Integración de las herramientas Python, ANSYS y Google Colab para el desarrollo de objetos de aprendizaje para el curso de Análisis por Elementos Finitos

Juan David Rodríguez Sánchez

Iván Jesús Martínez Garces

Trabajo de Grado para Optar al Título de Ingeniero Mecánico

Director

Octavio Andrés González Estrada

Doctor en Ingeniería Mecánica

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería Mecánica

Bucaramanga

2024

Tabla de Contenido

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Resumen

Título: Integración de las herramientas Python, ANSYS y Google Colab para el desarrollo de objetos de aprendizaje para el curso de Análisis por Elementos Finitos

Autor: Juan David Rodríguez Sánchez, Iván Jesús Martínez Garces

Palabras Clave: Análisis por Elementos Finitos, ANSYS, Python, Google Colab, Objetos de Aprendizaje.

Descripción: La ingeniería mecánica está en constante evolución, al igual que las herramientas utilizadas para analizar fenómenos físico-mecánicos. En el ámbito del Diseño, ANSYS se destaca como un software robusto para el análisis por elementos finitos (FEA), ampliamente empleado en entornos académicos y profesionales. No obstante, su curva de aprendizaje puede ser un desafío considerable para los estudiantes, por lo que resulta crucial desarrollar recursos educativos que faciliten su comprensión. Recientemente, lenguajes de programación como Python han emergido como un complemento valioso en ingeniería, gracias a sus numerosas bibliotecas especializadas. En particular, su integración con ANSYS mediante la librería PYMAPDL ofrece una oportunidad para simplificar el aprendizaje del software. Este estudio investiga el uso de Python y Google Colab para crear objetos de aprendizaje que introduzcan a los estudiantes en el uso de ANSYS. Se abordan cuatro problemas de ingeniería mecánica como estudios de caso a través de una aplicación web, que ilustra y parametriza cada problema. Los estudiantes evaluan esta implementación para comprobar su efectividad en mejorar la curva de aprendizaje y la experiencia en el curso de Elementos Finitos.

Abstract

Title: Integration of Python, ANSYS and Google Colab tools for the development of learning objects for the Finite Element Analysis course.

Author: Juan David Rodríguez Sánchez, Iván Jesús Martínez Garces

Key Words: Finite Element Analysis, ANSYS, Python, Google Colab, Learning Objects.

Description: Mechanical engineering is constantly evolving, as are the tools used to analyze physical-mechanical phenomena. In the field of Design, ANSYS stands out as a robust software for finite element analysis (FEA), widely used in academic and professional environments. However, its learning curve can be a considerable challenge for students, thus it is crucial to develop educational resources that facilitate its understanding. Recently, programming languages such as Python have emerged as a valuable complement in engineering, thanks to their numerous specialized libraries. In particular, its integration with ANSYS through the PYMAPDL library offers an opportunity to simplify software learning. This study investigates the use of Python and Google Colab to create learning objects that introduce students to the use of ANSYS. Four mechanical engineering problems are addressed as case studies through a web application, which illustrates and parameterizes each problem. Students evaluate this implementation to test its effectiveness in improving the learning curve and experience in the Finite Element course.

Introducción

En la actualidad, la mecánica computacional ha transformado profundamente la manera en que los ingenieros abordan el diseño y análisis de sistemas físicos. Estas herramientas permiten prever el comportamiento de estructuras, fluidos y otros fenómenos complejos antes de construir prototipos físicos, lo que optimiza los recursos, disminuye el tiempo del ciclo de diseño y mejora la precisión de los resultados (Brown & Rencis, 2004). Entre las plataformas más utilizadas en la simulación computacional, ANSYS se destaca por su versatilidad y robustez, brindando soluciones avanzadas en análisis estructurales, térmicos y fluidodinámicos. No obstante, su curva de aprendizaje puede representar un desafío significativo para los estudiantes y profesionales que buscan dominarlo (Brown et al., 2012; Ozlek, 2019).

La integración efectiva de herramientas computacionales avanzadas en el currículo de ingeniería no solo prepara a los estudiantes para el entorno profesional dinámico en el que se ha convertido la industria, sino que también les proporciona las habilidades necesarias para innovar y liderar el desarrollo tecnológico futuro (Magana et al., n.d.; Zhang et al., 2021). El análisis por elementos finitos (FEA) es una técnica computacional que permite modelar y analizar la respuesta de componentes estructurales, sistemas de transmisión de calor y la dinámica de fluidos bajo condiciones complejas (Srirekha & Bashetty, 2010). ANSYS Mechanical APDL (MAPDL) (Ansys, n.d.) es una herramienta robusta de simulación que se utiliza ampliamente para aplicaciones de ingeniería en diferentes sectores industriales. MAPDL ofrece capacidades avanzadas para el modelado de problemas físicos mediante FEA. Esta técnica se basa en la discretización del dominio de un problema en una serie de elementos finitos más pequeños, permitiendo una aproximación numérica a problemas complejos que serían, de otro modo, imposibles de resolver analíticamente (Bishay, 2020). La eficacia reside en su flexibilidad para

adaptarse a diferentes tipos de problemas mediante el uso de elementos finitos variados, como lineales o cuadráticos (Steif & Gallagher, 2004), que se utilizan para obtener una solución discreta. La elección del tipo de elemento se basa en factores como la geometría del problema, la precisión deseada y el análisis específico requerido, lo que permite una modelización precisa de una amplia gama de sistemas físicos (Wood, 1996).

El proceso de análisis por FEA implica varias etapas clave, desde la definición inicial de la geometría y las propiedades del material hasta la discretización del dominio, selección de tipos de elementos, aplicación de condiciones de contorno y, finalmente, la resolución del sistema de ecuaciones resultante. El uso de este análisis en el modelado de piezas y fenómenos mecánicos ofrece varias ventajas significativas, ya que permite a los ingenieros crear modelos virtuales detallados de piezas y sistemas antes de fabricar prototipos físicos, lo que no solo reduce los costos y el tiempo asociados con la producción de múltiples prototipos, sino que también permite identificar y corregir problemas potenciales en las etapas iniciales del diseño (Rojas-Sola & Barranco-Molina, 2024; Toraño et al., 2008). Esto facilita el análisis de escenarios y condiciones extremas que serían difíciles o imposibles de probar experimentalmente, permitiendo a los ingenieros, por ejemplo, modelar el comportamiento de una pieza bajo cargas extremas, vibraciones y temperaturas altas o bajas, proporcionando una comprensión profunda de cómo se desempeñará en el mundo real (Müzel et al., 2020), lo cual es crucial para la seguridad y la fiabilidad, especialmente en aplicaciones críticas como la aeronáutica, la automoción y la ingeniería civil. La capacidad de mejorar el diseño mediante modelado iterativo in silico permite mejorar significativamente el rendimiento y la eficiencia de los productos finales, ya que los ingenieros pueden ajustar parámetros y geometrías para encontrar la configuración óptima que

maximice la resistencia y minimice el peso, o cumpla con otros criterios de rendimiento específicos.

Respecto a la introducción de este tipo de métodos numéricos en el currículo de las Escuelas de Ingeniería, las herramientas de uso común en la industria han sido la estrategia general para introducir a los estudiantes al FEA, junto con una base sólida en los fundamentos matemáticos y principios subyacentes. Software comercial como ANSYS, Abaqus y COMSOL Multiphysics han sido fundamentales en el estudio del FEA, ofreciendo capacidades avanzadas para la práctica del modelado y análisis. Estas herramientas, aunque poderosas, a menudo requieren recursos financieros y tiempo considerable para que los estudiantes y profesionales adquieran competencia en su uso (Balamuralithara & Woods, 2009).

La creación de software personalizado por parte de educadores e investigadores ofrece una alternativa adaptada específicamente a los objetivos educativos, permitiendo a los estudiantes interactuar con aplicaciones diseñadas para mejorar su comprensión de los principios del FEA (Nickerson et al., 2007; Riojas et al., 2012). No obstante, esto requiere una inversión significativa en tiempo y recursos para su desarrollo y mantenimiento, lo que puede ser un obstáculo para muchas instituciones educativas (Ralph et al., 2020), así como las competencias en programación por parte de los estudiantes.

Lenguajes de programación como Python permiten de una forma más sencilla la aproximación a este tipo de problemas. Anteriormente había integración entre lenguajes de bajo nivel como Fortran o C junto con software de FEA, siendo un problema reiterativo la curva de aprendizaje. Python se plantea como una solución para estudiantes con baja exposición a los lenguajes de programación debido a la sintaxis sencilla que tiene frente a lenguajes de bajo nivel (Bai et al., 2021; Kuroki, 2021; Tufino et al., 2024). Esta característica facilita la enseñanza y el

aprendizaje de conceptos programáticos fundamentales, así como conceptos avanzados de ingeniería (Thaker et al., 2020). La capacidad de escribir código que es comprensible permite a los ingenieros y estudiantes centrarse en la solución de problemas de ingeniería en lugar de luchar contra la complejidad del lenguaje de programación (Feng et al., 2024). Python actúa como un lenguaje puente entre disciplinas, permitiendo a los ingenieros combinar análisis numérico, visualización de datos y automatización de tareas en un flujo de trabajo cohesivo.

Asimismo, la integración del lenguaje de programación Python con bibliotecas específicas para ANSYS, como PyMAPDL, ha abierto nuevas oportunidades para simplificar y optimizar el uso de herramientas de simulación. Estas bibliotecas no solo permiten automatizar procesos complejos, sino que también facilitan el análisis computacional utilizando una sintaxis más accesible y flexible, lo que resulta en una mayor eficiencia en la enseñanza y aplicación de conceptos de ingeniería (Oberkampf et al., 2004). Las librerías especializadas como NumPy ofrecen herramientas robustas para la simulación y análisis computacional en ingeniería (Sumets, 2022). NumPy proporciona estructuras de datos eficientes y operaciones matemáticas esenciales para la manipulación de grandes conjuntos de datos numéricos, cruciales en las simulaciones de FEA (Thaker et al., 2020).

La integración de estas bibliotecas en los flujos de trabajo de simulación computacional no solo complementa las capacidades de herramientas comerciales como ANSYS, sino que también permite una mayor interoperabilidad y flexibilidad en la investigación y desarrollo. Estos desarrollos recientes en el análisis por elementos finitos, como la optimización de diseño y el análisis de sensibilidad, destacan cómo la sinergia entre Python y herramientas de FEA fomenta la innovación y la investigación avanzada en ingeniería (Sumets, 2022; Thaker et al., 2020), competencias deseables en el currículo.

El presente trabajo desarrolla objetos de aprendizaje que integran ANSYS con Python a través de Google Colab, un entorno colaborativo en la nube que facilita el acceso a los recursos computacionales sin necesidad de instalaciones locales complejas, proponiendo una metodología que favorece la accesibilidad. La elección de estas herramientas responde a la necesidad de simplificar el proceso de aprendizaje del análisis por elementos finitos (FEA), un método ampliamente utilizado en la ingeniería mecánica. Así, se espera no solo mejorar la curva de aprendizaje de los estudiantes, sino también una mayor comprensión y aplicación de estos conceptos en situaciones reales de ingeniería y fomentar el desarrollo de competencias en programación (Higbee & Miller, 2021).

1. Objetivos

1.1 Objetivo General

Desarrollar objetos de aprendizaje para el curso de Análisis por Elementos Finitos mediante la integración de las herramientas Python, ANSYS y Google Colab.

1.2 Objetivos Específicos

- Definir y estructurar las competencias y aplicaciones más relevantes de Python, tales como programación orientada a objetos, manipulación de archivos y librerías de estadísticas básicas.
- Proporcionar una introducción a la librería PyMAPDL, presentando su instalación, configuración y uso básico destacando su flexibilidad, potencia y capacidad para abordar diversos aspectos del análisis por elementos finitos.
- Diseñar ejercicios prácticos utilizando PyMAPDL para realizar un análisis detallado de estructuras mecánicas variadas
- Crear una WebApp con PyMAPDL destinada a la visualización y análisis de los resultados de simulaciones de elementos finitos diseñando rúbricas analíticas para evaluar los resultados de aprendizaje establecidos en el curso, proporcionando criterios claros y objetivos para la evaluación.

3. Metodología.

Para la integración de las herramientas Python y ANSYS en el desarrollo de objetos de aprendizaje para el curso de Elementos Finitos, se abordó el diseño, implementación y evaluación de dichos objetos. Este proceso buscó mejorar la comprensión y el desempeño de los estudiantes, introduciendo nuevas herramientas para proporcionar retroalimentación y ampliar su visión sobre otras disciplinas, considerando tanto los procedimientos utilizados como las circunstancias del estudio y la estructura adoptada.

Inicialmente, se desarrollaron guías de introducción a Python mediante Jupyter Notebooks en Google Colab, permitiendo a los estudiantes familiarizarse con la programación básica y manipulación de datos de manera accesible y colaborativa. Posteriormente, se introdujo PyMAPDL a través de Jupyter Notebooks, utilizando Visual Studio Code para ejecutar los ejercicios localmente, dado que era necesario tener ANSYS instalado en las computadoras de los estudiantes.

Los objetos de aprendizaje diseñados abordaron cuatro problemas mecánicos específicos, cuyo análisis se profundiza en los apéndices: la placa con agujero (Apéndice B), el soporte de esquina (Apéndice C), el cortador de torno (Apéndice D) y la viga en voladizo (Apéndice E). Estos problemas fueron seleccionados por su relevancia en la ingeniería mecánica y permitieron a los estudiantes aplicar los conceptos de FEA a situaciones reales y variadas.

En el ejercicio de placa con agujero, se configuró un modelo paramétrico en PyMAPDL para una placa rectangular con múltiples agujeros. La simulación incluyó la definición de parámetros como dimensiones de la placa, número y radio de los agujeros, y propiedades del material. La configuración de ANSYS MAPDL permitió crear la geometría de la placa y los

agujeros, definir propiedades materiales, aplicar cargas y condiciones de contorno, generar la malla, y finalmente resolver el análisis estático y visualizar los contornos de tensión.

Para el análisis del soporte de esquina se creó la geometría mediante combinaciones de rectángulos y círculos. El análisis incluyó la creación de geometrías, definición de las propiedades del material, generación de la malla, aplicación de cargas y condiciones de contorno, y la resolución del modelo para determinar desplazamientos y tensiones. La visualización de resultados se facilitó mediante gráficos que mostraban áreas, tensiones, y deformaciones mediante PyMAPDL.

En el análisis estructural del cortador de torno el ejercicio implicó configuraciones iniciales de la sesión de MAPDL, importación y preparación de la geometría, definición de propiedades materiales y elementos, y post procesamiento detallado para visualizar los efectos de las cargas aplicadas sobre la integridad estructural del cortador.

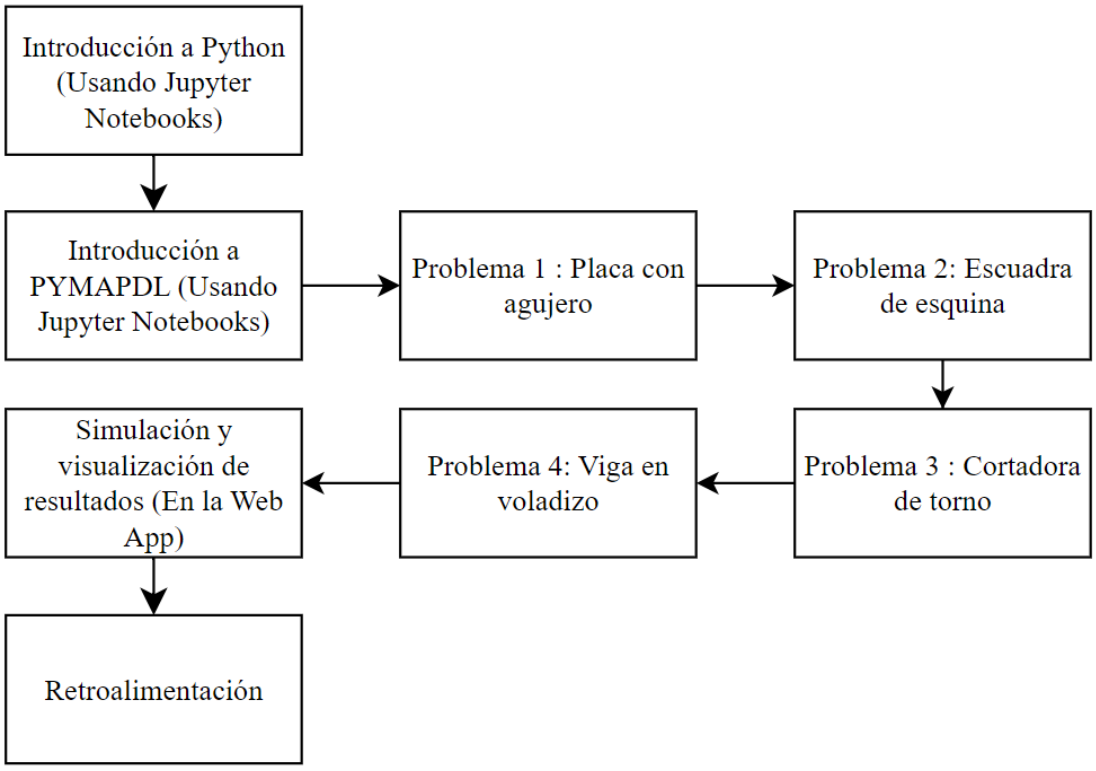
Finalmente, para la viga en voladizo, se simuló la delaminación de placas compuestas utilizando PyMAPDL y otras librerías para el tratamiento de datos de Python para el post procesamiento. El modelo incluyó la configuración de entradas geométricas, definición de parámetros materiales, creación de la geometría y el mallado, generación de elementos cohesivos entre superficies de contacto, y análisis estático no lineal. Los resultados se analizaron para evaluar la relación entre la fuerza y el desplazamiento y se visualizaron usando Matplotlib.

La simulación y visualización de los resultados se realizaron mediante una web app que integraba los cálculos realizados en los notebooks, proporcionando una experiencia práctica y directa de la teoría estudiada. En la figura 1, se aparecía el flujo de la metodología presentada a los estudiantes, incluyendo a demás notebooks y guías interactivas para tener una mayor comprensión, teniendo Notebooks introductorias para las partes de Python y guías para el estudio con MAPDL,

el proceso culminó con una etapa de retroalimentación donde los estudiantes pudieron evaluar su aprendizaje y entender mejor las aplicaciones prácticas de la teoría.

Figura 1

Diagrama de flujo de la metodología general



Nota. Flujograma con los temas de introducción, resolución de cuatro problemas (placa con agujero, escuadra de esquina, cortadora de torno y viga en voladizo), y la simulación, visualización de resultados en una app web más una retroalimentación.

