivo $(\lambda_p > 0)$.

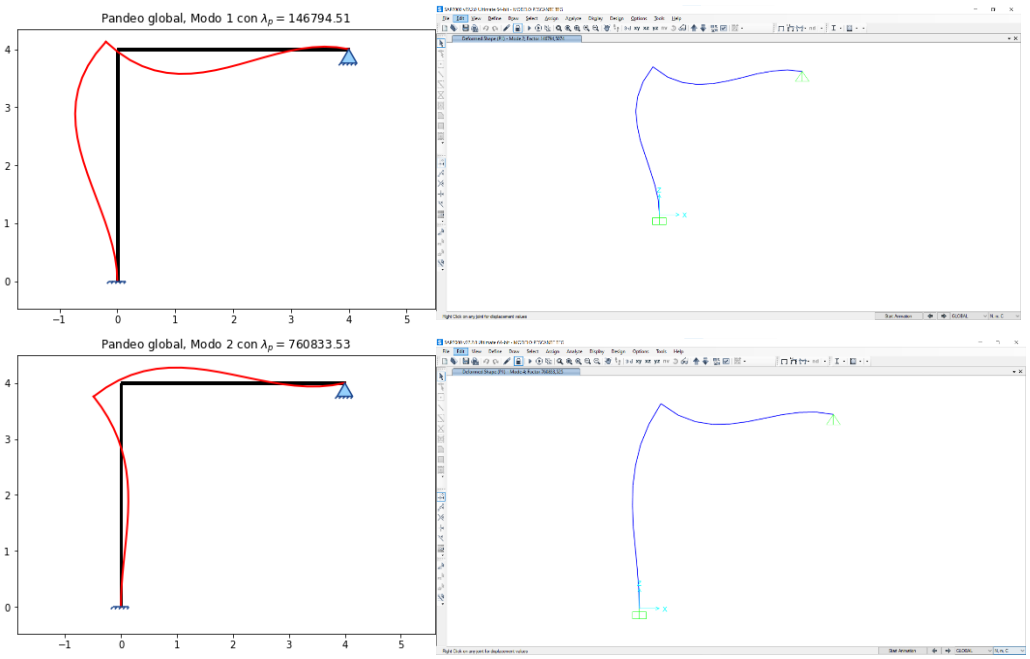


Figura 22. Deformación por pandeo global en el modelo pescante, caso 1.

5.4.2 Caso 2. A=8 B=8

Aunque el valor calculado para el Caso 1. A=1 B=1 puede ser de interés debido a la obtención de un factor de carga de referencia, es conveniente siempre realizar el mallado de la estructura para la obtención de factores más cercanos al valor real. Para poder obtener este dato se ha procedido al analisis de las barras formadas por 8 elementos cada una, obteniéndose los datos que se observan en la Tabla 14. Se han seleccionado los tres primeros datos de factor de carga $\lambda_p > 0$ y no se han considerado los factores de carga de pandeo con valores negativos.

Solución de pandeo StruPy2D		Solución de pandeo SAP2000	
$\lambda_{p,1}$	9001,451	$\lambda_{p,1}$	9001,451
$\lambda_{p,2}$	20108,790	$\lambda_{p,2}$	20108,790
$\lambda_{p,3}$	40511,522	$\lambda_{p,3}$	40511,522

Tabla 14. Resultados de pandeo global en pescante, caso 2.

En la Figura 23 se puede ver la deformada el primer modo de fallo con factor de carga positivo del problema ($\lambda_{p,1}$= 9001,451). Se puede observar que la deformada de la estructura cambia bastante con respecto al modelo sin mallar del apartado 5.4.1.

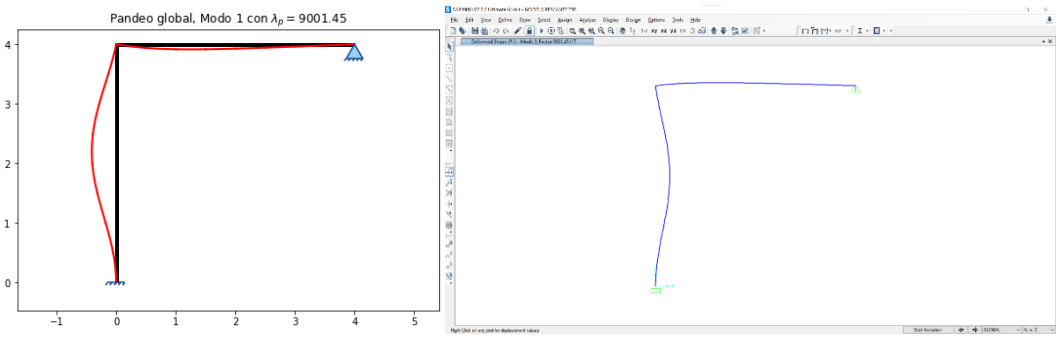


Figura 23. Modo de fallo 1 en el modelo pescante con mallado de barras.

Comparando los resultados del modelo obtenidos en este caso con los del Caso 1. A=1 B=1), podemos ver el interés del mallado de los modelos para el cálculo de pandeo de las estructuras (Tabla 15).

Solución de pandeo caso 1, (A=1, B=1)		Solución de pandeo caso 2, (A=8, B=8)	
$\lambda_{p,1}$	146794,507	$\lambda_{p,1}$	9001,451
$\lambda_{p,2}$	760833,525	$\lambda_{p,2}$	20108,790

Tabla 15. Comparativa del factor de carga en pescante entre los casos 1 y 2.