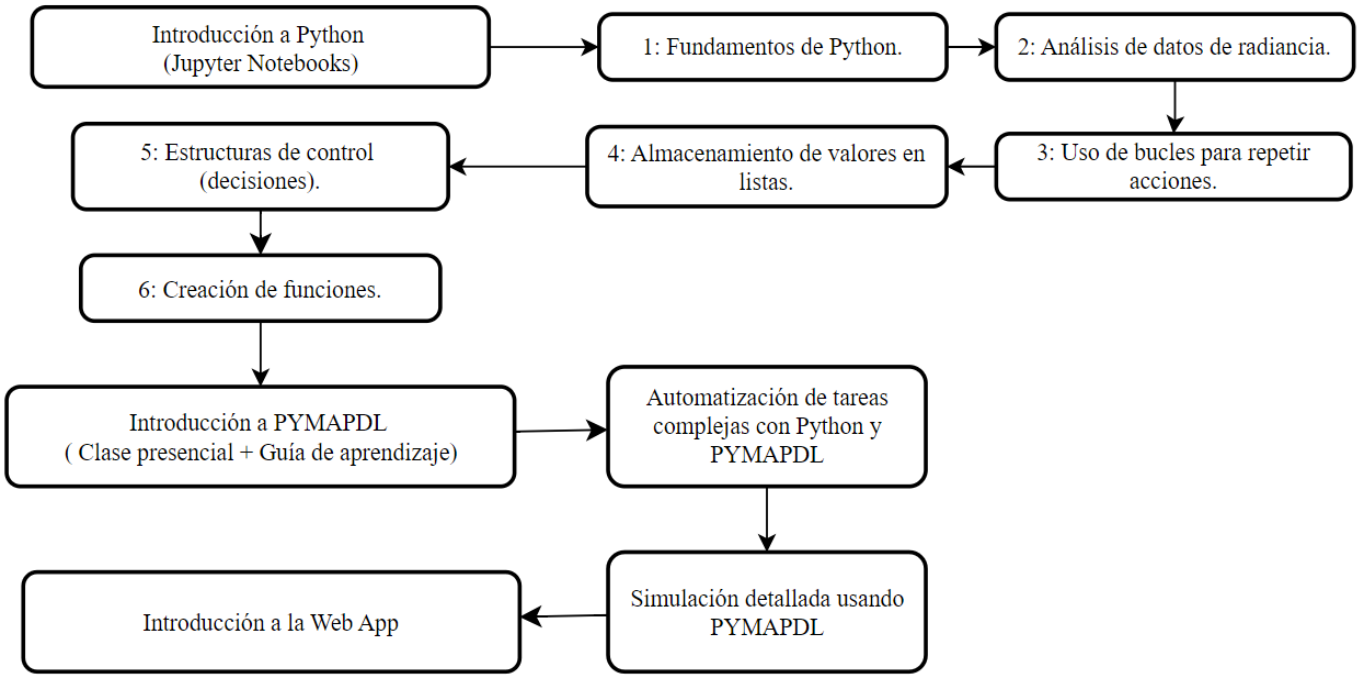
3.1 Selección de herramientas.

En la primera fase se familiariza a los estudiantes con Python. Después de que los estudiantes tuvieran una breve introducción con Python, se presentó la librería PyMAPDL,

mostrando cómo se pueden automatizar tareas complejas y la realización de simulaciones detalladas, las cuales antes realizaban con MAPDL. En la Figura 2, se muestra un diagrama que facilita el entendimiento del proceso y los conceptos básicos, proporcionando a los estudiantes una guía visual clara para mejorar la organización de las tareas y su aprendizaje.

Figura 2

Diagrama de flujo del proceso de introducción y conceptos básicos



Nota. Elaboración propia.

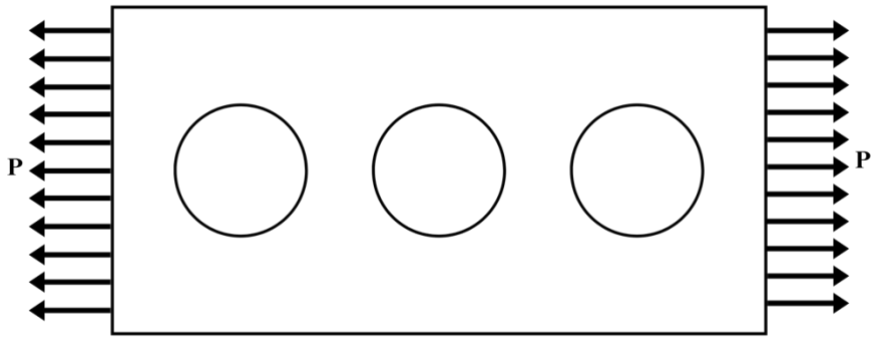
Posteriormente, se desarrollaron modelos específicos para abordar problemas mecánicos seleccionados, incluyendo la configuración de simulaciones, la aplicación de condiciones de contorno y el post procesamiento.

3.1.1 Modelo placa con agujero.

En este ejercicio utilizamos PyMAPDL para analizar una placa rectangular con múltiples agujeros bajo carga específica. La placa, de acero, mide 0.5 metros de longitud, 0.25 metros de ancho, y 0.1 metros de profundidad, con tres agujeros de 0.5 metros de radio. El acero tiene un módulo de Young de 2×10^11 [Pa] y un coeficiente de Poisson de 0.27, con una carga axial aplicada de 1000 Pa. En la Figura 3 se presenta el esquema del modelo de la placa con agujero.

Figura 3

Esquema del modelo placa con agujero



Nota. Elaboración propia, placa rectangular con agujeros centrales, donde se aplican condiciones de borde y cargas externas para simular el comportamiento bajo tensión.

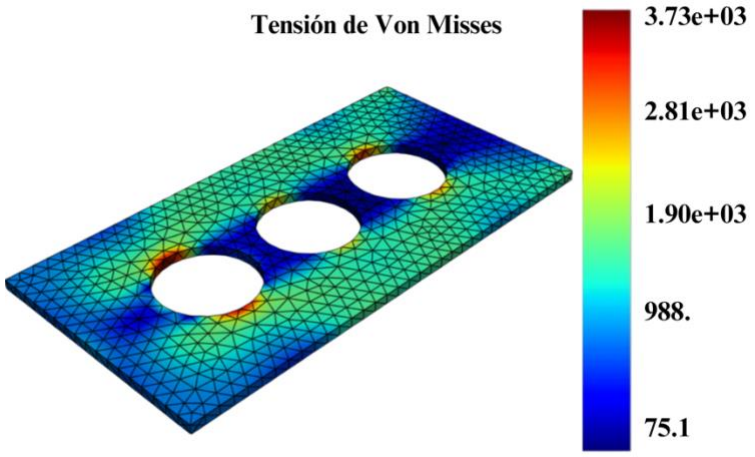
La configuración inicial del modelo incluye la definición de parámetros, seguido por la generación de la geometría de la placa y la configuración de las propiedades del material y el mallado. Además, se aplican condiciones de contorno fijas y se distribuye la presión sobre la

superficie especificada. El procedimiento se ilustra en detalle, lo que permite una comprensión profunda de las variables involucradas.

Tras ejecutar el solucionador de ANSYS, se resuelven las tensiones y deformaciones para obtener la solución al problema estructural. Los resultados se visualizan para evaluar el comportamiento estructural de la placa, mostrando las tensiones principales y equivalentes, como se evidencia en la Figura 4, donde se muestra la distribución de tensiones de Von Mises en la placa.

Figura 4

Distribución de Tensión de Von Mises



Nota. Elaboración propia, placa con tres agujeros bajo carga, donde las concentraciones de tensión se localizan alrededor de los bordes de los agujeros.

El análisis detallado de la placa con agujeros permite evaluar la influencia de las condiciones de contorno y las características del material en la respuesta estructural.

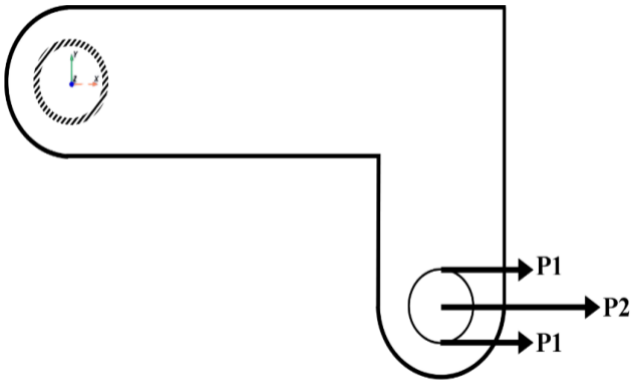
3.1.2 Modelo escuadra de esquina.

Se modela un soporte de esquina compuesto por dos partes rectangulares y terminaciones semicirculares en cada extremo. La primera sección rectangular tiene longitud de 6 unidades y altura de 2 unidades. La segunda sección, adyacente a la primera, con longitud de 2 unidades y altura de 3, extendiéndose verticalmente hacia abajo desde el extremo derecho de la primera sección. Cada extremo de la estructura se completa con semicírculos de 1 unidad de radio.

El material seleccionado para el soporte es acero, con módulo de Young de 3X10^7 psi y un coeficiente de Poisson de 0.27. El círculo en el extremo superior está fijado, impidiendo desplazamiento en esa zona, mientras que el círculo inferior recibe una carga que varía de 50 psi en sus extremos a 500 psi en el centro.

Figura 5

Geometría del modelo escuadra de esquina



Nota. Elaboración propia, estructura en forma de "L", zonas de carga y fijación indicadas.

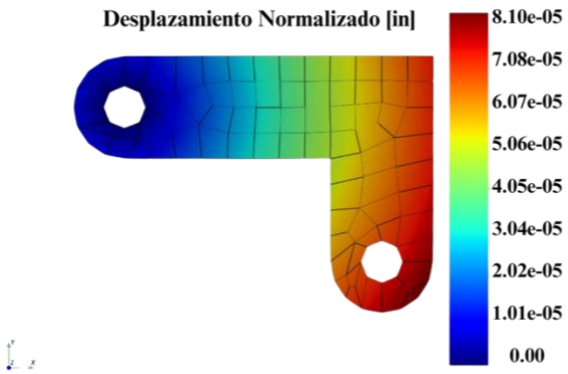
El análisis comienza con la configuración del modelo en PyMAPDL, seguido por la simulación de las condiciones de carga. Esta etapa es crucial para comprender cómo se aplican las

cargas y cómo se espera que el soporte responda a estas condiciones. Los detalles de la configuración inicial y la aplicación de las cargas se especifican con claridad, lo que ayuda a anticipar los posibles comportamientos estructurales bajo carga.

Posteriormente, se evalúan las respuestas estructurales, como los desplazamientos y las tensiones de Von Mises, proporcionando una comprensión detallada de cómo el diseño del soporte responde a las fuerzas aplicadas. Los resultados del análisis se visualizan en la Figura 6, donde se muestra la distribución de las tensiones de Von Mises en el soporte.

Figura 6

Desplazamiento Normalizado en Escuadra de Esquina



Nota. Elaboración propia, desplazamiento normalizado en un soporte de esquina bajo carga, con variaciones de desplazamiento a lo largo de la estructura, especialmente cerca de los agujeros

3.1.3 Modelo cortadora de torno.

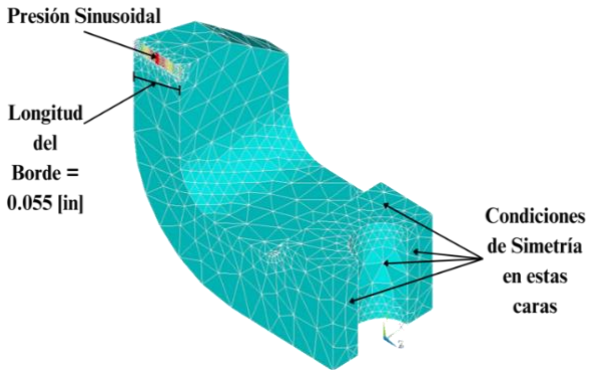
Para el ensayo de un cortador de torno, se implementa un modelo incluido en la biblioteca de ANSYS para estudiar las tensiones y deformaciones bajo condiciones de carga específicas. El modelo es un cortador con las dimensiones especificadas, y se simula aplicando una carga de

10000 psi distribuida como se aprecia en la Figura 7. El material utilizado para el cortador tiene un módulo de elasticidad de 1.0e7 psi y un coeficiente de Poisson de 0.27.

Se crea un sistema de coordenadas local para aplicar una carga de presión que varía según la longitud del área de aplicación, lo que permite una representación precisa de las condiciones reales de operación. En las Figura 8, se presentan las distribuciones de deformaciones y tensiones en la cortadora de torno. Estas visualizaciones permiten entender cómo se distribuyen las tensiones y deformaciones bajo las condiciones simuladas, proporcionando una imagen de la afectación de los parámetros en los esfuerzos.

Figura 7

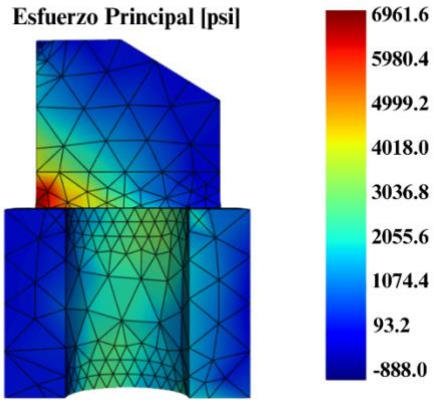
Geometría del modelo de cortadora de torno



Nota. Estructura curvada con malla triangular, aplicada a una simulación de cargas y restricciones en los extremos para analizar su comportamiento bajo esfuerzo.

Figura 8

Distribución de Tensión en Cortadora de Torno



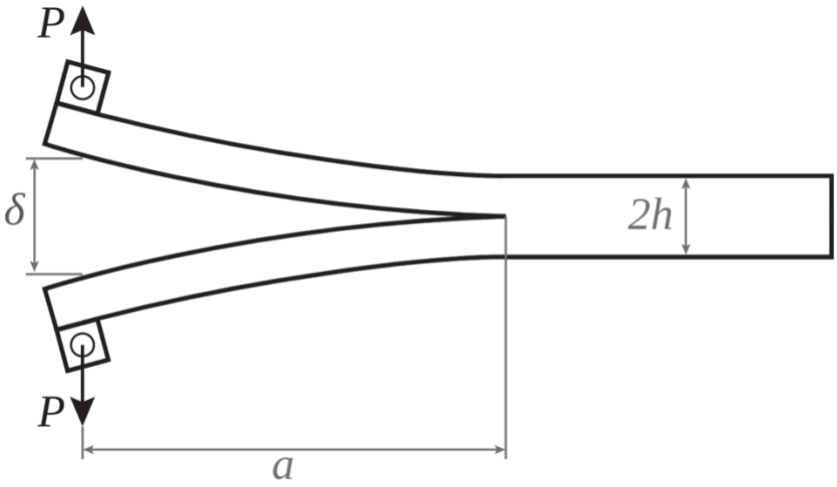
Nota. Distribución de tensiones en un modelo de cortadora de torno, destacando las áreas de mayor esfuerzo concentradas en la parte interna y la base de la estructura.

3.1.4 Modelo viga en voladizo.

La doble viga en voladizo implementa un modelo con el objetivo de analizar las deformaciones y tensiones bajo condiciones específicas de carga de flexión. La configuración del modelo incluye vigas de 75 mm de longitud, 10 mm de precorte, 25 mm de ancho y 1.7 mm de altura. Se simula un desplazamiento de 10 mm aplicado en el extremo de las vigas. El material utilizado para las vigas compuestas es caracterizado por un módulo de elasticidad de 61340 psi, una densidad de 1.42X10^-9 psi y un coeficiente de Poisson de 0.1. En la Figura 9 se presenta el esquema del modelo de la doble viga en voladizo.

Figura 9

Esquema de la geometría del modelo de viga en voladizo



Nota. Viga con una apertura en su extremo libre, sometida a cargas P en sus extremos, generando una deflexión δ a lo largo de la longitud a, con un espesor de 2h.

Tras configurar la geometría y las propiedades materiales, aplicamos condiciones de contorno y simulamos las cargas. Este proceso es esencial para entender la interacción de las cargas con la estructura, permitiendo una evaluación detallada de cómo las vigas responden a estas condiciones. La preparación meticulosa y la aplicación correcta de las cargas aseguran resultados de análisis estructurales precisos y fiables.

Figura 10

Gráfica de fuerza vs desplazamiento del modelo viga en voladizo

Nota. Elaboración propia. La figura representa las fuerzas contra el desplazamiento de la viga en voladizo.

Los resultados de nuestro análisis estructural se presentan en la Figura 10, que muestra la gráfica de Fuerza vs. Desplazamiento. Esta visualización proporciona una comprensión clara de la relación directa entre las fuerzas aplicadas y las deformaciones experimentadas por las vigas, ofreciendo datos valiosos para futuras mejoras en el diseño y evaluación de estructuras compuestas.

3.2 Web App

Para mejorar el acceso y la interacción con los ejercicios de análisis por elementos finitos, se ha desarrollado una aplicación web que facilita a los estudiantes la parametrización de datos de entrada, ejecución de simulaciones y visualización de resultados. Esta herramienta permite a los

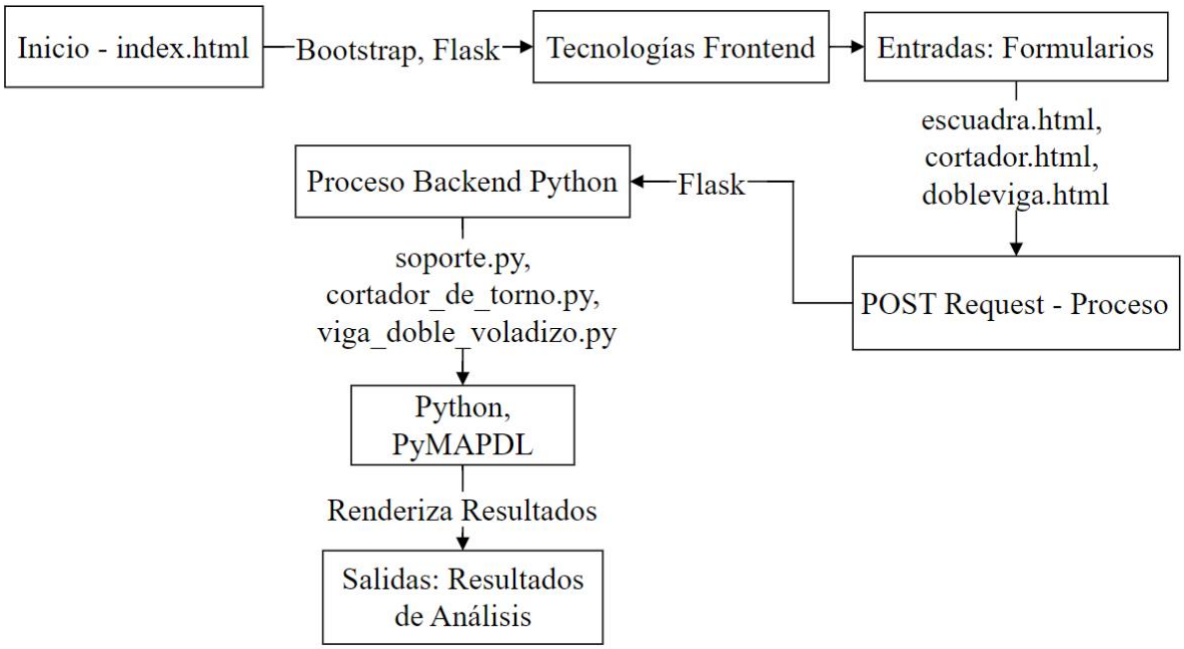
usuarios modificar parámetros y explorar cómo diferentes variables influyen en los resultados. La gestión de formularios e interacción del usuario es esquematizada en la Figura 11, mientras que el desarrollo completo de la aplicación se detalla en el Apéndice F.

En el backend, se utiliza Flask, un marco de aplicación web de Python, para gestionar las solicitudes del usuario y procesar datos de entrada a través de formularios. Flask coordina las simulaciones realizadas por diversos scripts, integrando las operaciones del sistema y ejecutando los cálculos estructurales. Bootstrap se emplea en el frontend para asegurar que los formularios y las páginas web sean responsivos y estéticamente agradables. Esta librería permite la creación de una interfaz de usuario coherente y funcional, mejorando la experiencia del usuario.

La combinación de Flask y Bootstrap permite una parametrización efectiva de los datos de entrada y una visualización clara de los resultados, mejorando así la interactividad y el aprendizaje práctico de los estudiantes. Estas tecnologías juntas no solo facilitan el manejo técnico de las simulaciones, sino que también aseguran que los estudiantes puedan interactuar de manera intuitiva con la plataforma, logrando un mayor entendimiento de los principios de ingeniería estructural y análisis por elementos finitos.

Figura 11

Diagrama de flujo de la WebApp



Nota. Flujo de trabajo de una aplicación web que utiliza Bootstrap y Flask para el frontend, con entradas proporcionadas por formularios HTML. Estos datos se envían mediante POST request al backend, donde Python y PyMAPDL procesan los scripts de análisis estructural. Los resultados del análisis son renderizados y devueltos al usuario.

4. Resultados.

Para evaluar la percepción de los estudiantes sobre las herramientas integradas, se realizaron encuestas en tres salones de clases: Dinámica, Diseño de Máquinas 2 y la electiva de Análisis por Elementos Finitos.

En cada sesión, el desarrollo de la clase siguió una metodología clara y estructurada. Inicialmente, se mostró la página web creada, explicando tanto su funcionamiento como el código base, mientras se detallaban las tecnologías utilizadas, como Python y ANSYS. Posteriormente, se presentó el ejercicio de placa con agujero detallado en el Apéndice B, donde se explicó cómo se aplican los conceptos teóricos en las simulaciones de análisis por elementos finitos. Esta demostración sirvió para mostrar la forma en que se imparte la materia y para profundizar en la explicación de los conceptos fundamentales.

Durante la sesión, se abrió un espacio para que los estudiantes realizaran preguntas. Las consultas más comunes giraron en torno al uso de ANSYS y la integración de Python, así como sobre la creación de informes automáticos mediante la aplicación web. Finalmente, se aplicó una encuesta para medir el nivel de satisfacción y la utilidad percibida de las herramientas presentadas.

Las clases de Diseño de Máquinas II se llevaron a cabo durante los días 19 y 27 de junio de 2024, con la participación de 29 estudiantes que respondieron la encuesta. En cuanto a la clase de Dinámica se realizó el 2 de julio de 2024, contando con 23 estudiantes que completaron la encuesta.

La encuesta aplicada a los estudiantes incluyó preguntas como su nivel de satisfacción con la información proporcionada durante la clase, así como la utilidad que percibieron en la integración de Python y ANSYS para el análisis de elementos finitos. También se les consultó si la presentación aumentó su interés en el uso de herramientas de programación para el análisis de ingeniería y si deseaban recibir más información sobre algún tema específico tratado durante la

sesión. Finalmente, se les preguntó si estarían interesados en matricular la electiva de Análisis por Elementos Finitos tras la charla.

Por su parte, la clase de la electiva de Análisis por Elementos Finitos tuvo lugar el 25 de junio de 2024. Durante esta sesión, se aplicó la misma metodología de enseñanza, mostrando primero la página web y su código base, seguida de la presentación del ejercicio de placa con agujero detallado en el Apéndice B, incluyendo una actividad complementaria respecto a la creación de la malla del ejercicio de soporte de esquina detallado en el Apéndice C. Se resolvieron dudas y preguntas relacionadas con la aplicación de las herramientas y la automatización de informes, y finalmente se aplicó una encuesta a los estudiantes. En esta ocasión, participaron 8 estudiantes, quienes respondieron preguntas similares sobre su percepción de la metodología, su utilidad y si esta les ayudó a comprender mejor los conceptos de elementos finitos.

4.1. Nivel de satisfacción y utilidad de las herramientas.

4.1.1. Evaluación de las herramientas en la clase de Diseño de Máquinas II.

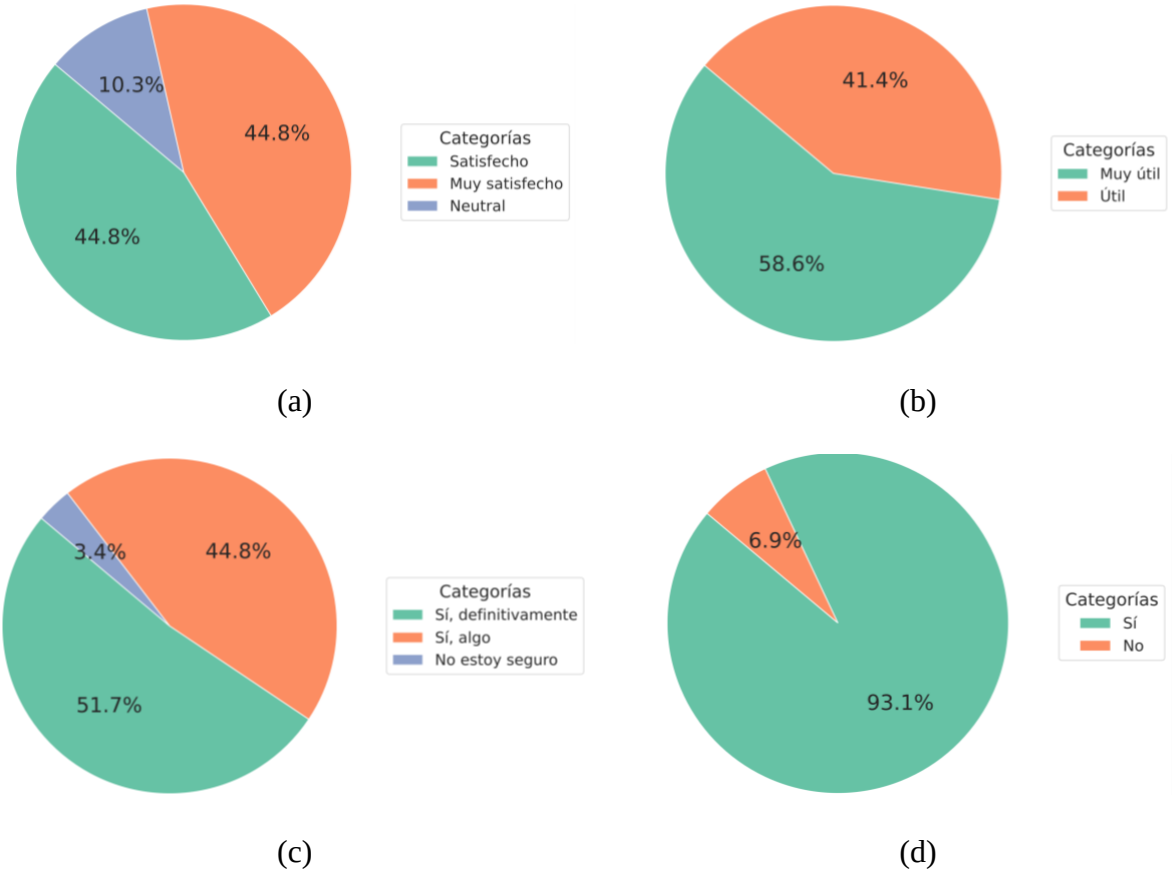
En el curso de Diseño de Máquinas II se evaluó la satisfacción de los estudiantes con las sesiones de aprendizaje. Los resultados mostraron que el 38% de los estudiantes se declaró “Muy satisfecho” y otro 38% “Satisfecho”, mientras que aproximadamente un 10% mantuvo una opinión “Neutral”, indicando una aceptación general del contenido. La integración de Python y ANSYS en el análisis por elementos finitos fue altamente valorada. El 87.5% de los estudiantes calificó esta combinación como “Muy útil” y el 12.5% como “Útil”, evidenciando que estas herramientas facilitaron la comprensión de conceptos complejos en análisis estructural. El interés por utilizar herramientas de programación en el análisis de ingeniería aumentó significativamente. Un 68% de los estudiantes expresó un incremento notable en su interés, y un 25% reportó un crecimiento

moderado, sumando un total del 93.1% con mayor disposición a explorar estas herramientas en futuros proyectos.

Respecto a la intención de matricular la electiva de Análisis por Elementos Finitos, el 91% de los estudiantes mostró interés en cursarla, reflejando un fuerte deseo de profundizar en el tema. Solo un 9% no manifestó interés, lo que resalta el impacto positivo del uso de herramientas avanzadas como Python y ANSYS en su formación académica.

Figura 12

Encuesta realizada en la clase de Diseño de Máquinas II



Nota. Elaboración propia. (a)¿Qué tan satisfecho estás con la información proporcionada hoy?, (b) ¿Cómo calificarías la utilidad de la integración de Python y ANSYS para el análisis de elementos finitos?, (c) ¿La presentación ha aumentado tu interés en el uso de herramientas de programación para el análisis de ingeniería?, (d) Después de esta charla, ¿estarías interesado en matricular la electiva de Análisis por Elementos Finitos?

Los temas específicos sobre los cuales los estudiantes desearían recibir más información, las respuestas fueron variadas. Varios estudiantes expresaron interés en aprender más sobre ANSYS y la creación de informes automáticos con la Web App.

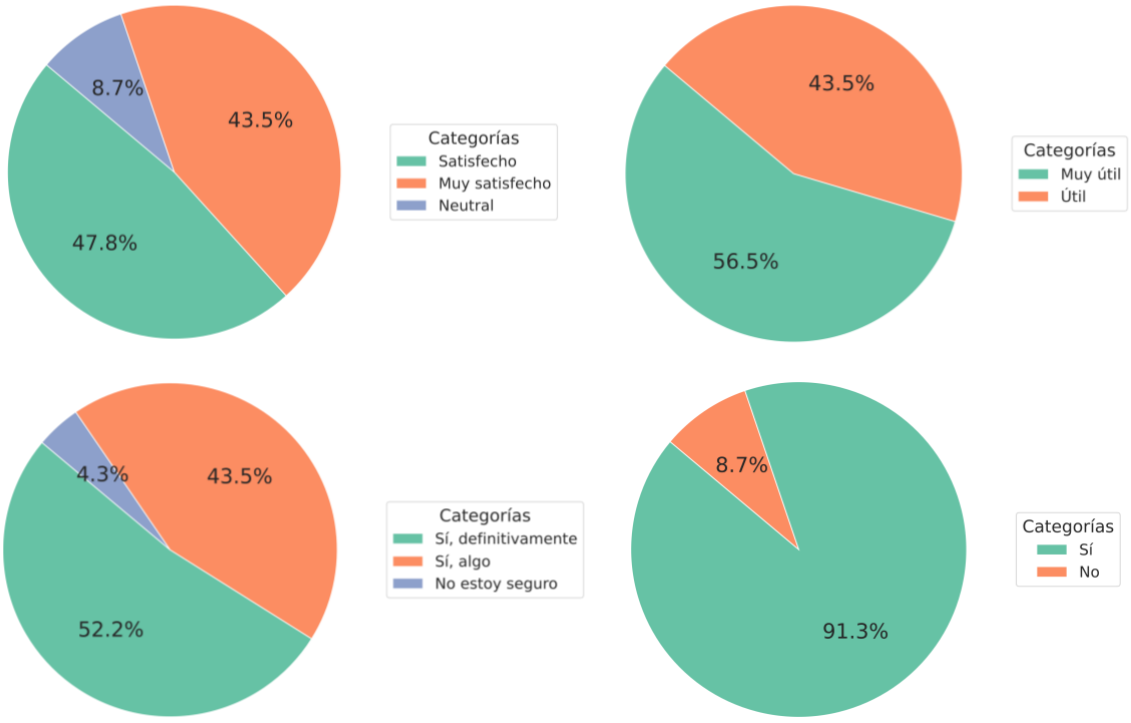
4.1.2. Evaluación de las herramientas en la clase de clase de Dinámica.

Con el objetivo de evaluar la efectividad de las herramientas empleadas en la clase de Dinámica, se aplicó la misma encuesta (Figura 13) utilizada en Diseño de Máquinas II. Los resultados reflejan una alta aceptación y aprecio por la integración de Python y ANSYS, similares a los obtenidos en el curso anterior. Específicamente, alrededor del 40% de los estudiantes indicó estar “Muy satisfecho” y otro 40% afirmó estar “Satisfecho” con la información presentada, sugiriendo que el material y las explicaciones fueron adecuadas y accesibles para la mayoría, al igual que en Diseño de Máquinas II. Un porcentaje menor, cercano al 8%, se mostró “Neutral”, lo que mantiene una tendencia positiva en la percepción general.

En cuanto a la utilidad de integrar Python y ANSYS para el análisis de elementos finitos, un 65% de los estudiantes calificó esta integración como “Muy útil” y un 35% la consideró “Útil”. Aunque estos porcentajes difieren ligeramente de los de Diseño de Máquinas II, donde el 87.5% la calificó como “Muy útil”, ambos resultados evidencian que las herramientas fueron bien recibidas en el contexto de la simulación de elementos finitos, se vio un incremento consistente con el observado en Diseño de Máquinas II, donde el 68% indicó un aumento notable, reflejando que la exposición a Python y su aplicación en la simulación computacional incentiva a los estudiantes a explorar más a fondo estas herramientas.

Figura 13

Encuesta realizada en la clase de Dinámica



Nota. Elaboración propia. (a)¿Qué tan satisfecho estás con la información proporcionada hoy?, (b) ¿Cómo calificarías la utilidad de la integración de Python y ANSYS para el análisis de elementos finitos?, (c) ¿La presentación ha aumentado tu interés en el uso de herramientas de programación para el análisis de ingeniería?, (d) Después de esta charla, ¿estarías interesado en matricular la electiva de Análisis por Elementos Finitos?.

La encuesta ofreció la oportunidad a los estudiantes de sugerir temas o áreas sobre las cuales les gustaría recibir más información o capacitación. Varias respuestas destacaron el interés en aprender más sobre el uso de ANSYS en profundidad, especialmente en aplicaciones industriales como los intercambiadores de calor o el análisis de elementos finitos en la industria automotriz.

4.1.2. Evaluación de las herramientas en la electiva de Elementos Finitos.

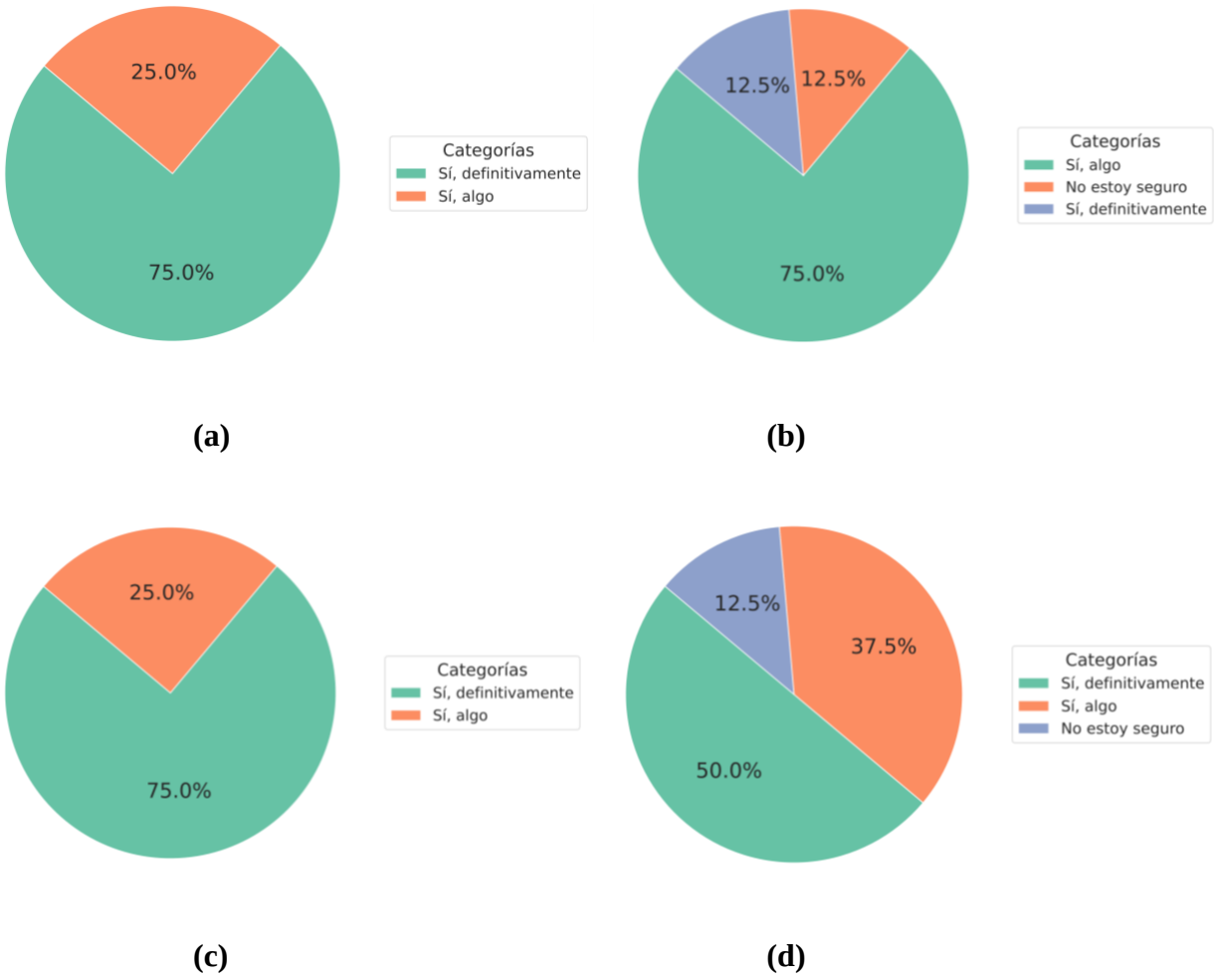
Los resultados muestran que esta metodología combinada ayudó significativamente a aplanar la curva de aprendizaje, facilitando una comprensión más clara y accesible de los conceptos fundamentales del FEA.

El 87.5% de los estudiantes expresó satisfacción con las nuevas herramientas, indicando una alta aceptación de la integración de Python con ANSYS, los resultados obtenidos (Figura 14). Al comparar estos resultados con los obtenidos en Diseño de Máquinas II y Dinámica, se observa que en los cursos anteriores los estudiantes también mostraron altos niveles de satisfacción y apreciación por la integración de Python y ANSYS.