Como se puede apreciar a simple vista en los resultados, el refinado de la malla ha devuelvo unos factores de carga mucho menores, siendo un 93,87% inferior el valor del factor de carga λ_{p1} en el Caso 2. A=8 B=8, y siendo también menor el factor λ_{p2} en el caso 2 al primer modo de fallo del caso sin mallar, reflejando el peligro de la realización del cálculo de pandeo global sin refinar las barras en los suficientes elementos.

5.5 Análisis de un pórtico global

El pórtico es una de las estructuras más utilizadas en construcción. Habitual en la mayoría de las naves industriales, es un tipo de estructura de alto interés en cuanto a su cálculo se refiere. En la Figura 24 podemos ver un ejemplo de cómo puede ser el esqueleto principal de una nave industrial común, donde observamos en rojo uno de los seis pórticos que componen la estructura principal de la misma, los cuales sostendrán las cubiertas y los cerramientos de esta.

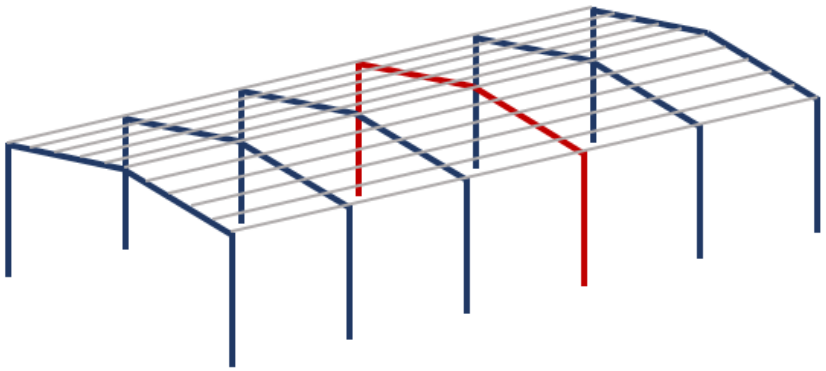


Figura 24. Estructura de una nave industrial.

El pórtico que se va a analizar en el siguiente ejemplo consta de dos barras en dirección vertical, ambas empotradas en su base y separadas una determinada distancia las cuales soportan una viga, una celosía u otra estructura a las que se anclarán las correas donde se sujetará la cubierta.

Para la realización del cálculo a pandeo global de un pórtico industrial se ha considerado una estructura como la que se puede ver en la Figura 25.

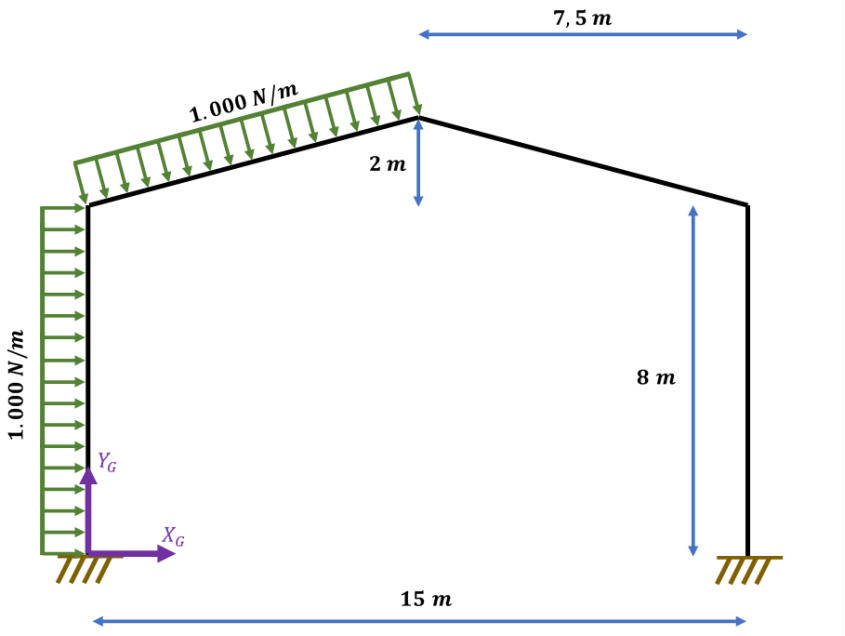


Figura 25. Modelo de pórtico plano para pandeo global.

En esta estructura se aplican cargas simulando el viento lateral en uno de los lados de la nave las cuales tiene un valor de 1000 N/m. Las patas del pórtico están empotradas al terreno y este tiene 15 metros de ancho con 8 metros de altura de las columnas y unas vigas dispuestas para formar una cubierta a dos aguas simétrica con una elevación en el vértice central de 2 metros con respecto a la altura de las columnas que forman el pórtico. Las barras tendrán las mismas características que las descritas en el apartado 5.3, disponibles en la Tabla 4. Propiedades de las barras utilizadas para el análisis de los modelos (página 47) y, como en los apartados anteriores, no se considerará el peso propio de las mismas.

Este problema es interesante tanto por sus características a nivel industrial como para realizar la comprobación del correcto funcionamiento de los sistemas del programa en cuanto a cambios de ejes se refiere, al componerse la estructura de ángulos diferentes a ±90°. En la Figura 26 podemos ver el pórtico ya definido en los dos programas con los que se ha realizado el cálculo.