Sin embargo, en la electiva de Análisis por Elementos Finitos, donde el tema se aborda en profundidad, los estudiantes percibieron aún más beneficios en la metodología combinada, destacando su impacto en la simplificación de conceptos complejos y en la eficiencia del proceso de aprendizaje. La consistencia en la alta aceptación de las herramientas integradas a través de diferentes cursos sugiere que la combinación de Python y ANSYS es efectiva en diversos contextos académicos. Mientras que en Diseño de Máquinas II y Dinámica la introducción de estas herramientas aumentó el interés por el uso de programación en ingeniería y por matricular la electiva de Análisis por Elementos Finitos, en esta última se evidencia que la metodología integrada no solo despierta interés, sino que también mejora significativamente la comprensión y aplicación de los conceptos de FEA.

Figura 14

Encuesta realizada en la clase de método de Elementos Finitos



Nota. Elaboración propia. (a) ¿Esta metodología puede ayudar al programa de elementos finitos? (b) ¿El aprendizaje con esta metodología podría ser más sencilla? (c)¿La metodología se podría aplicar a otras materias?" (d) ¿Se siente satisfecho con las nuevas herramientas que la ingeniería está implementando?

La metodología combinada ha demostrado ser una herramienta valiosa no solo para hacer el proceso de aprendizaje más eficiente, sino también para motivar a los estudiantes a explorar nuevas formas de abordar problemas de ingeniería a través de la programación.

Conclusiones

La integración de Python y ANSYS en el curso de análisis por Elementos Finitos ha demostrado ser una estrategia altamente eficaz para mejorar la comprensión teórica y aplicación práctica de los estudiantes. El 87.5% de los estudiantes consideraron esta integración como “muy útil”, evidenciando un impacto positivo en la mejora de su comprensión la electiva. Además, el

93.1% de los encuestados manifestó un mayor interés en el uso de herramientas de programación para el análisis de ingeniería, lo que se tradujo en una participación más activa y motivada en entornos educativos.

El desarrollo de una WebApp permite mejorar la experiencia en la configuración del entorno de ANSYS, gracias a la portabilidad y accesibilidad de la plataforma, los estudiantes pudieron ajustar parámetros de simulación en tiempo real y visualizar los resultados de manera inmediata, facilitando un aprendizaje autónomo y flexible. Este enfoque no solo mejoró el acceso a las herramientas de simulación, sino que también optimizó la eficiencia en la realización de simulaciones, permitiendo a los estudiantes abordar problemas complejos con mayor confianza.

En el marco de la electiva de Análisis por Elementos Finitos, el 88% de los estudiantes informó que la integración de Python les ayudó a mejorar su capacidad para comprender conceptos teóricos en escenarios prácticos. Asimismo, se observó una participación más dinámica en clase, con un mayor número de simulaciones realizadas en menos tiempo, lo que resultó en una mayor eficiencia en el aprendizaje y una aplicación más efectiva de los conceptos clave del FEA. La metodología combinada permitió a los estudiantes no solo reducir el tiempo necesario para la preparación de simulaciones, sino también enfocarse más en la interpretación y análisis de resultados.

Estos resultados subrayan la efectividad de integrar Python con ANSYS para mejorar tanto la enseñanza como la aplicación práctica del análisis por elementos finitos, destacando el potencial de esta metodología para abordar curvas de aprendizaje elevadas en disciplinas técnicas. Al combinar estas herramientas tecnológicas, se facilita un enfoque más accesible y eficiente para los estudiantes, lo que no solo refuerza su comprensión teórica, sino que también les brinda competencias prácticas cruciales para su futura carrera profesional. Esta conexión entre nuevas

tecnologías y metodologías de enseñanza no solo transforma la forma en que los estudiantes aprenden, sino que también abre la puerta a nuevas aplicaciones en otras áreas de la ingeniería.

Referencias Bibliográficas

Asef, P., & Kalyvas, C. (2022). Computer-Aided Teaching Using Animations for Engineering Curricula: A Case Study for Automotive Engineering Modules. IEEE Transactions on Education, 65(2), 141–149. https://doi.org/10.1109/TE.2021.3100471

Bai, H., Wang, X., & Zhao, L. (2021). Effects of the Problem-Oriented Learning Model on Middle School Students' Computational Thinking Skills in a Python Course. Frontiers in Psychology, 12, 771221. https://doi.org/10.3389/FPSYG.2021.771221/BIBTEX

Balamuralithara, B., & Woods, P. C. (2009). Virtual laboratories in engineering education: The simulation lab and remote lab. Computer Applications in Engineering Education, 17(1), 108–118. https://doi.org/10.1002/CAE.20186

Bishay, P. L. (2020). Teaching the finite element method fundamentals to undergraduate students through truss builder and truss analyzer computational tools and student-generated assignments mini-projects. Computer Applications in Engineering Education, 28(4), 1007–1027. https://doi.org/10.1002/CAE.22281

Brown, A. O., Jensen, D., Rencis, J., Wood, K., Wood, J., White, C., Raaberg, K. K., & Coffman, J. (2012). Finite Element Learning Modules as Active Learning Tools. Advances in Engineering Education, 3(1).

Brown, A. O., & Rencis, J. J. (2004). AC 2012-3981: Improving student learning using finite element learning modules: an update in research finding. ASEE Mechanics Division James L. Meriam Service Award.

Cleary, P. W., Thomas, D., Hetherton, L., Bolger, M., Hilton, J. E., & Watkins, D. (2020). Workspace: A workflow platform for supporting development and deployment of modelling and simulation.

Mathematics and Computers in Simulation, 175, 25–61. https://doi.org/10.1016/J.MATCOM.2019.11.011

Ansys Mechanical Advanced - Use of MAPDL in Mechanical | Ansys Training. (n.d.). Retrieved September 21, 2024, from https://www.ansys.com/training-center/course-catalog/structures/ansys-mechanical-advanced-use-of-mapdl-in-mechanical

Feng, Y., Mena, J. A. A., Yang, H., Wang, H., & Jeremić, B. (2024). Domain specific language for finite element modeling and simulation. Advances in Engineering Software, 193, 103666. https://doi.org/10.1016/J.ADVENGSOFT.2024.103666

Grigoropoulos, C., Meier, T., Li, R., Mavrikos, S., Blankenship, B., Vangelatos, Z., Yildizdag, M. E., Erden Yildizdag, M., & Grigoropoulos, C. P. (2023). Obtaining auxetic and isotropic metamaterial in counterintuitive design spaces: An automated optimization approach and experimental characterization. https://doi.org/10.21203/RS.3.RS-3290629/V1

Gupta, A., & Pathania, P. (2021). To study the impact of Google Classroom as a platform of learning and collaboration at the teacher education level. Education and Information Technologies, 26(1), 843–857. https://doi.org/10.1007/S10639-020-10294-1/METRICS

Higbee, S., & Miller, S. (2021). Finite Element Analysis as an Iterative Design Tool for Students in an Introductory Biomechanics Course. Journal of Biomechanical Engineering, 143(12). https://doi.org/10.1115/1.4051659/1114360

Kuroki, M. (2021). Using Python and Google Colab to teach undergraduate microeconomic theory. International Review of Economics Education, 38, 100225. https://doi.org/10.1016/J.IREE.2021.100225

Loveland, M., Valseth, E., Lukac, M., & Dawson, C. (2022). Extending FEniCS to work in higher dimensions using tensor product finite elements. Journal of Computational Science, 64, 101831. https://doi.org/10.1016/J.JOCS.2022.101831

Magana, A. J., Brophy, S. P., & Bodner, G. M. (2012). Student views of engineering professors technological pedagogical content knowledge for integrating computational simulation tools in nanoscale science and engineering. The International Journal of Engineering Education, ISSN-e 0949-149X, Vol. 28, No. Extra 5, 2012, Págs. 1033-1045, 28(5), 1033–1045. https://dialnet.unirioja.es/servlet/articulo?codigo=7392924&info=resumen&idioma=ENG

Mazier, A., Bilger, A., Forte, A. E., Peterlik, I., Hale, J. S., & Bordas, S. P. A. (2022). Inverse deformation analysis: an experimental and numerical assessment using the FEniCS Project. Engineering with Computers, 38(5), 4099–4113. https://doi.org/10.1007/S00366-021-01597-Z/FIGURES/12

Müzel, S. D., Bonhin, E. P., Guimarães, N. M., & Guidi, E. S. (2020). Application of the Finite Element Method in the Analysis of Composite Materials: A Review. Polymers 2020, Vol. 12, Page 818, 12(4), 818. https://doi.org/10.3390/POLYM12040818

Nickerson, J. V., Corter, J. E., Esche, S. K., & Chassapis, C. (2007). A model for evaluating the effectiveness of remote engineering laboratories and simulations in education. Computers & Education, 49(3), 708–725. https://doi.org/10.1016/J.COMPEDU.2005.11.019

Oberkampf, W. L., Trucano, T. G., & Hirsch, C. (2004). Verification, validation, and predictive capability in computational engineering and physics. Applied Mechanics Reviews, 57(5), 345–384. https://doi.org/10.1115/1.1767847

Ozlek, S. (2019). Finite Element Educational Program Improves Mechanical Engineering Technology Student Performance in the Finite Element Class. ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE), 5. https://doi.org/10.1115/IMECE2018-86583

Ralph, B. J., Schwarz, A., & Stockinger, M. (2020). An Implementation Approach for an Academic Learning Factory for the Metal Forming Industry with Special Focus on Digital Twins and Finite Element Analysis. Procedia Manufacturing, 45, 253–258. https://doi.org/10.1016/J.PROMFG.2020.04.103

Riojas, M., Lysecky, S., & Rozenblit, J. (2012). Educational Technologies for Precollege Engineering Education. IEEE Transactions on Learning Technologies, 5(1), 20–37. https://doi.org/10.1109/TLT.2011.16

Rodríguez Pereira, C., de la Roza, F., Gracia Rodríguez, J., Sánchez Lasheras, F., Ranjbar, A., Copley, D., Torres-Gil, M. A., Harvey, É. J., Oria Carerras, A., & de Cos Juez, F. J. (2022). Update and preliminary performance analysis of the New Robotic Telescope structure. Https://Doi.Org/10.1117/12.2629886, 12182, 1216–1222.

Rojas-Sola, J. I., & Barranco-Molina, J. C. (2024). Study of the Mechanical Behavior of a Single-Cylinder Horizontal Steam Engine with a Crosshead Trunk Guide through the Finite-Element Method. Applied Sciences 2024, Vol. 14, Page 5878, 14(13), 5878. https://doi.org/10.3390/APP14135878

Srirekha, A., & Bashetty, K. (2010). Infinite to finite: An overview of finite element analysis. Indian Journal of Dental Research, 21(3), 425–432. https://doi.org/10.4103/0970-9290.70813

Steif, P. S., & Gallagher, E. (2004). Transitioning students to finite element analysis and improving learning in basic courses. Proceedings - Frontiers in Education Conference, FIE, 3. https://doi.org/10.1109/FIE.2004.1408752

Sumets, P. (2022). Computational Framework for the Finite Element Method in MATLAB® and Python. Computational Framework for the Finite Element Method in Matlab, 1–165. https://doi.org/10.1201/9781003265979/Computational-Framework-Finite-Element-computational-frameowork-finite-element-method-matlab

Thaker, N., Shukla, A., Chandaben, S., & Patel, M. (2020). Python as Multi Paradigm Programming Language. Article in International Journal of Computer Applications, 177(31), 975–8887. https://doi.org/10.5120/ijca2020919775

Toraño, J., Diego, I., Menéndez, M., & Gent, M. (2008). A finite element method (FEM) – Fuzzy logic (Soft Computing) – virtual reality model approach in a coalface longwall mining simulation. Automation in Construction, 17(4), 413–424. https://doi.org/10.1016/J.AUTCON.2007.07.001

Tufino, E., Oss, S., & Alemani, M. (2024). Integrating Python data analysis in an existing introductory laboratory course. European Journal of Physics, 45(4), 045707. https://doi.org/10.1088/1361-6404/AD4FCC

Wood, S. L. (1996). A new approach to interactive tutorial software for engineering education. IEEE Transactions on Education, 39(3), 399–408. https://doi.org/10.1109/13.538765

Zhang, J., Cho, H., & Mago, P. J. (2021). Improving Student Learning of Energy Systems through Computational Tool Development Process in Engineering Courses. Sustainability 2021, Vol. 13, Page 884, 13(2), 884. https://doi.org/10.3390/SU13020884

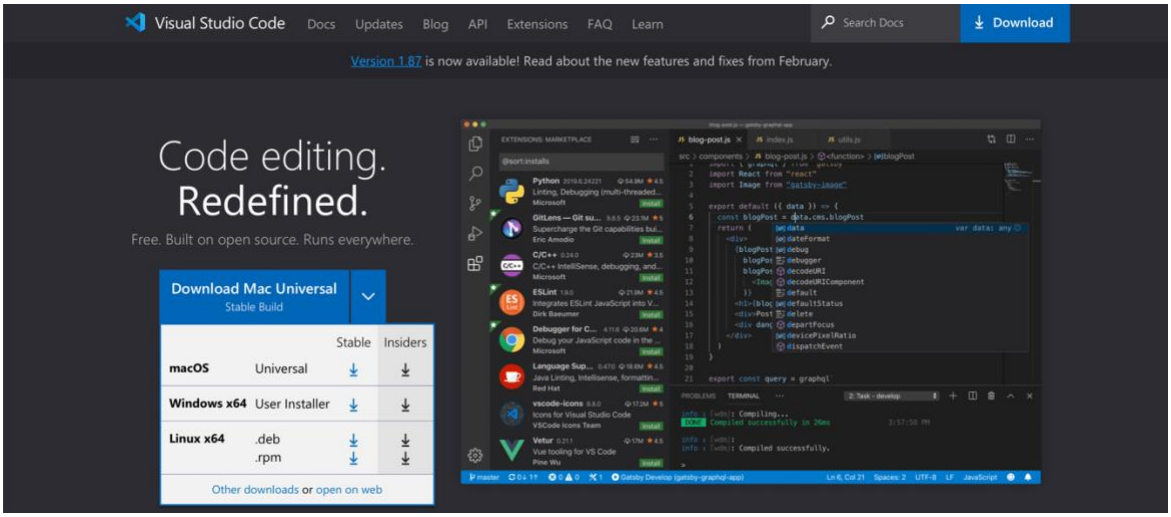
Apéndices

Apéndice A

Instalación Python y entorno de desarrollo.

1.1 Visual Studio Code

Descarga Visual Studio Code
Visita la página oficial de Visual Studio Code: https://code.visualstudio.com/
Haz clic en el botón de descarga para Windows. Se descargará el instalador .exe.

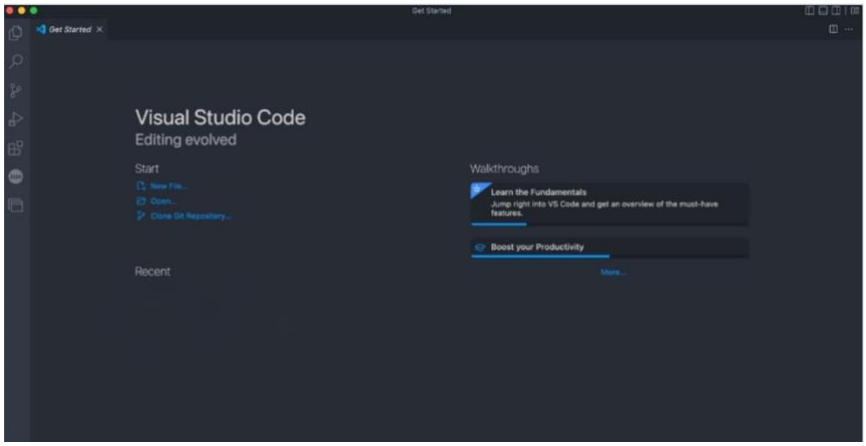


Instalación

Después de la descarga, active el instalador con un doble clic en el archivo .exe. Es crucial aceptar el acuerdo de licencia para seguir adelante; pulse "Siguiente" para avanzar.

El siguiente paso requiere la selección de la ruta de instalación para Visual Studio Code; una vez determinada, proceda con "Siguiente". A continuación, podrá elegir realizar configuraciones adicionales, como añadir un icono de acceso directo en el escritorio; tras realizar su elección, avance con "Siguiente".

Antes de iniciar la instalación, revise y confirme su configuración con el botón "Instalar". Al finalizar la instalación, asegúrese de seleccionar "Launch Visual Studio Code" y clique en "Finalizar" para abrir Visual Studio Code.



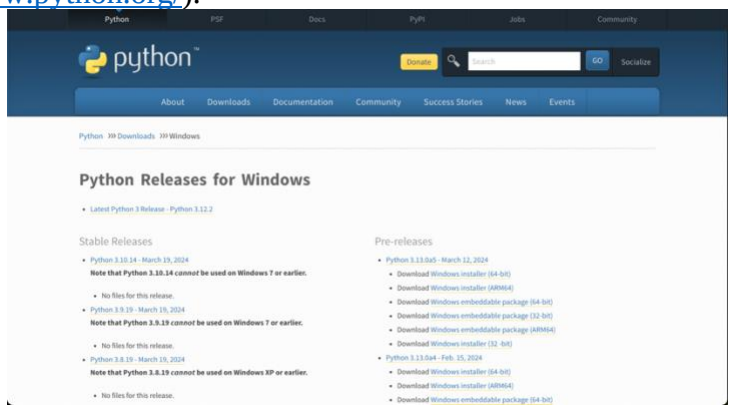
1.2 Introducción a Python enfocado en ANSYS

1. Preparación del Entorno

Para empezar, asegúrate de tener instalado Python 3.6 o superior, así como ANSYS. La integración con Python se puede hacer de varias maneras, incluyendo el uso de PyMAPDL, pero también mediante la automatización de la creación de archivos de entrada (.inp) para ANSYS o el procesamiento de archivos de salida.

Para Windows

-Instalación de Python: Descarga e instala Python desde [python.org](https://www.python.org/).



-Ejecuta el instalador :

Localiza el archivo instalador descargado (normalmente en tu carpeta de Descargas) y haz doble clic sobre él para ejecutar el proceso de instalación. Es posible que el Control de Cuentas de Usuario (UAC – User Account Control) te pida que permitas la instalación. Haz clic en Sí para continuar.

-Verifica la instalación:

Una vez finalizada la instalación, puedes comprobar que Python se ha instalado correctamente abriendo el Símbolo del sistema (busca «cmd» en el menú Inicio) y escribiendo el siguiente comando:

```python

Python –version
```

Pulsa Intro, y deberías ver en la salida la versión de Python que has instalado. Esto confirma que Python se ha instalado correctamente en tu ordenador.

2. Automatización de Archivos de Entrada y Salida

Muchas tareas en ANSYS comienzan con la creación de un archivo de entrada que describe el modelo, materiales, cargas, y análisis. Python puede automatizar la generación de estos archivos.