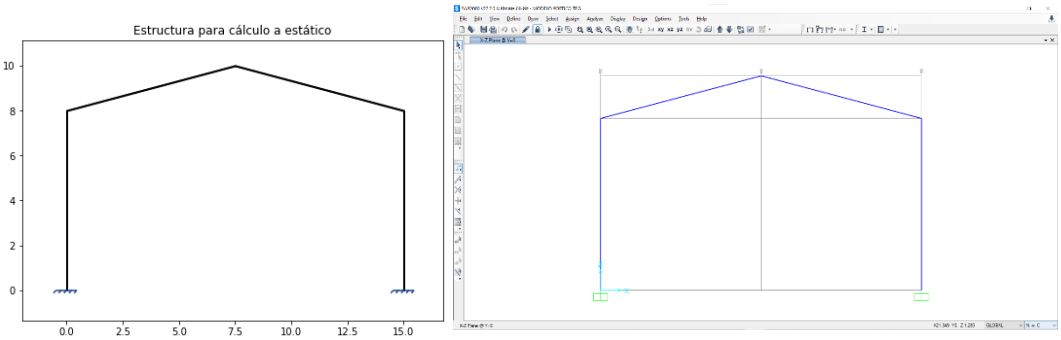


Figura 26. Modelo pórtico en los programas de cálculo. A la izquierda, la estructura representada por StruPy2D, a la derecha representada mediante SAP2000.

Analizado el pórtico del problema tanto con SAP2000 como con StruPy2D los resultados que se obtuvieron del mismo son los que pueden verse en la Tabla 16. Se puede comprobar de nuevo que los resultados del mismo no dependen del programa y que ambos programas realizan el cálculo correctamente.

Solución de pandeo StruPy2D	
$\lambda_{p,1}$	1020,021
$\lambda_{p,2}$	1687,278
$\lambda_{p,3}$	3864,379
$\lambda_{p,4}$	6284,777
$\lambda_{p,5}$	10455,909
$\lambda_{p,6}$	973008,492

Solución de pandeo SAP2000	
$\lambda_{p,1}$	1020,021
$\lambda_{p,2}$	1687,278
$\lambda_{p,3}$	3864,379
$\lambda_{p,4}$	6284,777
$\lambda_{p,5}$	10455,909
$\lambda_{p,6}$	973008,492

Tabla 16. Resultados de pandeo global en pórtico.

A continuación se muestra, en la Figura 27, la comparativa gráfica a pandeo global del pórtico 2D con los tres primeros modos de fallo obtenida tanto en StruPy2D como en SAP2000. Hay que recordar lo explicado en el apartado 5.3.1, que las representaciones de ambos programas tengan cambiado el signo no supone problema alguno, por lo que se considera que la representación gráfica de los diferentes modos de fallo coincide para los tres primeros factores de carga λ_p aunque con el signo cambiado.