- Generación de Archivos de Entrada: Crea scripts de Python que escriban archivos de entrada de ANSYS (.inp) basados en parámetros definidos por el usuario.

```python
def generar_archivo_inp(nombre_archivo, parametros):
    with open(nombre_archivo, 'w') as f:
        f.write("/PREP7\n")
        f.write(f"ET,1,SOLID185\n")
        # Añadir más comandos basados en `parametros`
```

```
generar_archivo_inp("modelo.inp", {"largo": 100, "ancho": 50})
```

- Procesamiento de Archivos de Salida Después de ejecutar simulaciones, utiliza Python para leer y analizar los archivos de salida (por ejemplo, .rst) para extraer datos relevantes.

```python
def leer_resultados(nombre_archivo):
    # Implementación de la lectura del archivo
    pass

resultados = leer_resultados("resultado.rst")
```

3. Automatización de Flujos de Trabajo de Simulación

Python puede automatizar la ejecución de ANSYS, desde la preparación de archivos de entrada hasta la ejecución de análisis y la recopilación de resultados.

- Ejecución de ANSYS desde Python: Usa el módulo `subprocess` para ejecutar ANSYS con archivos de entrada generados automáticamente.

```python
import subprocess

subprocess.run(["ansys2023R1", "-b", "-i", "modelo.inp", "-o", "salida.out"])
```

4. Análisis de Datos de Simulación

Utiliza Python para analizar y visualizar los datos de simulación, aprovechando bibliotecas como Pandas, Matplotlib, y SciPy.

- Análisis Estadístico y Visualización:

```python
import pandas as pd
import matplotlib.pyplot as plt

# Suponiendo que `resultados` es un DataFrame de Pandas
resultados.plot(kind='bar')
plt.title('Resultados de Simulación')
plt.xlabel('Elemento')
plt.ylabel('Desplazamiento')
plt.show()
```

1.3 Librería Numpy

NumPy, que significa Numerical Python, es la columna vertebral de muchas otras bibliotecas de Python debido a su capacidad para realizar operaciones matemáticas complejas de manera rápida y eficiente.

1. Conceptos Básicos y Creación de Arreglos

NumPy introduce el concepto de arrays n-dimensionales (ndarray), que son colecciones homogéneas de elementos (generalmente números), permitiendo operaciones matemáticas y lógicas eficientes.

Instalación de NumPy

Si aún no tienes NumPy instalado, puedes instalarlo fácilmente utilizando pip:

```python
pip3 install python
```

Creación de Arreglos

NumPy ofrece múltiples maneras de crear arreglos, desde listas de Python hasta funciones específicas de NumPy para generar arreglos con valores predefinidos.

```python
import numpy as np

a = np.array([1, 2, 3])        # Arreglo unidimensional
b = np.array([[1, 2, 3], [4, 5, 6]])  # Arreglo bidimensional

c = np.zeros((3, 3))           # Arreglo 3x3 de ceros
d = np.ones((2, 2))            # Arreglo 2x2 de unos
e = np.full((2, 2), 7)         # Arreglo 2x2, lleno de 7s
f = np.eye(3)                  # Matriz identidad 3x3
g = np.linspace(0, 10, 5)      # Arreglo de 5 valores, espaciados uniformemente entre 0 y 10
```

2. Manipulación y Acceso a los Elementos de Arreglos

NumPy proporciona una amplia gama de técnicas para acceder y modificar elementos, rebanadas (slices) y subarreglos.

Indexación y Slicing

La indexación y el slicing permiten seleccionar elementos específicos y secciones de arreglos.

```python
arr = np.array([[-1, 2, 0], [4, -0.5, 6]])

# Accediendo a un elemento
elem = arr[0, 1]  # 2

# Slicing
subarr = arr[:, 1:3]  # Selecciona todas las filas, columnas 1 a 2
```

Manipulación de Forma:

Cambiar la forma de un arreglo es crucial para ajustar datos para análisis o visualización.

```python
arr = np.arange(6)  # [0 1 2 3 4 5]
reshaped = arr.reshape((2, 3))  # Cambia la forma a 2x3
```

3. Operaciones Matemáticas y de Álgebra Lineal

NumPy es excepcionalmente poderoso cuando se trata de operaciones matemáticas, ofreciendo desde simples operaciones aritméticas hasta álgebra lineal compleja.

Operaciones Aritméticas:

Las operaciones aritméticas con arreglos se realizan elemento a elemento.

```python
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

suma = a + b        # Suma elemento a elemento
producto = a * b    # Producto elemento a elemento
```

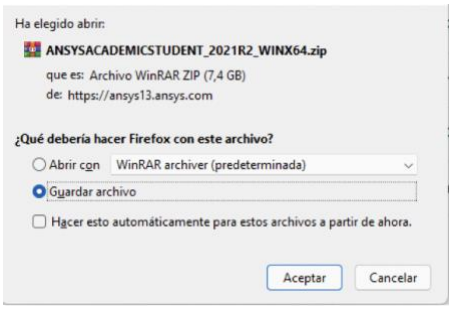
1.4 Ansys

Descarga de Ansys

Para la descarga de la versión free suministrada por Ansys debe ingresar al siguiente enlace: https://www.ansys.com/academic/students/ansys-student

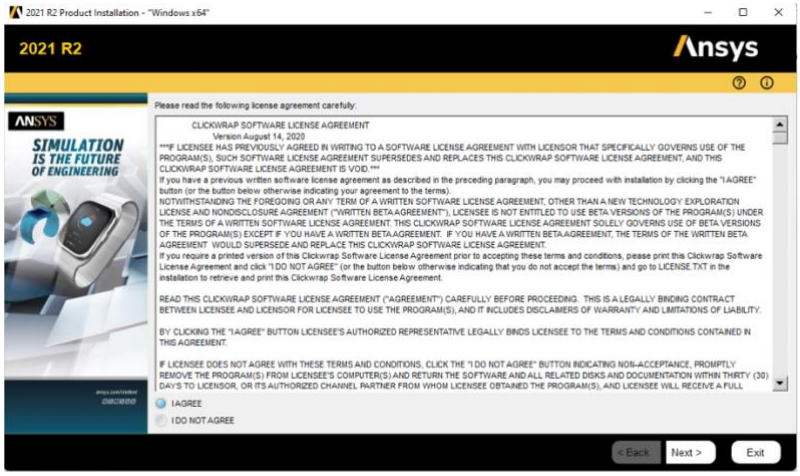
Una vez allí, dar click en el botón **DOWNLOAD ANSYS STUDENT 2021 R2**

En la ventana que se despliega elegir la opción **Guardar archivo** y luego dale **Aceptar**

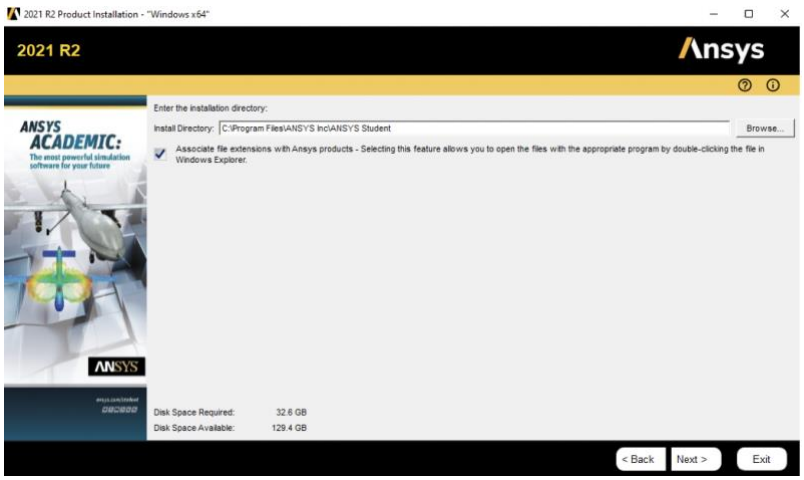


Una vez finalizada la descarga, abre el **.zip** que se ha descargado previamente, una vez abierto ubicar el archivo **setup.exe**

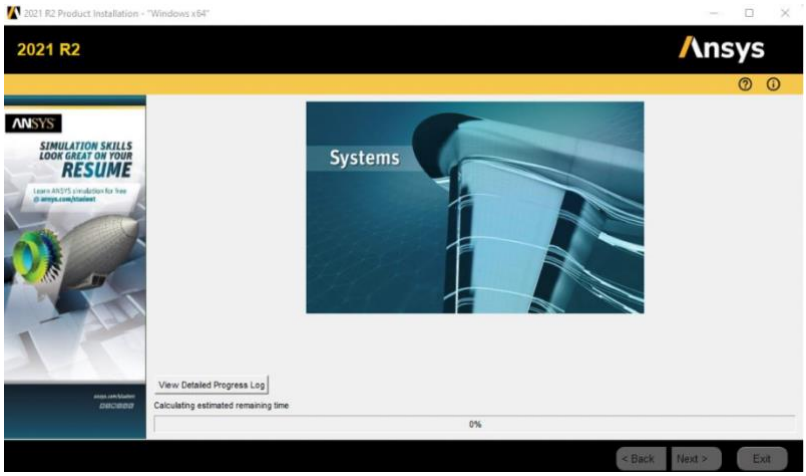
Dale doble click sobre **setup.exe** y espera a que se inicie el instalador. En la ventana de permisos de administrador elige la opción **si ,** y en la siguiente ventana elegir **I AGREE** seguido de **Next >**



En la siguiente ventana, se elige la ubicación en donde se van a alojar los archivos de la instalación. Este apartado se deja de manera predeterminada en la ubicación sugerida y nuevamente **Next> .**



Luego se preparará la instalación



1.5 Introducción a PyMAPDL

PyMAPDL es una interfaz de Python para ANSYS Mechanical APDL, ofreciendo una puerta a la automatización de tareas de simulación, el análisis y post-procesamiento de datos. Esto se logra a través de la combinación de la poderosa funcionalidad de simulación de ANSYS con la flexibilidad y las amplias capacidades de análisis de datos de Python.

Instalación y Configuración Inicial

```python
pip3install ansys-mapdl-core
```

Esta sencilla línea de comando instala PyMAPDL junto con sus dependencias, preparando tu entorno para conectarse y automatizar ANSYS.

Estableciendo la Conexión con ANSYS MAPDL

La conexión con una instancia de ANSYS MAPDL es el primer paso para automatizar tareas. PyMAPDL permite iniciar ANSYS en modo servidor o conectarse a una instancia existente:

```python
from ansys.mapdl.core import launch_mapdl

mapdl = launch_mapdl()
print(mapdl)  # Verifica la conexión y muestra la versión de MAPDL
```

Este código inicia ANSYS MAPDL en segundo plano y establece una conexión, permitiéndote enviar comandos y recibir resultados directamente desde scripts de Python.

Ejercicio análisis estático de una escuadra

Descripción del problema

Se trata de un sencillo análisis estático estructural de una escuadra angular con un solo paso de carga. El orificio del pasador superior izquierdo está constreñido (soldado) en toda su circunferencia y se aplica una carga de presión cónica a la parte inferior del orificio del pasador inferior derecho. Se utiliza el sistema de unidades estadounidense. El objetivo es demostrar cómo se utiliza típicamente el APDL mecánico en un análisis.

Los objetivos de aprendizaje de este ejemplo son:

- Ejecutar PyMAPDL en una máquina local.
- Configurar y resolver un modelo paramétrico utilizando PyMAPDL
- Trazado interactivo de CAD, malla y resultados en la interfaz Pythonica.

Diseño del modelo

Las dimensiones de la escuadra se muestran en la siguiente figura. El soporte es de acero A36 con un módulo de Young de $3*10^7$ [psi] y una relación de Poisson de 0.27.

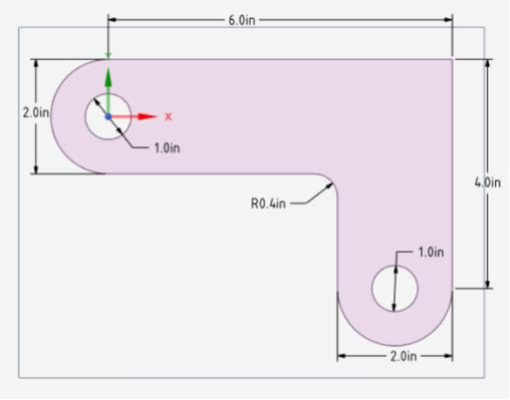


Figura 15.Dimensiones del modelo de soporte.

Parámetros del modelo:

Longitud, anchura y profundidad de la placa
Número de orificios
Raid de los agujeros
Propiedades del material (módulo de Young y coeficiente de Poisson)
Presión aplicada

Planteamiento e hipótesis

Dado que el soporte es delgado en la dirección z (1/2 pulgada de grosor) en comparación con sus dimensiones x e y, y dado que la carga de presión actúa sólo en el plano x-y, asuma la tensión plana para el análisis.

El enfoque consiste en utilizar el modelado sólido para generar el modelo 2D y mallarlo automáticamente con nodos y elementos. Una alternativa sería crear los nodos y elementos directamente.

Inicio de MAPDL

```python
from ansys.mapdl.core import launch_mapdl

mapdl = launch_mapdl()
print(mapdl)  # Verifica la conexión y muestra la versión de MAPDL
```

```

**Construcción de la geometría**

**Definir los rectángulos**

Existen varias formas de crear la geometría del modelo dentro de Mechanical APDL, y algunas son más convenientes que otras. El primer paso es reconocer que puede construir el soporte fácilmente con combinaciones de primitivas de rectángulos y círculos.

Seleccione una ubicación de origen global arbitraria y, a continuación, defina las primitivas de rectángulo y círculo en relación con ese origen. Para este análisis, utilice el centro del agujero superior izquierdo. Comience definiendo un rectángulo relativo a esa ubicación.

El comando mapdl.prep7() de APDL se utiliza para crear un rectángulo con las dimensiones X1, X2, Y1 e Y2. En PyMAPDL la clase mapdl() se utiliza para llamar al comando APDL.

**Caja de dimensiones 1**

Ingresa los siguientes valores :

```python
box1 = [0,6,-1,1]
```

Caja de dimensión 2

```python
box2= [0,6,-1,1]
```

El comando mapdl.prep7() inicia el preprocesador APDL para comenzar la construcción del análisis. Este es el procesador donde se crea la geometría del modelo.

```python
mapdl.prep7()
```

Parametrizar tanto como sea posible, aprovechando las características de Python como la lista de Python o la clase dict. Una buena práctica sería tener todos los parámetros cerca o en la parte superior del archivo de entrada. Sin embargo, para este tutorial interactivo, están en línea.

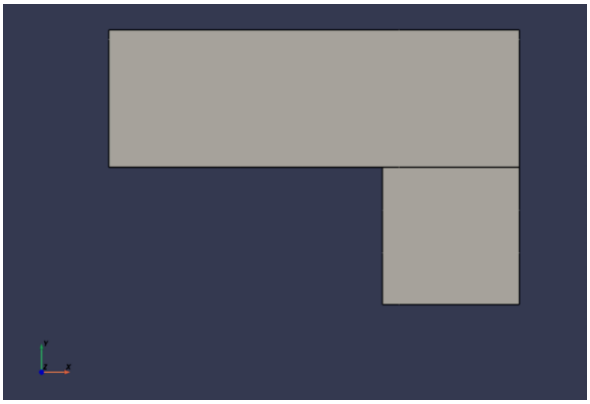
```python
mapdl.rectng(box1[0], box1[1], box1[2], box1[3])
```

Superficies

Los trazados de PyMAPDL pueden controlarse mediante argumentos pasados a los diferentes métodos de trazado, como mapdl.aplot().

El gráfico de área muestra ambos rectángulos, que son áreas, en el mismo color. Para distinguir más claramente entre áreas, active los números de área. Para más información, consulte el método mapdl.aplot().
```

```python
mapdl.aplot(cpos="xy", show_lines=True)
```



Crear el primer círculo

Con el uso de la lógica y las operaciones geométricas booleanas, puedes utilizar los parámetros geométricos originales (caja1, caja2) para localizar los círculos.

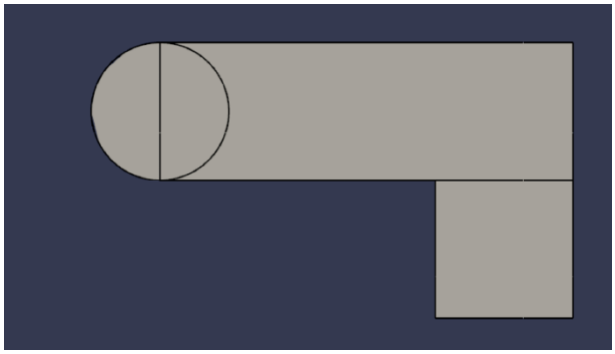
Crea el semicírculo en cada extremo del soporte. En primer lugar, cree un círculo completo en cada extremo y, a continuación, combine los círculos y rectángulos con una operación de suma booleana (comentada en Sustraer agujeros de alfiler del soporte).

El comando APDL para crear los círculos es mapdl.cyl4().

La primera área del círculo se encuentra en el lado izquierdo en la ubicación X,Y, y su radio es 1.

```python
radius = 1
circle1_X = box1[0]
circle1_Y = (box1[2] + box1[3]) / 2
mapdl.cyl4(circle1_X, circle1_Y, radius)

mapdl.aplot(vtk=True, cpos="xy", show_lines=True)
```



Crear un segundo círculo

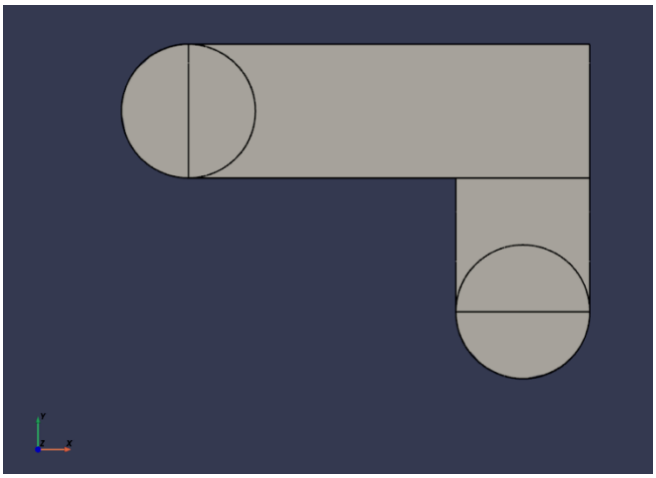
Crea el segundo círculo en la posición X,Y:

```python
circle2_X = (box2[0] + box2[1]) / 2
circle2_Y = box2[3]
```

Utilice estos valores de parámetro para crear la nueva área con el mismo radio de 1 que la primera área circular.

```python
line1 = mapdl.lsel("S", "LOC", "Y", box1[2])
l1 = mapdl.get("line1", "LINE", 0, "NUM", "MAX")
```

Resultado



Añadir las áreas.

Ahora que las piezas apropiadas del modelo (rectángulos y círculos) están definidas, agréguelas para que el modelo se convierta en un área continua. Utilice la operación booleana mapdl.aadd() para sumar las áreas.

Utilice el argumento all para sumar todas las áreas.

```python
mapdl.aadd(“all”)
```

Crear filete de línea

El ángulo recto entre las dos cajas puede mejorarse utilizando un filete con un radio de 0.4. Puede hacerlo seleccionando las líneas alrededor de esta área y creando el filete.