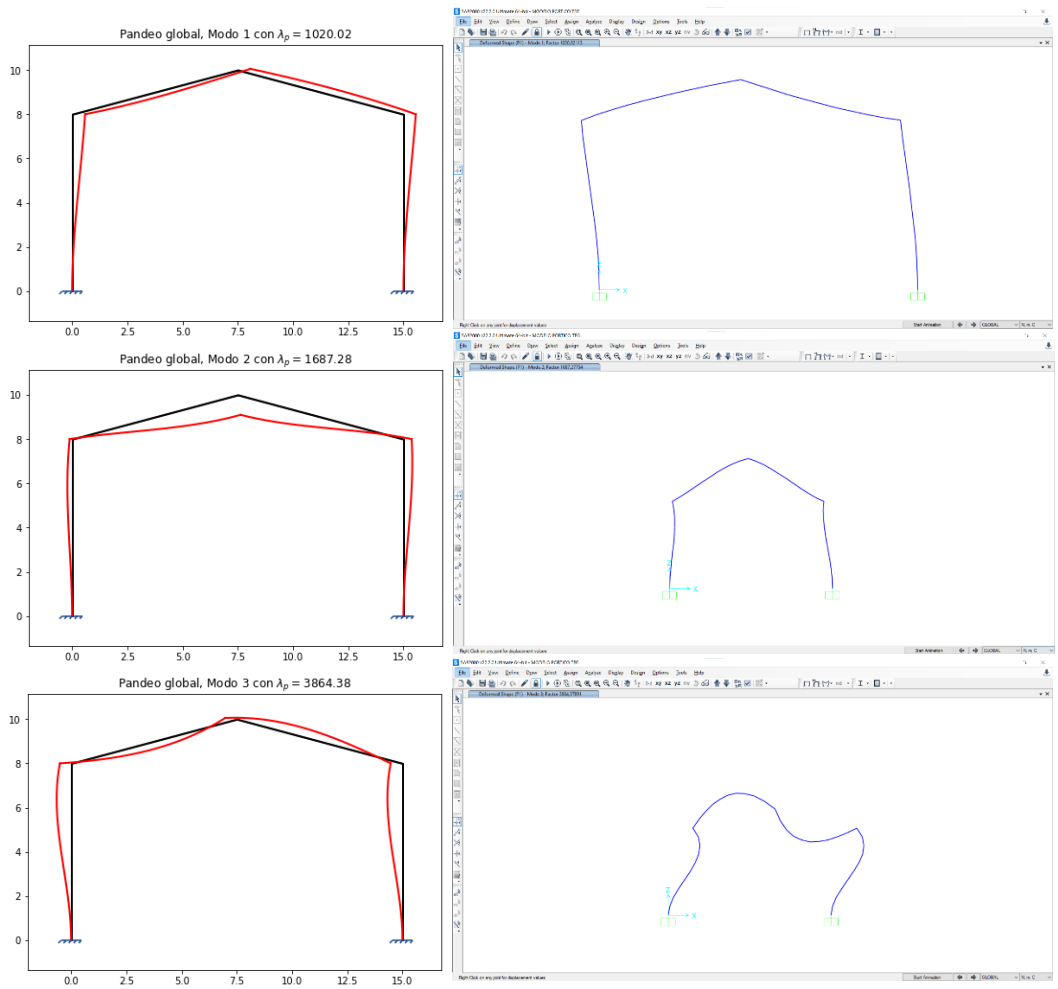


Figura 27. Modos de fallo por pandeo global del modelo pórtico.

5.6 Análisis de una estructura de uso residencial

En el campo de la ingeniería estructural, uno de los problemas habituales es el diseño de edificaciones para el uso residencial, las cuales sufren grandes cargas a compresión en las columnas que soportan los voladizos de cada planta, siendo mayores estas cargas cuanto más cerca del suelo se encuentran. Un ejemplo de estructura residencial puede ser la que se observa en la Figura 28.

Figura 28. Estructura de hormigón. [13]. Licencia CC-BY.

La optimización de este tipo de estructuras es de gran interés, tanto a nivel económico como a nivel medioambiental. Los materiales utilizados habitualmente en estas construcciones son de difícil recuperación y reciclado. Es por ello el gran interés en reducir los materiales necesarios, acelerando el proceso constructivo al requerirse menores cantidades y reduciendo los residuos finales al final de la vida útil de la misma.

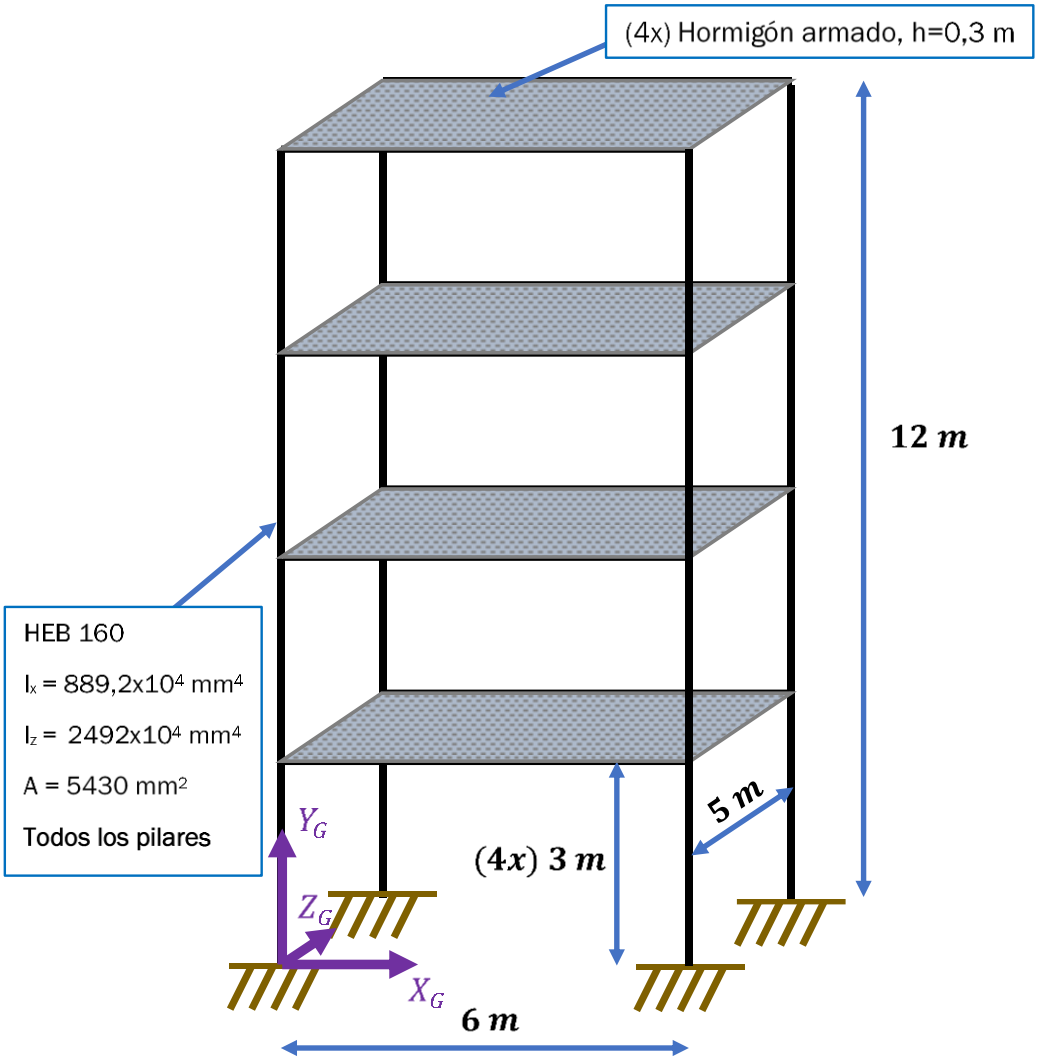


Figura 29. Modelo de estructura de un edificio residencial en 3D.

Para poder generar un modelo de interés para su análisis se va a hacer una simplificación de un modelo de tres plantas en 3D con cubierta plana por un modelo en 2D del mismo. La estructura que se va a simplificar es la que se puede ver en la Figura 29. Se han introducido las siguientes simplificaciones en el modelo:

- **Valor de la inercia de cada pilar** $= 2xI = 4984\text{x}10^{-12}\text{m}^4$
- **Valor del área de cada pilar** $= 2xA = 5430\text{x}10^{-6}\text{m}^2$
- **Carga añadida al eliminar la tercera dimensión del hormigón armado**
  $= \rho \cdot h \cdot b \cdot g = 2500kg/m^3 \cdot 0{,}3m \cdot 5m \cdot 9{,}8N/kg = 36750N/m$

Adicionalmente se ha considerado la inclusión de la sobrecarga de uso correspondiente según el código técnico de la edificación (CTE) para viviendas [14], aplicando una carga uniforme 2 kN/m^2, que en el modelo simplificado pasa a ser de valor 10 kN/m al reducir la tercera dimensión.

Con estas modificaciones, la carga distribuida que se considerará en el problema es una carga total de **46750 N/m** en cada forjado de hormigón armado. No se van a considerar fuerzas de viento laterales y se considera que en la dirección $\overline{OZ}$ el desplazamiento de la estructura se encuentra restringido. Se han orientado los pilares de forma que la componente de inercia mayor de estos coincide con el eje $\overline{OZ}$. El problema simplificado al completo que se va a considerar se puede ver en la Figura 30.

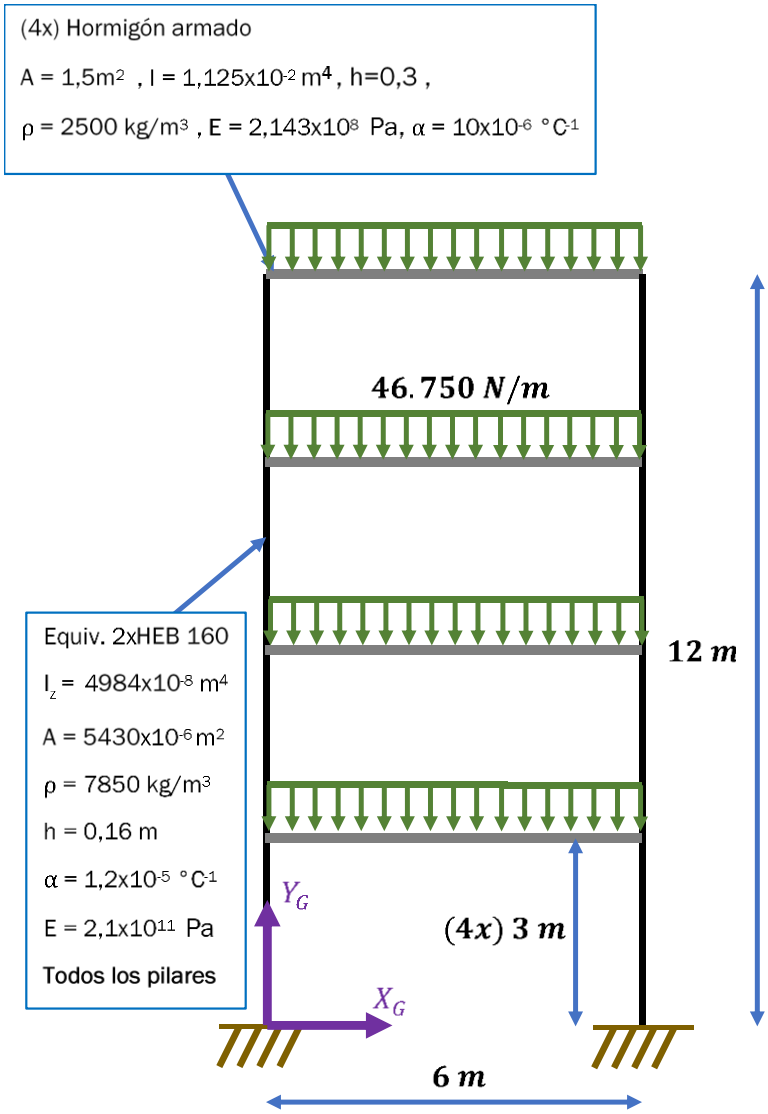


Figura 30. Modelo de estructura de un edificio residencial en 2D para análisis.

Los resultados para los factores de carga de los tres primeros modos de fallo que se han obtenido en los programas con los que se ha realizado el análisis los siguientes (Tabla 17).

Solución de pandeo StruPy2D		Solución de pandeo SAP2000	
$\lambda_{p,1}$	3,573	$\lambda_{p,1}$	3,573
$\lambda_{p,2}$	8,637	$\lambda_{p,2}$	8,637
$\lambda_{p,3}$	16,697	$\lambda_{p,3}$	16,697

Tabla 17. Resultados de pandeo global en una estructura de uso residencial.

De nuevo se puede observar la coincidencia de resultados entre los dos modelos calculados por ambos programas. Las deformadas de los diferentes modos de fallo por pandeo global del problema analizado (Figura 31) se puede ver que también son idénticas entre ambos programas, pudiendo dar por correcto el resultado obtenido mediante StruPy2D.