Utilice el método mapdl.lsel() de APDL para seleccionar líneas. Aquí, las posiciones X e Y de las líneas son usadas para crear las cajas para crear su selección.

Después de seleccionar la línea, es necesario escribirla en un parámetro para poder utilizarla para generar la línea de filete. Esto se hace utilizando el método mapdl.get().

Como ha seleccionado una línea, puede utilizar los argumentos MAX y NUM para el método mapdl.get().

Seleccione la primera línea para el filete

```python
line2 = mapdl.lsel("S", "LOC", "X", box2[0])
l2 = mapdl.get("line2", "LINE", 0, "NUM", "MAX")
```

Una vez que tenga ambas líneas seleccionadas, puede utilizar el comando mapdl.lfillt() de PyMAPDL para generar el filete entre las líneas.

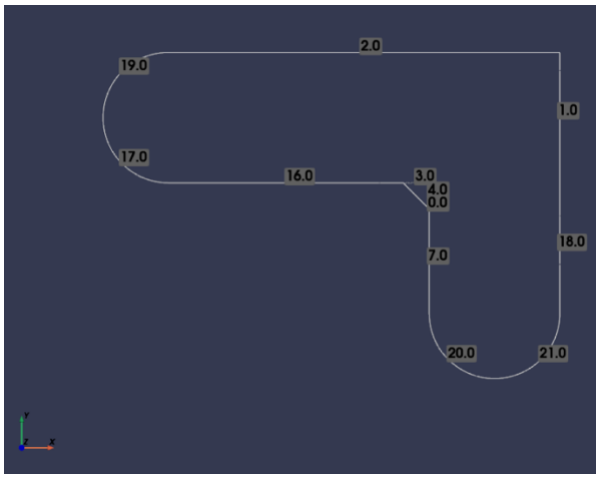
Ten en cuenta que Python podría devolver una lista si hay más de una línea seleccionada.

Aquí se utiliza una mezcla del parámetro APDL como cadena line1 y el parámetro l2 de Python para crear la línea de filete.

Crear línea de filete usando la línea seleccionada (nombres de parámetros).

```python
fillet_radius = 0.4
mapdl.alsel()
line3 = mapdl.lfillt("line1", l2, fillet_radius)

mapdl.alsel()
mapdl.lplot(vtk=True, cpos="xy")
```



Crear área de filete

Para crear el área delineada por la línea1, la línea2 y la recién creada línea3, utilice el método mapdl.al(). Las tres líneas son la entrada. Si las seleccionas todas, puedes utilizar el argumento 'ALL' para crear el área.

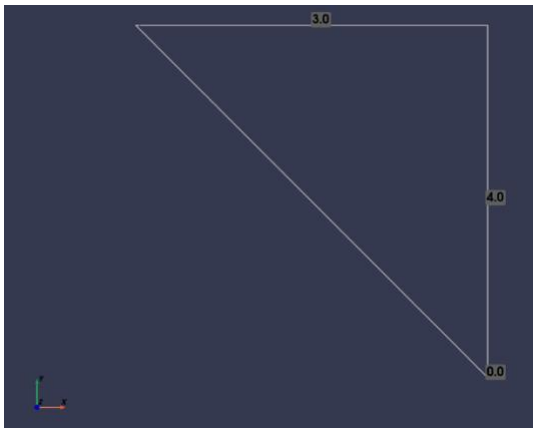
Primero tienes que volver a seleccionar las líneas recién creadas en el área del filete. Para ello, puede utilizar el parámetro radio_fillet y el comando mapdl.lsel().

Para las dos líneas rectas recién creadas, la longitud es la misma que el valor de radio de redondeo. Por lo tanto, puede utilizar el argumento longitud con el comando mapdl.lsel().

```python
mapdl.allsel()
mapdl.lsel("S", "LENGTH", " ", filled radius)
```

Además, es necesario obtener la línea de filete en sí (line3). Puede utilizar el comando mapdl.lsel() de nuevo con el argumento 'RADIUS' si sólo hay una línea con ese radio en el modelo o más directamente utilizar el nombre del parámetro de la línea. Tenga en cuenta la 'A' para seleccionar adicionalmente los elementos.

```python
mapdl.lplot ("A," LINE,,"" line3)
mapdl.lplot(vkt=True, cpos= 'xy", show_line_numbering = True
```

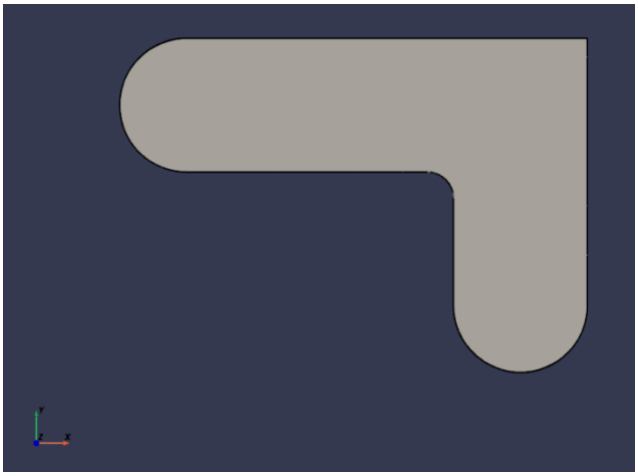


A continuación, utilice el comando mapdl.al() para crear las áreas a partir de las líneas.

```python
mapdl.al("All")
```

Sumar superficies

Añada todas las áreas de nuevo utilizando el método mapdl.aadd(). Como sólo tiene las dos áreas para combinar, utilice el argumento 'ALL'.



Crear el primer agujero

El primer agujero de alfiler está situado a la izquierda de la primera caja. Por lo tanto, puede utilizar las dimensiones de la caja para localizar su nuevo círculo.

El valor X (centro) del agujero de alfiler está en la primera coordenada de la caja1 (X1). El valor Y es la media de los dos valores Y de la caja1:

```python
fillet_radius = 0.4
mapdl.alsel()
line3 = mapdl.lfillt("line1", l2, fillet_radius)

mapdl.alsel()
mapdl.lplot(vtk=True, cpos="xy")
```

Como tienes dos círculos de agujeros de alfiler, utilizas el comando dos veces.

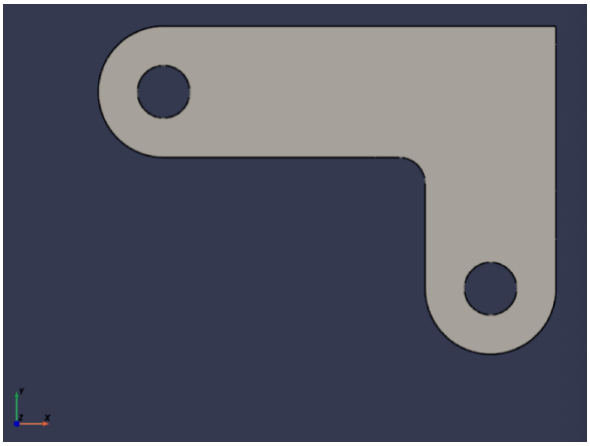
Crear un segundo orificio para el alfiler

El segundo agujero de alfiler se encuentra en la parte inferior de la segunda caja, por lo que de nuevo podemos utilizar las dimensiones de la caja 2 para localizar el círculo. Para este agujero de alfiler las dimensiones son:

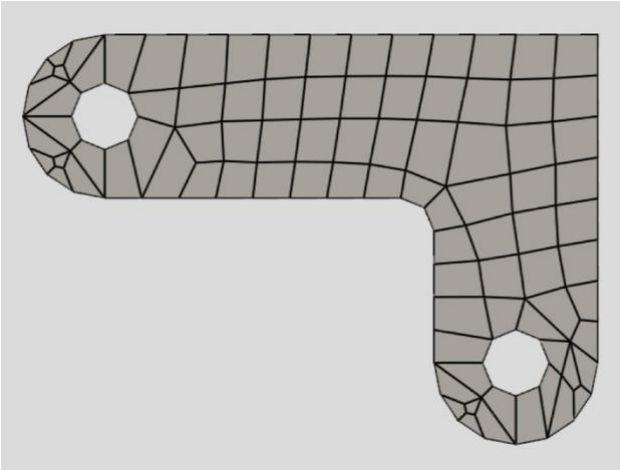
```python
pinhole2_X = (box2[0] + box2[1]) / 2
pinhole2_Y = box2[3]

pinhole2 = mapdl.cyl4(pinhole2_X, pinhole2_Y, pinhole_radius)
pinhole2_lines = mapdl.asl("S", 0)```
```

Resultado final



Reto final: Crea el mallado de la herramienta utilizando la herramienta de PYMAPDL para obtener el resultado de la imagen.



Apéndice B

Archivo fuente Jupyter de Análisis Placa Con Agujeros

Ejercicio placa con agujero

Descripción del problema:

En este ejemplo vamos a configurar un modelo paramétrico en PyMAPDL para una placa rectangular con múltiples agujeros. El modelo está configurado para que se cambien las dimensiones de la placa, el número de agujeros y su radio, las propiedades del material y la presión aplicada.

Diseño del modelo:

Las dimensiones de la escuadra se muestran en la siguiente figura. El soporte es de acero A36 con un módulo de Young de 3*107 [psi] y un coeficiente de Poisson de 0.27.

* Longitud, anchura y profundidad de la placa

* Número de orificios

* Raid de los agujeros

* Propiedades del material (módulo de Young y coeficiente de Poisson)

* Presión aplicada

Paso 1 - Definir todos los parámetros

```python
# Definimos todos los parámetros necesarios (m, Kg, s)
LENGTH = 0.5 # Longitud de la placa en metros
WIDTH = 0.25 # Ancho de la placa en metros
DEPTH = 0.1 # Profundidad de la placa en metros
# Datos del agujero
RADIUS = 0.5 # Radio
NUM = 3 # Numero de agujeros
E = 2e11 # Módulo de elasticidad en Pa (kg/(m*s**2))
NU = 0.27 # Coeficiente de Poisson
PRESSURE = 1000
```

Paso 2 - Correr MAPDL y crear la geometria

```python
# Importamos la librería necesaria para lanzar ANSYS MAPDL
from ansys.mapdl.core import launch_mapdl
# Lanzamos MAPDL C:\Program Files\ANSYS Inc\ANSYS Student\v242\ANSYS
mapdl = launch_mapdl('C:/Program Files/ANSYS Inc/ANSYS Student/v242/ANSYS/bin/win64/ANSYS242.exe')
```

```
mapdl.clear()
# Configuramos y lanzamos la geometría de la placa
mapdl.prep7()  # Preparar el procesador de pre-procesamiento para recibir comandos
mapdl.block(0, LENGTH, 0, WIDTH, 0, DEPTH) # Crear un bloque que representa la placa
for i in range(1,NUM+1):
   mapdl.cyl4(i*LENGTH/(NUM+1),WIDTH/2,RADIUS,",",",2*DEPTH)  # Crear un cilindro para el agujero
mapdl.vsbv(1,'all') # Realizar una operación booleana para sustraer el cilindro del bloque
mapdl.vplot('all')
```

Paso 3 - Definir las propiedades del material, atributos del mallado y generar la malla.

```python
# Definir el tipo de material y las propiedades mecánicas
mapdl.mp('ex',1,E) # Módulo de elasticidad en Pascal (Acero)
mapdl.mp('nuxy',1,NU) # Coeficiente de Poisson
mapdl.et(1,'SOLID186') # Elemento de 2D estructural
# Malla del modelo
mapdl.lesize("ALL", 0.15, layer1=1) # Establecer el tamaño del elemento de la malla
mapdl.mshape(1, "3D")
mapdl.mshkey(0)
mapdl.vmesh('all') # Aplicar la malla al modelo
```

```
mapdl.eplot()
```

Paso 4 - Aplicar las cargas y condiciones de contorno

```python
# Aplicamos condiciones de frontera
mapdl.nsel('s', 'loc', 'x', 0)  # Seleccionar nodos en la posición x=0
mapdl.d('all', 'all', 0)  # Fijar todos los grados de libertad en los nodos seleccionados
mapdl.nsel('s','loc','x', LENGTH)
mapdl.sf('all','pres',PRESSURE)
mapdl.allsel()
mapdl.finish()
mapdl.eplot(plot_bc=True, title="Condiciones de cortono")
```

Paso 5 - Solucionar el problema estatico.

```python
# Lanzamos la solución
mapdl.slashsolu()  # Acceder al solucionador de ANSYS
mapdl.solve()  # Resolver el análisis
mapdl.finish()  # Finalizar el solucionador
# Entrar en la rutina del solucionador y resolver
mapdl.slashsolu()
output = mapdl.solve()
print(output)
```

```

## **Paso 6 - Mostrar los resultados**

```python
# Extraemos resultados y mostramos contornos de tensión
result = mapdl.result  # Obtener el objeto de resultados
result.plot_principal_nodal_stress(0, 'SEQV', background='w', show_edges=True, text_color='k', add_text=True)
```

Salir de MAPDL

```python
mapdl.exit()
```
```

Apéndice C

Archivo fuente Jupyter de análisis estático de un soporte de esquina.

Descripción del problema

Este es un análisis estático estructural simple de un paso de carga de un soporte de ángulo de esquina. El agujero del pasador superior izquierdo está restringido (soldado) alrededor de toda su circunferencia, y se aplica una carga de presión cónica en la parte inferior del agujero del pasador inferior derecho. Se utiliza el sistema de unidades estadounidense. El objetivo es demostrar cómo se utiliza típicamente Mechanical APDL en un análisis.

Modelo del soporte

Las dimensiones del soporte de esquina se muestran en la siguiente figura. El soporte está hecho de acero A36 con un módulo de Young de 3 · 10^7 psi y una relación de Poisson de 0.27.

Enfoque y suposiciones

Dado que el soporte es delgado en la dirección z (1/2 pulgada de grosor) en comparación con sus dimensiones x e y, y dado que la carga de presión actúa solo en el plano x-y, se asume un estrés plano para el análisis. El enfoque es utilizar modelado sólido para generar el modelo 2D y mallas automáticas con nodos y elementos. Una alternativa sería crear los nodos y elementos directamente.

Iniciar MAPDL

```python
from ansys.mapdl.core import launch_mapdl
```

Titulo = "soporte" # opcional

mapdl = launch_mapdl(jobname=Titulo)

```

# **Construcción de la geometría**

## Definir rectángulos

Existen varias maneras de crear la geometría del modelo dentro de Mechanical APDL, y algunas son más convenientes que otras. El primer paso es reconocer que puedes construir el soporte fácilmente con combinaciones de rectángulos y primitivas circulares.

Selecciona una ubicación de origen global arbitraria y luego define las primitivas de rectángulo y círculo en relación con ese origen. Para este análisis, utiliza el centro del agujero superior izquierdo. Comienza definiendo un rectángulo en relación con esa ubicación.

El comando APDL `mapdl.prep7()` se utiliza para crear un rectángulo con las dimensiones `X1`, `X2`, `Y1` y `Y2`. En PyMAPDL, la clase `mapdl()` se utiliza para invocar el comando APDL.

### Dimensiones de caja 1

Introduce lo siguiente:

```python

X1 = 0

X2 = 6

Y1 = -1

Y2 = 1

```python

box1 = [0, 6, -1, 1]
```

```

### Dimensiones de caja 1

```python
box2 = [4, 6, -1, -3]
```

Iniciar el preprocesador APDL

El comando `mapdl.prep7()` inicia el preprocesador APDL para comenzar la construcción del análisis. Este es el procesador donde se crea la geometría del modelo.

Uso de `mapdl.prep7()`

Este comando es fundamental en el proceso de configuración de cualquier análisis en Mechanical APDL porque establece el entorno necesario para la definición de geometrías, propiedades de materiales, y condiciones de contorno, entre otros aspectos. Al invocar `mapdl.prep7()`, se prepara el software para recibir y procesar instrucciones específicas de modelado y simulación.

Ejemplo de Activación del Preprocesador

Para comenzar a definir la geometría de cualquier modelo, como en el caso de nuestro soporte de esquina, primero necesitamos asegurarnos de que estamos en el modo adecuado. Aquí está cómo activar el preprocesador:

```python
# Activar el modo preprocesador en PyMAPDL
mapdl.prep7()
```

Parametrización en PyMAPDL
```

La parametrización de su modelo utilizando características de Python como las clases `list` o `dict` es una práctica recomendada para mantener el código organizado y fácilmente ajustable. Idealmente, todos los parámetros deberían estar ubicados cerca o al principio del archivo de entrada. Sin embargo, para este tutorial interactivo, los incluiremos en línea.

Beneficios de la Parametrización

1. **Modularidad**: Facilita la modificación de los valores sin necesidad de buscar y reemplazar múltiples instancias a lo largo del código.

2. **Claridad**: Mejora la legibilidad del código al separar los parámetros de las operaciones.

3. **Reutilización**: Permite reutilizar el mismo script para diferentes configuraciones de análisis con mínimos ajustes.

Implementación con Listas y Diccionarios

Uso de Listas

Las listas de Python son ideales para almacenar valores que son parte de una secuencia o que no requieren etiquetas identificativas.

```python
# Definir dimensiones del soporte como lista
bracket_dimensions = [0, 6, -1, 1]  # X1, X2, Y1, Y2
# Crear un rectángulo utilizando las dimensiones especificadas
mapdl.rectng(*bracket_dimensions)
```

```python
# Construir las cajas
```

mapdl.rectng(box1[0], box1[1], box1[2], box1[3])

```

## **Uso del operador `*` en Python para desempacar objetos**

En Python, puedes usar el operador `*` para desempacar un objeto al realizar una llamada a una función. Esto es útil cuando trabajas con listas o tuplas con múltiples valores que deseas pasar como argumentos individuales a una función.

```python

mapdl.rectng(*box2)

```

Graficar áreas

Los gráficos de PyMAPDL se pueden controlar mediante argumentos pasados a los diferentes métodos de gráficos, como `mapdl.aplot()`.

El gráfico de áreas muestra ambos rectángulos, que son áreas, en el mismo color. Para distinguir más claramente entre las áreas, activa los números de área. Para obtener más información, consulta el método `mapdl.aplot()`.

```python

mapdl.aplot(cpos="xy", show_lines=True)

```

Crear el primer círculo

Con el uso de lógica y operaciones geométricas Booleanas, puedes utilizar los parámetros geométricos originales (`box1`, `box2`) para ubicar los círculos.
```

Crea el semicírculo en cada extremo del soporte. Primero, crea un círculo completo en cada extremo y luego combina los círculos y rectángulos con una operación de adición Booleana (discutido en [Subtract pin holes from bracket]()).

El comando APDL para crear los círculos es `mapdl.cyl4()`.