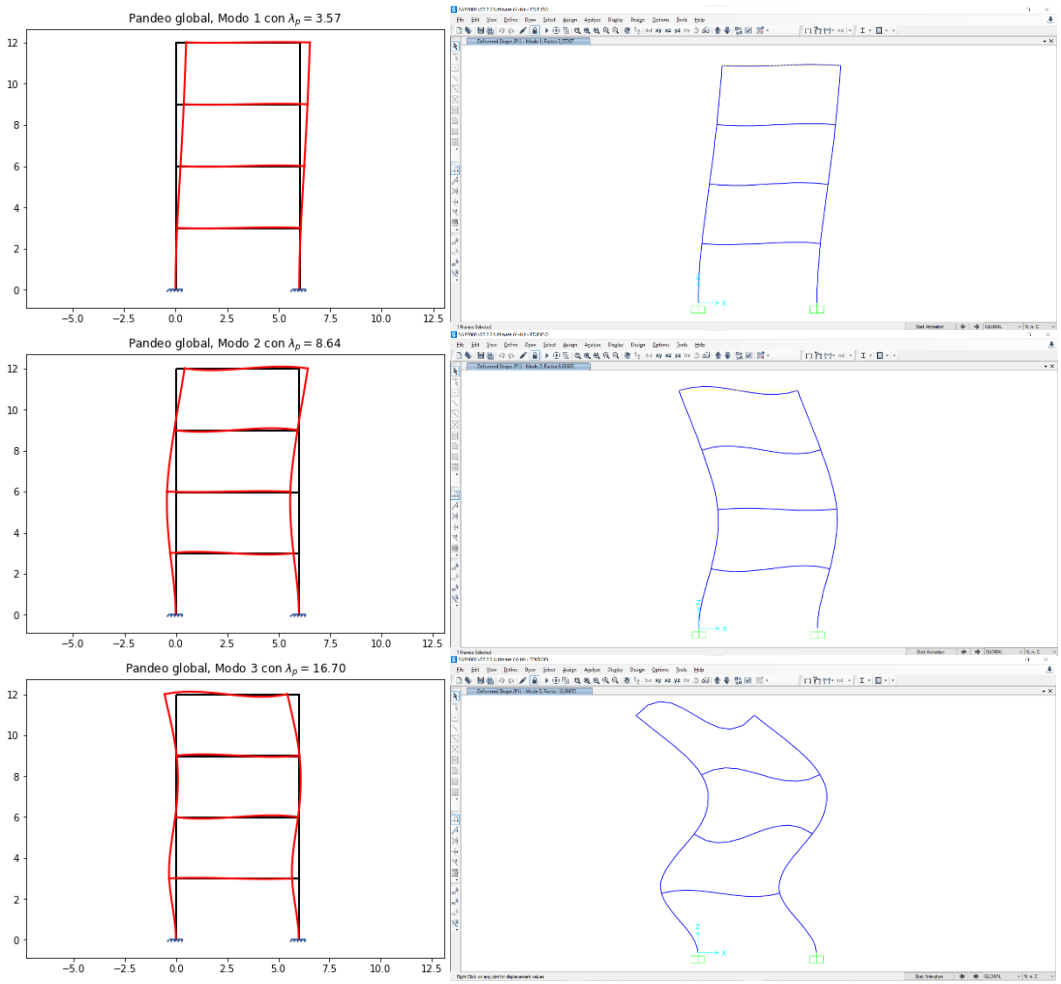


Figura 31. Representación de los modos de fallo del modelo estructura.

A método informativo hay que indicar que, en este tipo de estructura, aumentar el área y la inercia en las secciones de las columnas de los pisos inferiores y hacer más esbeltas las de los pisos superiores ayuda sustancialmente a la resistencia a pandeo de la estructura. Por un lado, por la mejora directa de las propiedades de esta al pandeo global mediante la mejora de las capacidades resistivas de la estructura y por otra parte por la reducción de peso de las columnas menos solicitadas a compresión, haciendo que las que se encuentran en pisos inferiores soporten a su vez esfuerzos a compresión menores al no soportar el peso adicional. En este problema no podemos realizar esta comprobación completamente debido a que no se está considerando el peso propio de las barras y se ha ejecutado el análisis sin considerar este debido a que no está implementado para su ejecución en el caso de pandeo global.

6 Conclusiones y líneas de trabajo futuras

6.1 Conclusiones sobre los resultados obtenidos

Se han podido cumplir todos los objetivos planteados inicialmente para la realización de este TFG, concretamente:

- **Programación de un software para la realización de cálculos de pandeo global de estructuras planas.** Implementado en el *script* Buckling.py, que contiene los elementos necesarios para su resolución.

- **Integración del programa para la obtención de los factores de pandeo global con el programa ya existente de cálculo de estructuras para cargas estáticas.** Mediante la realización de las modificaciones pertinentes en los archivos afectados.

- **Obtención de la deformada de los diferentes modos de fallo en el caso de pandeo global y visualización del modo de fallo la estructura.**

- **Obtención de la deformada estructural para cargas estáticas sin pandeo de la estructura y visualización de la misma.**

- **Profundizar en aspectos computacionales adquiridos en la asignatura de Estructuras y Construcciones Industriales.**

- **Aplicar y ampliar conocimientos de Python y cálculo mediante ordenador a un problema concreto de interés en ingeniería.** Tras la realización de programación orientada a objetos, POO, favoreciendo la segmentación del código mediante el uso de clases, métodos y atributos.