El área del primer círculo se encuentra en el lado izquierdo en la ubicación X,Y, y su radio es 1.

```python
# Crear el primer círculo
radius = 1
circle1_X = box1[0]
circle1_Y = (box1[2] + box1[3]) / 2
mapdl.cyl4(circle1_X, circle1_Y, radius)


mapdl.aplot(vtk=True, cpos="xy", show_lines=True)
```

Crear el segundo círculo

Crea el segundo círculo en la ubicación X,Y:

```python
circle2_X = (box2[0] + box2[1]) / 2
circle2_Y = box2[3]
```

Usa estos valores de parámetros para crear el área nueva con el mismo radio de 1 que el área del primer círculo.

```python
mapdl.cyl4(circle2_X, circle2_Y, radius)
mapdl.aplot(vtk=True, cpos="xy", show_lines=True)
```

Agregar áreas

Ahora que las piezas apropiadas del modelo (rectángulos y círculos) están definidas, agrégalas juntas para que el modelo se convierta en un área continua. Usa la operación de adición Booleana `mapdl.aadd()` para agregar las áreas juntas.

Usa el argumento `all` para agregar todas las áreas.

```python
mapdl.aadd("all") # Agregar todas las áreas definidas juntas
```

Crear chaflán en línea

El ángulo recto entre las dos cajas puede mejorarse utilizando un chaflán con un radio de 0.4. Puedes hacer esto seleccionando las líneas alrededor de esta área y creando el chaflán.

Usa el método `mapdl.lsel()` para seleccionar líneas. Aquí, las ubicaciones X y Y de las líneas se usan para crear las cajas para tu selección.

Después de seleccionar la línea, necesitas escribirla en un parámetro para que puedas usarla para generar la línea de chaflán. Esto se hace utilizando el método `mapdl.get()`.

Debido a que has seleccionado una línea, puedes usar los argumentos `MAX` y `NUM` para el método `mapdl.get()`.

Seleccionar la primera línea para el chaflán:

```python
```

Seleccionar la primera línea para el chaflán

line1 = mapdl.lsel("S", "LOC", "Y", box1[2])

l1 = mapdl.get("line1", "LINE", 0, "NUM", "MAX")

```

Si escribes el comando en un parámetro de Python (`line1`), puedes usar

ya sea el parámetro APDL `l1` o el parámetro de Python `line1` cuando

crees la línea de chaflán.

Seleccionar la segunda línea para el chaflán y crear el parámetro de Python

```python

line2 = mapdl.lsel("S", "LOC", "X", box2[0])

l2 = mapdl.get("line2", "LINE", 0, "NUM", "MAX")

```

Una vez que tengas ambas líneas seleccionadas, puedes usar el comando de PyMAPDL `mapdl.lfillt()` para generar el chaflán entre las líneas.

Nota que Python podría devolver una lista si se selecciona más de una línea.

Aquí usas una mezcla del parámetro APDL como una cadena `line1` y el parámetro Python `line2` para crear la línea de chaflán.

Crear línea de chaflán usando la línea seleccionada (nombres de parámetros)

```python

fillet_radius = 0.4

mapdl.allsel()

line3 = mapdl.lfillt("line1", l2, fillet_radius)

```
mapdl.allsel()
mapdl.lplot(vtk=True, cpos="xy")
```

Crear área del chaflán

Para crear el área delimitada por `line1`, `line2` y la recién creada `line3`, utiliza el método `mapdl.al()`. Las tres líneas son la entrada. Si las seleccionas todas, puedes usar el argumento `'ALL'` para crear el área.

Primero, debes volver a seleccionar las líneas recién creadas en el área del chaflán. Para hacer esto, puedes usar el parámetro `fillet_radius` y el comando `mapdl.lsel()`.

Para las dos líneas rectas recién creadas, la longitud es la misma que el valor de `fillet_radius`. Por lo tanto, puedes usar el argumento de longitud con el comando `mapdl.lsel()`.

```python
mapdl.allsel()
# Seleccionar línea para el chaflán
mapdl.lsel("S", "LENGTH", "", fillet_radius)
```

Además, necesitas obtener la línea del chaflán en sí (`line3`). Puedes usar el comando `mapdl.lsel()` nuevamente con el argumento `'RADIUS'` si hay solo una línea con ese radio en el modelo o, más directamente, usar el nombre del parámetro de la línea. Nota el uso de `'A'` para seleccionar elementos adicionalmente.

```python
mapdl.lsel("A", "LINE", "", line3)
# Mostrar areas
```

mapdl.lplot(vtk=True, cpos="xy", show_line_numbering=True)

```

Luego usa el comando `mapdl.al()` para crear las áreas a partir de las líneas.

```python
# Crear el área usando las líneas seleccionadas
mapdl.al("ALL")
```

Agregar áreas juntas

Une todas las áreas nuevamente usando el método `mapdl.aadd()`. Dado que solo tienes dos áreas para combinar, usa el argumento `'ALL'`.

```python
# Agregar todas las áreas juntas usando el argumento 'ALL'
mapdl.aadd("all")
mapdl.aplot(vtk=True, cpos="xy", show_lines=True)
```

Crear el primer agujero de pasador

El primer agujero de pasador se encuentra en el lado izquierdo de la primera caja. Por lo tanto, puedes usar las dimensiones de la caja para ubicar tu nuevo círculo.

El valor X (centro) del agujero de pasador está en la primera coordenada de la `box1` (`X1`). El valor Y es el promedio de los dos valores Y de la `box1`:

```python
# Crear el primer agujero de pasador con un radio especificado
pinhole_radius = 0.4
```
```

```
pinhole1_X = box1[0]
pinhole1_Y = (box1[2] + box1[3]) / 2
pinhole1 = mapdl.cyl4(pinhole1_X, pinhole1_Y, pinhole_radius)
```

Crear el segundo agujero de pasador

El segundo agujero de pasador se encuentra en la parte inferior de la segunda caja, por lo que nuevamente podemos usar las dimensiones de la caja 2 para ubicar el círculo. Para este agujero de pasador, las dimensiones son:

```python
pinhole2_X = (box2[0] + box2[1]) / 2
pinhole2_Y = box2[3]
pinhole2 = mapdl.cyl4(pinhole2_X, pinhole2_Y, pinhole_radius)
pinhole2_lines = mapdl.asll("S", 0)
```

Restar agujeros de pasador del soporte

Si usas el comando `mapdl.aplot()` con líneas, en este punto habrás creado dos áreas circulares que se superponen con el soporte. Puedes usar el comando `mapdl.asba()`, que es el comando Booleano para restar áreas, para eliminar los círculos del soporte.

```python
# Restar las áreas de los agujeros de pasador del soporte
mapdl.asba("all", pinhole1)
bracket = mapdl.asba("all", pinhole2)
mapdl.aplot(vtk=True, show_lines=True, cpos="xy")
```

```

**Definición del modelo**

Definir propiedades del material

Solo hay una propiedad de material que definir para el soporte, acero A36, con valores dados para el módulo de elasticidad de Young y la relación de Poisson.

```python
ex = 30e6  # Módulo de elasticidad de Young en psi
poissons_ratio = 0.27  # Relación de Poisson
# Asignar las propiedades del material en PyMAPDL
mapdl.mp("EX", 1, ex)
mapdl.mp("PRXY", 1, poissons_ratio)
```

Definir tipos de elementos y opciones

Usa el comando `mapdl.et()` para seleccionar un elemento.

En cualquier análisis, seleccionas elementos de una biblioteca de tipos de elementos y defines los apropiados para el análisis. En este caso, solo se utiliza un tipo de elemento: [PLANE183](), un elemento estructural 2D, cuadrático y de orden superior.

Un elemento de orden superior te permite tener una malla más gruesa que con elementos de orden inferior, manteniendo aún la precisión de la solución. Además, Mechanical APDL genera algunos elementos en forma de triángulo en la malla que, de lo contrario, serían inexactos al usar elementos de orden inferior.

Opciones para [PLANE183]
```

Especifica esfuerzo plano con espesor como una opción para [PLANE183](). (El espesor se define como una constante real en [Definir constantes reales]()). Selecciona esfuerzo plano con la opción de espesor para el comportamiento del elemento. La opción de espesor se establece usando la keyoption 3 del elemento. Para obtener más información, consulta la definición del elemento [PLANE183]() en la ayuda de Ansys.

```python
# Seleccionar el tipo de elemento PLANE183
mapdl.et(1, "PLANE183", kop3=3)# Establecer la opción de espesor usando keyoption 3. Opción de comportamiento de esfuerzo plano con espesor
```

Definir constantes reales

Asumiendo esfuerzo plano con espesor, introduce el espesor como una constante real para [PLANE183]().

Usa el comando `mapdl.r()` para establecer constantes reales.

```python
# Definir el espesor como una constante real para PLANE183
thick = 0.5
mapdl.r(1, thick)
```

ACTIVIDAD

Malla

Puedes mallar el modelo sin especificar controles de tamaño de malla. Si no estás seguro de cómo determinar la densidad de la malla, puedes permitir que Mechanical APDL aplique una

malla predeterminada. Sin embargo, para este modelo, deseas especificar un tamaño de elemento global para controlar la densidad general de la malla. Establece el control de tamaño global usando el comando `mapdl.esize()`. Establece un tamaño de 0.5 o un valor ligeramente menor para mejorar la malla.

Malla las áreas usando el comando `mapdl.amesh()`. Tu malla puede variar ligeramente de la malla mostrada. Puedes ver resultados ligeramente diferentes durante el postprocesamiento.

Ahora puedes usar el comando `mapdl.eplot()` para ver la malla.

```python

```

Condiciones de contorno

La carga se considera parte del comando `mapdl.solu()` o del procesador de soluciones en APDL. Pero también se puede hacer en el preprocesador con el comando `mapdl.prep7()`.

Puedes activar el procesador de soluciones llamando a la clase `mapdl.solution()`, usando el comando `mapdl.slashsolu()`, o usando `mapdl.run("/solu")` para llamar al comando APDL `/SOLU`.

```python
mapdl.allsel()
mapdl.solution()
```

Establecer el tipo de análisis

Establece el tipo de análisis con el comando `mapdl.antype()`.

```python
mapdl.antype("STATIC")
```

Aplicar restricciones de desplazamiento

Aquí agregas las condiciones de contorno al modelo. Primero, deseas fijar el modelo estableciendo un desplazamiento cero en el primer agujero de pasador. Puedes aplicar restricciones de desplazamiento directamente a las líneas.

Para hacer esto sin la interfaz gráfica, necesitarías volver a trazar las líneas. O puedes usar operaciones booleanas y generar las líneas a partir de las ubicaciones de los agujeros de pasador/parámetros de la caja. Usando los parámetros que has creado, puedes seleccionar las líneas y fijar un extremo del soporte.

Selecciona las cuatro líneas alrededor del agujero izquierdo usando el comando `mapdl.lsel()` y los parámetros `pinehole1`.

```python
bc1 = mapdl.lsel(
    "S", "LOC", "X", pinhole1_X - pinhole_radius, pinhole1_X + pinhole_radius)
print(f"Number of lines selected : {len(bc1)}")
```

Luego, para la carga, selecciona y aplica la condición de contorno a los nodos adjuntos a esas líneas usando el comando `mapdl.nsll()`.

```python
fixNodes = mapdl.nsll(type_="S")
```

```

A continuación, usa el comando `mapdl.d()` para establecer el desplazamiento a cero (restricción fija).

```python
# Configurar condiciones de contorno
mapdl.d("ALL", "ALL", 0) # El 0 no es necesario ya que el valor predeterminado es cero
# Seleccionar todo de nuevo
mapdl.allsel()
```

Aplicar carga de presión

Aplica la carga de presión cónica al agujero de pasador inferior derecho. En este caso, cónica significa que varía linealmente. Cuando se crea un círculo en Mechanical APDL, cuatro líneas definen el perímetro; por lo tanto, aplica la presión a dos líneas que forman la mitad inferior del círculo. Debido a que la presión varía desde un valor máximo (500 psi) en la parte inferior del círculo hasta un valor mínimo (50 psi) en los lados, aplica la presión en dos pasos separados, con valores de conicidad inversa para cada línea.

```python
p1 = 50
p2 = 500
```

Aplicar carga de presión

La convención de Mechanical APDL para la carga de presión es que un valor de carga positivo representa presión hacia la superficie (compresiva).
```

Para seleccionar la línea, usa el mismo comando `mapdl.lsel()` utilizado en el bloque de código anterior y luego convierte las líneas a una selección nodal con el comando `mapdl.nsel()`.

Nota que tenemos un procedimiento de selección ligeramente más complicado para los dos cuartos del círculo completo. Un método para seleccionar las líneas sería seleccionar la mitad inferior del círculo del segundo agujero de pasador.

```python
mapdl.lsel("S", "LOC", "Y", pinhole2_Y - pinhole_radius, pinhole2_Y)
```

Ahora vuelve a seleccionar de esa selección las líneas que están a la izquierda del centro X de ese agujero de pasador.

```python
mapdl.lsel("R", "LOC", "X", 0, pinhole2_X)
mapdl.lplot(vtk=True, cpos="xy")
```

Una vez que tengas la línea correcta, usa el comando `mapdl.sf()` para cargar la línea con la carga superficial variable.

```python
# Aquí se carga el lado izquierdo de la mitad inferior del segundo agujero de alfiler.
mapdl.sf("ALL", "PRES", p1, p2)
mapdl.allsel()
```

Repita el procedimiento para el segundo orificio del pasador.

```python
```

```
mapdl.lsel("S", "LOC", "Y", pinhole2_Y - pinhole_radius, pinhole2_Y)
mapdl.lsel("R", "LOC", "X", pinhole2_X, pinhole2_X + pinhole_radius)
mapdl.lplot(
    vtk=True,
    cpos="xy",
    show_line_numbering=True,
)
mapdl.sf("ALL", "PRES", p2, p1)
mapdl.allsel()
```

Solución

Para resolver un análisis de Ansys FE, debe activarse el procesador de soluciones, utilizando el comando `mapdl.solution() <ansys.mapdl.core.solution.Solution>`{.interpreted-text role="class"} o la clase mapdl.slashsolu() <ansys.mapdl.core.Mapdl.slashsolu>`{.interpreted-text role="meth"}. Esto se hizo unos pasos antes.

El modelo está listo para ser resuelto utilizando el comando `mapdl.solve() <ansys.mapdl.core.Mapdl.solve>`{.interpreted-text role="meth"}.

```python
# Resolver el modelo
output = mapdl.solve()
print(output)
```

Mechanical APDL almacena los resultados de este problema de un solo paso de carga en la base de datos y en el archivo de resultados, `Jobname.RST`{.interpreted-text role="file"} (o `Jobname.RTH`{.interpreted-text role="file"} para problemas térmica o `Jobname.RMG`{.interpreted-text role="file"} para magnética).

La base de datos sólo puede contener un conjunto de resultados en un momento dado, por lo que en un análisis de múltiples pasos de carga o múltiples subpasos, Mechanical APDL sólo almacena la solución final en la base de datos.

Mechanical APDL almacena todas las soluciones en el archivo de resultados.

Revise los resultados

Este paso representa el inicio de la fase de postprocesamiento.

Nota: Los resultados que vea varían ligeramente de lo que se muestra por variaciones en la malla.

Entra en el postprocesador

El postprocesador APDL de Ansys es un procesador independiente llamado con el comando `mapdl.post1() <ansys.mapdl.core.Mapdl.post1>`{.interpreted-text role="meth"}.

```python
mapdl.post1()
```

Trazar la forma deformada

Aquí `mapdl.result <ansys.mapdl.core.Mapdl.result>`{.interpreted-text role="class"} se utiliza para recuperar los resultados y para el trazado.

```python
# Trazar desplazamiento
```

```
result = mapdl.result
result_set = 0 # Trazar los primeros resultados
disp_fact = 1e10
result.plot_nodal_displacement(
    result_set,
    cpos="xy",
    displacement_factor=5,
    show_displacement=True,
    show_edges=True,
)
```

Trazar la tensión equivalente de von Mises

También puedes generar gráficos de tensiones utilizando la función `mapdl.plot_principal_nodal_stress() <ansys.mapdl.core.Mapdl.plot_principal_nodal_stress>`{.interpreted-text role="meth"} comando.

```python
result.plot_principal_nodal_stress(
    0,
    "SEQV",
    cpos="xy",
    background="w",
    text_color="k",
```

```
    add_text=True,
    show_edges=True,)
```

Obtener las tensiones de von Mises.

```python
nnum, stress = result.principal_nodal_stress(0)
# La tensión de Von Mises es la última columna de los resultados de tensión
von_mises = stress[:, -1]
```

Lista solución de reacción

Para listar las fuerzas de reacción FY utilice el APDL

`mapdl.prrsol() <ansys.mapdl.core.Mapdl.prrsol>`{.interpreted-text role="meth"} comando que imprime la solución de reacción del nodo restringido.

Puede utilizar el comando to_dataframe <ansys.mapdl.core.commands.CommandListingOutput>`{.interpreted-text role="meth"} para convertir la salida en un marco de datos y obtener más información. impresión estática:

```python
reactForces = mapdl.prrsol(lab="FY").to_dataframe(columns=["NODE", "FY"])
print(reactForces)
```

Salida Mecánica APDL

```python
mapdl.exit()
```

```

## Apéndice D

## Archivo fuente Jupyter de Análisis Estructural de un Cortador de Torno con ANSYS MAPDL

### Objetivo

En este notebook, se realizará un análisis estructural detallado de un cortador de torno utilizando ANSYS Mechanical APDL (MAPDL). Este tipo de análisis es crucial para prever el comportamiento del cortador bajo condiciones de carga reales y asegurar su rendimiento y seguridad en operaciones de manufactura.

### Contenido

- Modelar geométricamente un cortador de torno en ANSYS MAPDL.
- Aplicar condiciones de carga y simular el comportamiento bajo estas cargas.
- Analizar las tensiones y deformaciones resultantes para evaluar la integridad estructural del cortador.
- Visualizar los resultados para una interpretación clara y detallada de las tensiones y deformaciones.

ANSYS MAPDL será utilizado para:

- Importar y preparar la geometría del cortador.
- Definir materiales, propiedades, y tipos de elementos.
- Aplicar condiciones de frontera y cargas.
```

- Resolver el modelo de elementos finitos.

- Post-procesar los resultados para obtener visualizaciones de las tensiones y deformaciones principales.

Paso 1: Configuración Inicial

```python
# Importamos las librerías necesarias
import os
import numpy as np
from ansys.mapdl.core import launch_mapdl
from ansys.mapdl.core.examples.downloads import download_example_data
PI = np.pi
EXX = 1.0e7
NU = 0.27
Fuerza = 10000
```

En este paso inicial, se importan las bibliotecas necesarias y se lanza la sesión de ANSYS MAPDL en modo SMP para aprovechar múltiples núcleos del procesador.

```python
# Configuramos el directorio actual y algunas constantes básicas
path = os.getcwd()  # Obtiene el directorio de trabajo actual
PI = np.pi  # Constante de Pi para cálculos trigonométricos
# Inicializar ANSYS MAPDL
mapdl = launch_mapdl(additional_switches="-smp")  # Lanza MAPDL en modo SMP
```

Reiniciar la base de datos de MAPDL y configurar el entorno

mapdl.clear() # Limpia cualquier configuración anterior en MAPDL

```

**## Paso 2: Importación y Preparación de la Geometría**

```python

# Cargar la geometría del modelo y establecer parámetros

archivo_geometria = download_example_data("LatheCutter.anf", "geometry")

mapdl.input(archivo_geometria)  # Carga la geometría desde un archivo

mapdl.finish()  # Termina el procesamiento del input

longitud_presion = mapdl.parameters["PRESS_LENGTH"]  # Extrae un parámetro de MAPDL

```

Cambias unidades y titulo

```python

mapdl.units("Bin")

mapdl.title("Cortador de Torno")

```

Propiedades del material

```python

mapdl.prep7()  # Prepara el procesador de pre-procesamiento

mapdl.mp("EX", 1, EXX)  # Define el módulo de elasticidad

mapdl.mp("NUXY", 1, NU)  # Define el coeficiente de Poisson

```

El tipo de elemento MAPDL `SOLID285` se utiliza con fines de demostración. Considere utilizar un tipo de elemento o densidad de malla apropiados para su aplicación real.