- **Documentar el código desarrollado para su futura consulta por otros alumnos y la posible ampliación de las capacidades del mismo, en un proceso de mejora continua.**

- **Mejora de las capacidades de razonamiento crítico, analisis, síntesis y trabajo autónomo.**

El cálculo a pandeo global de estructuras es necesario para la obtención de elementos estructurales seguros en los que se pueda garantizar que los elementos constructivos no sobrepasen los estados límite últimos. El método que se utiliza en el cálculo de estas estructuras, aunque se trata de un método aproximado, devuelve valores muy cercanos a los valores reales del problema, más cercanos a la solución real cuanto más fino se realice el mallado de la estructura.

Los resultados obtenidos en los diferentes modelos que se han comprobado son coherentes y tienen sencilla interpretación, debido a su vez a la sencillez de las estructuras aquí representadas. Es conveniente recordar que, aunque el método de pandeo global es uno de los cálculos más interesantes de realizar en una estructura, estas pueden tener formas de fallar adicionales no contempladas en StruPy2D, sirviendo este actualmente más como un complemento para la obtención de datos orientativos que para la resolución de estructuras completas siguiendo el código técnico de edificación (CTE).

En cuanto al código del programa, este permite modificaciones a gusto del usuario para poder obtener resultados a demanda personalizados, pudiendo ser personalizado por cada usuario para sus propios intereses con unos conocimientos mínimos en programación. Además, es interesante ver que el coste de almacenamiento de este tipo de programas es mucho menor. Todo el programa StruPy2D tiene un tamaño inferior a 1 MB , y el programa con el que se han comprobado los resultados, SAP2000, ocupa en el disco duro del ordenador 1,29 GB. Adicionalmente los modelos en SAP2000 generados para las comprobaciones generan archivos mucho mayores. Todos los modelos creados para la comprobación del funcionamiento utilizados han ocupado 9,75 KB en código Python y los generados en SAP2000 ocupan 5,24 MB, por lo que hablamos de diferencias de tamaño de entre 500 y más de 1000 veces de diferencia entre el programa StruPy2D y SAP2000.

Aunque los modelos no son archivos pesados con las disponibilidades de memoria de los ordenadores actuales, hay que aclarar que son modelos muy pequeños en comparación a lo que pueden ser los modelos de construcciones reales que se realizan en grandes proyectos.

El tiempo de desarrollo de una estructura sencilla en StruPy2D es bastante corto, requiriendo de un tiempo relativamente corto si el análisis de la misma se va a realizar con barras sin discretizar. Con el programa actual si se realiza la discretización de las barras el tiempo de desarrollo se incrementa considerablemente si se compara con el desarrollo de la estructura en SAP2000, el cual permite tanto el mallado manual como el mallado automático de forma mucho más rápida que con StruPy2D.

Este programa puede ser muy interesante a día de hoy para su uso en el ámbito educativo, pudiendo facilitar la comprobación de problemas simples en 2D de manera sencilla para los estudiantes de las asignaturas relacionadas con el cálculo estructural.

6.2 Líneas de desarrollo futuro

Como líneas futuras en las que se pueda desarrollar el programa de cara a su mejora continua, la opción que se puede considerar más interesante para el cálculo de pandeo global es la implementación del mallado estructural de forma automática, haciendo que el mallado de la estructura sea refinado por el programa hasta que se considere que el resultado ha convergido mediante un criterio adecuado.

En el código actual los problemas de pandeo con peso propio de las barras tienen un gran margen de error, debido a que en estos problemas no se considera que el axil pueda varía a lo largo de toda la barra, por lo que para poder obtener un problema con peso propio que tenga un resultado óptimo es necesario realizar un gran mallado para aproximar el valor del peso de manera menos errática por cada uno de los tramos.

La adaptación del programa a una entrada de datos mediante una interfaz gráfica acercaría el programa a muchos más usuarios. La utilización del programa actual requiere unos conocimientos mínimos en cuanto a programación en general y al lenguaje Python en particular, aunque la interpretación de los programas sea sencilla debido a la similitud de muchos de los códigos a la lengua inglesa. La introducción de nodos, barras, cargas y condiciones de contorno mediante la solicitud por parte del programa facilitaría su utilización a usuarios sin conocimientos técnicos de programación.

También reseñar que, la mayor parte de los usuarios a los que les puede interesar la utilización de este tipo de programas dispondrán de las nociones de programación necesarias y solo necesitarán la inversión del tiempo correspondiente al conocimiento del lenguaje de Python si no disponen ya de este.

Otra línea de mejora es la implementación de gráficos en la resolución para representar los diagramas correspondientes a los axiles, cortantes y momentos flectores de las barras en las estructuras analizadas. Actualmente el programa no realiza su representación en los cálculos y es necesario por parte del usuario comprender que los máximos de los esfuerzos que sufre la estructura del problema pueden no darse en los nodos del mismo pudiendo encontrarse en un punto intermedio de las barras del problema. El usuario será quien deba

considerar en que puntos se puede dar esta situación y comprobar que no se sobrepasan los límites de los materiales utilizados en la estructura en ningún punto.

Por último, se puede considerar implementar el cálculo estructural tanto para casos estáticos como para pandeo de las estructuras en 3D, intentando que este pueda llegar a realizar el cálculo de estructuras completas sin importar como sean estas. aunque este tipo de mejora lleva un gran número de cambios y de implementaciones adicionales que no se tienen en consideración para el análisis de estructuras planas.

6.3 Consideraciones adicionales

Por las características intrínsecas a este TFG, el impacto medioambiental del mismo ha sido mínimo. No siendo necesaria la compra de ningún material adicional al disponible por parte del alumno. La mayor huella que puede considerarse es el desgaste de los componentes informáticos utilizados y a la electricidad consumida por los mismos.

Para analizar el impacto económico del mismo, se han considerado los siguientes aspectos:

La realización de este TFG ha requerido aproximadamente 300 horas de trabajo, 200 horas en lo referente a programación y 100 horas adicionales para la redacción de los documentos necesarios. Todas estas horas se han realizado con el uso del ordenador, un modelo HP ENVY 13-ah0001ns con procesador i5 8200u, un ordenador portátil de consumo con procesador de bajo coste. La fuente de energía del mismo indica que el consumo de energía en operación es de 72,3 Wh = 0,0723 kWh, con un uso total de 400 h hace un total de 28,92 kWh. Con un precio aproximado de 0,15 €/kWh sale un coste total de 4,338 € por el uso del ordenador.

En cuanto a la mano de obra, el sueldo de un ingeniero con contrato en prácticas no puede ser inferior durante el primer año al 60% del salario de un empleado que desarrolle la misma labor [15]. Considerando un sueldo según convenio de 25.000 € anuales, obtenemos un sueldo en prácticas de **15.000 €** anuales, con aproximadamente 250 días laborales al año con jornada de **8 horas**, lo que suman **2000 h/año**. Esto hace que el coste por hora sea de **7,5 €/hora.** Esto indica que el coste de personal aproximado es de **2250 €** en total.

Un ordenador es un componente de larga duración, que puede ser utilizado por los usuarios por tiempos superiores a 5 años desde su adquisición por

jornadas de 8 horas diarias por norma general, lo que hace una vida útil total cercana a las 10.000 horas de uso, lo que conlleva que este proyecto ha consumido un 3% de su vida útil. El coste del ordenador fue de **661,16 €** por lo que el coste debido a su desgaste se puede estimar en **19,83 €**

Adicionalmente se ha consumido material de oficina como apoyo en algunas partes del TFG, el cual se puede considerar despreciable, no llegando el coste del mismo a superar **1€** de precio total.

Todo esto hace un coste total del proyecto aproximado de **2275 €.** Aun existiendo este coste, el mismo puede ser considerado como una inversión, debido a que este trabajo puede ser aprovechado por parte de los alumnos durante la etapa formativa y su posible ampliación para aumentar las capacidades del programa de cara a un futuro. Además, se han potenciado y ampliado los conocimientos técnicos desarrollados durante el grado en las asignaturas de Estructuras y Construcciones Industriales, Matematicas III y Fundamentos de la Informática (pandeo global, cálculo matricial, resolución por métodos numéricos y programación aplicada a un problema de interés ingenieril).

Durante la realización de este TFG, se han conseguido las competencias indicadas en la guía docente, en especial las capacidades de analisis y síntesis, el aprendizaje y trabajo autónomo, el razonamiento crítico y el analisis lógico y las capacidades personales de expresión tanto oral como escrita.

Es por esto por lo que se considera haber cumplido con todos los objetivos planteados en el proyecto, tanto a nivel del plan de estudios como los propios objetivos del TFG realizado.

Bibliografía

[1] A. Chajes, Principles of structural stability, Prentice Hall, 1974.

[2] CYPE, «CYPE Ingenieros, S.A.,» Junio 2021. [En línea]. Available: http://www.cype.es/.

[3] «CSI Spain,» Junio 2021. [En línea]. Available: https://www.csiespana.com/software/2/sap2000.

[4] ANSYS, «ANSYS,» Junio 2021. [En línea]. Available: https://www.ansys.com/.

[5] Dlubal Software, Inc, «Dlubal,» Junio 2021. [En línea]. Available: https://www.dlubal.com/en.

[6] Tekla, «Tekla Structures,» Junio 2021. [En línea]. Available: https://www.tekla.com/la/productos/tekla-structures.

[7] Open Source, «Python,» Junio 2021. [En línea]. Available: https://www.python.org.

[8] o. Licensed MIT, «Spyder IDE,» Junio 2021. [En línea]. Available: https://www.spyder-ide.org/.

[9] Open Source, «Anaconda,» Junio 2021. [En línea]. Available: https://www.anaconda.com/.

[10] Open Source, «Scipy,» Junio 2021. [En línea]. Available: https://www.scipy.org/.

[11] Open Source, «Matplotlib,» Junio 2021. [En línea]. Available: https://matplotlib.org/.

[12] Vinca equipos industriales, «Vinca equipos industriales,» 06 Julio 2021. [En línea]. Available: https://www.vinca.es/project/proyecto-seat-grua-semiportico-birrail/.

[13] D. Lobo, «Flickr,» Junio 2008. [En línea]. Available: https://www.flickr.com/photos/daquellamanera/2508180589/. [Último acceso: 23 06 2021].

[14] «Codigo Técnico,» Junio 2021. [En línea]. Available: https://www.codigotecnico.org/pdf/Documentos/SE/DBSE-AE.pdf.

[15] Servicio Publico de Empleo Estatal, «SEPE,» Junio 2021. [En línea]. Available: https://www.sepe.es/HomeSepe/.