```python
mapdl.et(1, 285)  # Establece el tipo de elemento a usar
mapdl.smrtsize(4)  # Tamaño inteligente de los elementos
mapdl.aesize(14, 0.0025)  # Tamaño específico para la malla
mapdl.vmesh(1)  # Malla el volumen seleccionado
# Definir condiciones de simetría y cargar la configuración de presión
mapdl.da(11, "symm")  # Aplica condiciones de simetría
# Similar para otros lados o caras del modelo
mapdl.da(16, "symm")
mapdl.da(9, "symm")
mapdl.da(10, "symm")
```

Sistema de coordenadas y carga

Cree un Sistema de Coordenadas local (CS) para la presión aplicada como una función de X local.

El ID del CS local es 11

```python
# Configurar y aplicar la carga de presión
mapdl.cskp(11, 0, 2, 1, 13)  # Crea un sistema de coordenadas local
```
```

```
mapdl.csys(1)  # Cambia al sistema de coordenadas creado
mapdl.view(1, -1, 1, 1)  # Configura la vista en MAPDL
mapdl.psymb("CS", 1)  # Establece símbolos para la vista
mapdl.vplot(
    color_areas=True,
    show_lines=True,
    cpos=[-1, 1, 1],
    smooth_shading=True,
)
```

Los trazados VTK no muestran símbolos de trazado MAPDL. Sin embargo, para utilizar las puede establecer la opción de palabra clave `vtk` en `False`.

```python
mapdl.lplot(vtk=False)
```

Carga de presión

Cree una carga de presión, cárguela en MAPDL como un array de tablas, verifique la y resuelve.

```python
# Define y aplica una carga variable a lo largo de una dimensión
puntos = 10 # Número de puntos para definir la carga
longitud_x = np.linspace(0, longitud_presion, puntos)  # Define la distribución de la carga
```

presion = Fuerza * np.sin(PI * longitud_x / longitud_presion) # Calcula la carga como función del seno

presion = np.stack((longitud_x, presion), axis=-1) # Combina x con la presión

mapdl.load_table("MY_PRESS", presion, "X", csysid=11) # Carga la tabla de presiones en MAPDL

```

`length_x` y `press` son vectores. Para combinarlos en la forma necesaria para definir el array de la tabla MAPDL, puedes utilizar [numpy.stack](https://numpy.org/doc/stable/reference/generated/numpy.stack.html).

```python

# Selecciona áreas y aplica la carga de presión

mapdl.asel("S", "Area", "", 14)

mapdl.nsla("S", 1)

mapdl.sf("All", "Press", "%MY_PRESS%")

mapdl.allsel()  # Selecciona todo de nuevo

```

Configurar la solución.

```python

# Configura y resuelve el modelo

mapdl.finish()  # Finaliza la configuración

mapdl.slashsolu()  # Accede al solucionador

mapdl.nlgeom("On")  # Activa la geometría no lineal si es necesario

mapdl.psf("PRES", "NORM", 3, 0, 1)  # Configura opciones de presión

```
mapdl.solve() # Resuelve el análisis
mapdl.finish() # Finaliza el solucionador
```

Trazado

```python
mapdl.post1()
mapdl.set("last")
mapdl.allsel()
mapdl.post_processing.plot_nodal_principal_stress("1", smooth_shading=False)
```

Trazado - Parte del modelo

```python
mapdl.csys(1)
mapdl.nsel("S", "LOC", "Z", -0.5, -0.141)
mapdl.esln()
mapdl.nsle()
mapdl.post_processing.plot_nodal_principal_stress(
    "1", edge_color="white", show_edges=True
)
```

Trazado - Opciones de leyenda

```python
mapdl.allsel()
```

```
sbar_kwargs = {
    "color": "black",
    "title": "1st Principal Stress (psi)",
    "vertical": False,
    "n_labels": 6,
}
mapdl.post_processing.plot_nodal_principal_stress(
    "1",
    cpos="xy",
    background="white",
    edge_color="black",
    show_edges=True,
    scalar_bar_args=sbar_kwargs,
    n_colors=9,
)
```

Vamos a probar algunas opciones de barra escalar de la [PyVista documentación](). Por ejemplo, pongamos texto negro sobre fondo beige.

Las palabras clave de la barra escalar definidas como un diccionario Python son un método alternativo alternativo al uso de {clave:valor}. Puede utilizar el método de hacer clic y arrastrar para reposicionar la barra escalar. Haga clic con el botón izquierdo del ratón y manténgalo pulsado mientras mueve el ratón.

```python
 sbar_kwargs = dict(
    title_font_size=20,
    label_font_size=16,
    shadow=True,
    n_labels=9,
    italic=True,
    bold=True,
    fmt="%.1f",
    font_family="arial",
    title="1st Principal Stress (psi)",
    color="black",  )
mapdl.post_processing.plot_nodal_principal_stress(
    "1",
    cpos="xy",
    edge_color="black",
    background="beige",
    show_edges=True,
    scalar_bar_args=sbar_kwargs,
    n_colors=256,
    cmap="jet",  )
```

Tratamiento posterior

Lista de resultados

Obtenga todas las tensiones nodales principales.

```python
 mapdl.post_processing.nodal_principal_stress("1")
 ```

Obtiene las tensiones principales nodales del subconjunto de nodos.

```python
 mapdl.nsel("S", vmin=1200, vmax=1210)
 mapdl.esln()
 mapdl.nsle()
print("«Los números de los nodos son:»")
 print(mapdl.mesh.nnum) # obtener los números de los nodos
print("«Las tensiones nodales principales son:»")
 mapdl.post_processing.nodal_principal_stress('1')
 ```

```python
 mapdl.exit()
 ```

Apéndice E

Archivo fuente Jupyter de Simulación estática del ensayo de viga doblemente empotrada mediante elementos cohesivos.

Objetivo

Este ejemplo muestra cómo utilizar PyMAPDL para simular la delaminación en materiales compuestos. También se utilizan los módulos PyDPF para el postprocesamiento de los resultados.

Procedimiento

1. Iniciar la instancia de MAPDL.

2. Configurar el modelo.

3. Resolver el modelo.

4. Graficar los resultados utilizando PyMAPDL.

5. Graficar los resultados utilizando PyDPF.

6. Graficar la fuerza de reacción.

Paquetes adicionales

Estos paquetes adicionales son importados para su uso:

- [Matplotlib](https://matplotlib.org) para graficación

- [Pandas](https://pandas.pydata.org/) para análisis y manipulación de datos

Iniciar MAPDL como un servicio

Este ejemplo comienza importando los paquetes necesarios y luego lanzando Ansys Mechanical APDL.

```python

import os

import tempfile

from ansys.dpf import core as dpf

import matplotlib.pyplot as plt

import numpy as np

import pyvista as pv
```

from ansys.mapdl import core as pymapdl

Iniciar MAPDL

mapdl = pymapdl.launch_mapdl()

print(mapdl)

```

**# Configurar entradas geométricas**

Establecer las entradas geométricas para el modelo.

```python

# longitud, precorte, ancho, altura, desplazamiento; estas son las variables:

longitud = 75.0 # Longitud en milímetros

precorte = 10.0 # Longitud del precorte en milímetros

ancho = 25.0  # Ancho en milímetros

altura = 1.7  # Altura en milímetros

desplazamiento = 10.0 # Desplazamiento en milímetros

# Una pequeña cantidad definida para evitar errores de redondeo al seleccionar entidades geométricas

eps = 1e-1

```

 # Configurar el modelo

Configurar el modelo seleccionando el sistema de unidades y los tipos de elementos para las simulaciones. Dado que se ha elegido un enfoque completamente tridimensional para este
```

ejemplo, se utilizan elementos `SOLID186` para el mallado de volúmenes, y `TARGE170` y `CONTA174` para modelar elementos cohesivos entre superficies de contacto.

Definir parámetros del material

Las placas compuestas se modelan utilizando propiedades ortotrópicas elásticas lineales homogéneas, mientras que para los elementos cohesivos se utiliza una ley cohesiva bilineal.

```python
# Configurar el modelo
mapdl.prep7()
mapdl.units("mpa")
# Definir elementos y tamaños para simulación completamente tridimensional
mapdl.et(1, 185)
mapdl.et(2, 170)
mapdl.et(3, 174)
mapdl.esize(10.0)
# Propiedades del material para las placas compuestas
mapdl.mp("ex", 1, 61340)
mapdl.mp("dens", 1, 1.42e-09)
mapdl.mp("nuxy", 1, 0.1)
# Ley cohesiva bilineal para elementos de contacto
mapdl.mp("mu", 2, 0)
mapdl.tb("czm", 2, 1, "", "bili")
mapdl.tbtemp(25.0)
mapdl.tbdata(1, 50.0, 0.5, 50, 0.5, 0.01, 2)
```

```

**# Crear la geometría en el modelo y el mallado**

Las dos placas se generan como dos paralelepípedos. Luego se asignan las propiedades del material compuesto y los elementos tridimensionales.

```python
# Creación y malla de la geometría
vnum0 = mapdl.block(0.0, longitud + precorte, 0.0, ancho, 0.0, altura)
vnum1 = mapdl.block(0.0, longitud + precorte, 0.0, ancho, altura, 2 * altura)
mapdl.mat(1)
mapdl.type(1)
mapdl.vmesh(vnum0)
mapdl.vmesh(vnum1)
mapdl.eplot()
```

 # Generar elementos cohesivos entre las superficies de contacto

La generación de elementos cohesivos es la parte más delicada del enfoque de modelado. Primero, se identifican y definen las dos superficies de contacto como componentes (en este caso `cm_1` y `cm_2`). Luego, se configuran las constantes reales para los elementos `CONTA174` y `TARGE170`, y sus opciones clave para capturar el comportamiento correcto. Las descripciones de cada uno de estos parámetros se pueden encontrar en la documentación de elementos de Ansys. Finalmente, se generan elementos en las superficies respectivas `cm_1` y `cm_2`.

```python
# Generación de elementos cohesivos entre superficies de contacto
```
```

```
mapdl.allsel()

mapdl.asel("s", "loc", "z", 1.7)

areas = mapdl.geometry.anum

mapdl.geometry.area_select(areas[0], "r")

mapdl.nsla("r", 1)

mapdl.nsel("r", "loc", "x", precorte, longitud + precorte + eps)

mapdl.components["cm_1"] = "node"

mapdl.allsel()

mapdl.asel("s", "loc", "z", 1.7)

areas = mapdl.geometry.anum

mapdl.geometry.area_select(areas[1], "r")

mapdl.nsla("r", 1)

mapdl.nsel("r", "loc", "x", precorte, longitud + precorte + eps)

mapdl.components["cm_2"] = "node"

# Configuración de opciones reales y de elementos

mapdl.allsel()

mapdl.components["_elemcm"] = "elem"

mapdl.mat(2)

mapdl.r(3, "", "", 1.0, 0.1, 0, "")

mapdl.rmore("", "", 1.0e20, 0.0, 1.0, "")

mapdl.rmore(0.0, 0.0, 1.0, "", 1.0, 0.5)

mapdl.rmore(0.0, 1.0, 1.0, 0.0, "", 1.0)

mapdl.rmore("", "", "", "", "", 1.0)
```

```
mapdl.keyopt(3, 4, 0)
mapdl.keyopt(3, 5, 0)
mapdl.keyopt(3, 7, 0)
mapdl.keyopt(3, 8, 0)
mapdl.keyopt(3, 9, 0)
mapdl.keyopt(3, 10, 0)
mapdl.keyopt(3, 11, 0)
mapdl.keyopt(3, 12, 3)
mapdl.keyopt(3, 14, 0)
mapdl.keyopt(3, 18, 0)
mapdl.keyopt(3, 2, 0)
mapdl.keyopt(2, 5, 0)
# Generar elementos TARGE170 sobre cm_1
mapdl.nsel("s", "", "", "cm_1")
mapdl.components["_target"] = "node"
mapdl.type(2)
mapdl.esln("s", 0)
mapdl.esurf()
# Generar elementos CONTA174 sobre cm_2
mapdl.cmsel("s", "_elemcm")
mapdl.nsel("s", "", "", "cm_2")
mapdl.components["_contact"] = "node"
mapdl.type(3)
```

```
mapdl.esln("s", 0)
mapdl.esurf()
```

Generar condiciones de contorno

Asignar condiciones de contorno para replicar las condiciones reales del ensayo. Un extremo de las dos placas compuestas está fijo contra la traslación en los ejes x, y y z. En el lado opuesto de la placa, se aplican condiciones de desplazamiento para simular la apertura de la grieta interfacial. Estas condiciones se aplican a los nodos superiores e inferiores correspondientes a los bordes geométricos ubicados respectivamente en estas coordenadas (x, y, z): `(0.0, y, 0.0)` y `(0.0, y, 3.4)`. Se asignan dos componentes diferentes a estos conjuntos de nodos para una identificación más rápida de los nodos que soportan las fuerzas de reacción.

```python
# Condiciones de contorno y resolución del modelo
mapdl.allsel()
mapdl.nsel(type_="s", item="loc", comp="x", vmin=0.0, vmax=0.0)
mapdl.nsel(type_="r", item="loc", comp="z", vmin=2 * altura, vmax=2 * altura)
mapdl.d(node="all", lab="uz", value=desplazamiento)
mapdl.components["top_nod"] = "node"
mapdl.allsel()
mapdl.nsel(type_="s", item="loc", comp="x", vmin=0.0, vmax=0.0)
mapdl.nsel(type_="r", item="loc", comp="z", vmin=0.0, vmax=0.0)
mapdl.d(node="all", lab="uz", value=-10)
mapdl.components["bot_nod"] = "node"
```

```
# Fijar el modelo y resolver
mapdl.allsel()
mapdl.nsel(
    type_="s",
    item="loc",
    comp="x",
    vmin=longitud + precorte,
    vmax=longitud + precorte,
)
mapdl.d(node="all", lab="ux", value=0.0)
mapdl.d(node="all", lab="uy", value=0.0)
mapdl.d(node="all", lab="uz", value=0.0)
mapdl.eplot(plot_bc=True, bc_glyph_size=3, title="")
```

 # Resolver el análisis estático no lineal

Ejecutar un análisis estático no lineal. Para facilitar una progresión suave de la apertura de la grieta y facilitar la convergencia del solucionador estático, se solicitan 100 subpasos.

```python
# Ingresar al procesador de soluciones y definir la configuración del análisis
mapdl.allsel()
mapdl.finish()
mapdl.run("/SOLU")
mapdl.antype("static")
```

```
# Activar la geometría no lineal
mapdl.nlgeom("on")
# Solicitar subpasos
mapdl.autots(key="on")
mapdl.nsubst(nsbstp=100, nsbmx=100, nsbmn=100)
mapdl.kbc(key=0)
mapdl.outres("all", "all")
# Resolver
output = mapdl.solve()
```

 # Postprocesamiento

Utilizar PyMAPDL y PyDPF para el postprocesamiento.

Postprocesar resultados usando PyMAPDL

Esta sección muestra cómo utilizar PyMAPDL para el postprocesamiento de resultados. Dado que es importante medir la longitud de la delaminación, se graficará el parámetro de daño cohesivo. Aunque el parámetro de daño es un parámetro de elemento, el resultado se proporciona en términos de un resultado nodal. Así, el resultado para solo uno de los elementos cohesivos de cuatro nodos `NMISC = 70` se presenta. El resultado para los otros nodos está presente en `NMISC = 71, 72, 73`. Puedes recuperar los valores nodales reales del parámetro de daño del modelo resuelto en forma de una tabla (o un arreglo).

```python
# Entrar al postprocesador
mapdl.post1()
```

```
# Seleccionar el subpaso
mapdl.set(1, 100)
# Seleccionar elementos ``CONTA174``
mapdl.allsel()
mapdl.esel("s", "ename", "", 174)
# Graficar los valores de los elementos
mapdl.post_processing.plot_element_values(
    "nmisc", 70, scalar_bar_args={"title": "Daño Cohesivo"}
)
# Extraer los valores nodales del parámetro de daño
mapdl.allsel()
mapdl.esel("s", "ename", "", 174)
mapdl.etable("damage", "nmisc", 70)
damage_df = mapdl.pretab("damage").to_dataframe()
```

 # Análisis de Fuerza vs Desplazamiento

Este script realiza un análisis postprocesamiento para evaluar la relación entre la fuerza y el desplazamiento en un modelo estructural. Inicialmente, se configura el entorno para manejar los resultados, descargando y asegurando que los datos del análisis están accesibles. Posteriormente, se extraen las fuerzas y desplazamientos nodales para una selección específica de la malla en cada subpaso de tiempo. Los resultados se acumulan y se visualizan en un gráfico de fuerza versus desplazamiento para facilitar la interpretación del comportamiento estructural del modelo bajo carga.

```python
# Configurar el directorio y acceso a los resultados
directorio_temporal = tempfile.gettempdir()
ruta_resultados = mapdl.download_result(directorio_temporal)
dpf.core.make_tmp_dir_server(dpf.SERVER)
# Determinar la ruta correcta de acceso a los resultados en función del servidor
if dpf.SERVER.local_server:
    fuente_path = ruta_resultados
else:
    fuente_path = dpf.upload_file_in_tmp_folder(ruta_resultados)

# Cargar el modelo y seleccionar la malla
modelo = dpf.Model(fuente_path)
seleccion_malla = modelo.metadata.named_selection("BOT_NOD")
mapdl.exit()
# Inicializar listas para almacenar los totales de fuerzas y desplazamientos
f_totales = []
d_totales = []
# Calcular las fuerzas y desplazamientos para cada subpaso
for i in range(100):
    evaluacion_fuerza = modelo.results.element_nodal_forces(time_scoping=i,
mesh_scoping=seleccion_malla).eval()
    fuerza = evaluacion_fuerza[0].data
```

```
        f_totales.append(np.sum(fuerza[:, 2]))
        d = abs(modelo.results.displacement(time_scoping=i,
mesh_scoping=seleccion_malla).eval()[0].data[0])
        d_totales.append(d[2])
    # Asegurar que los valores iniciales sean cero
    d_totales[0] = 0
    f_totales[0] = 0
    # Crear y mostrar el gráfico de fuerza vs desplazamiento
    fig, ax = plt.subplots()
    plt.plot(d_totales, f_totales, "b")
    plt.title('Fuerza vs Desplazamiento')
    plt.xlabel('Desplazamiento [mm]')
    plt.ylabel('Fuerza [N]')
    plt.show()
```

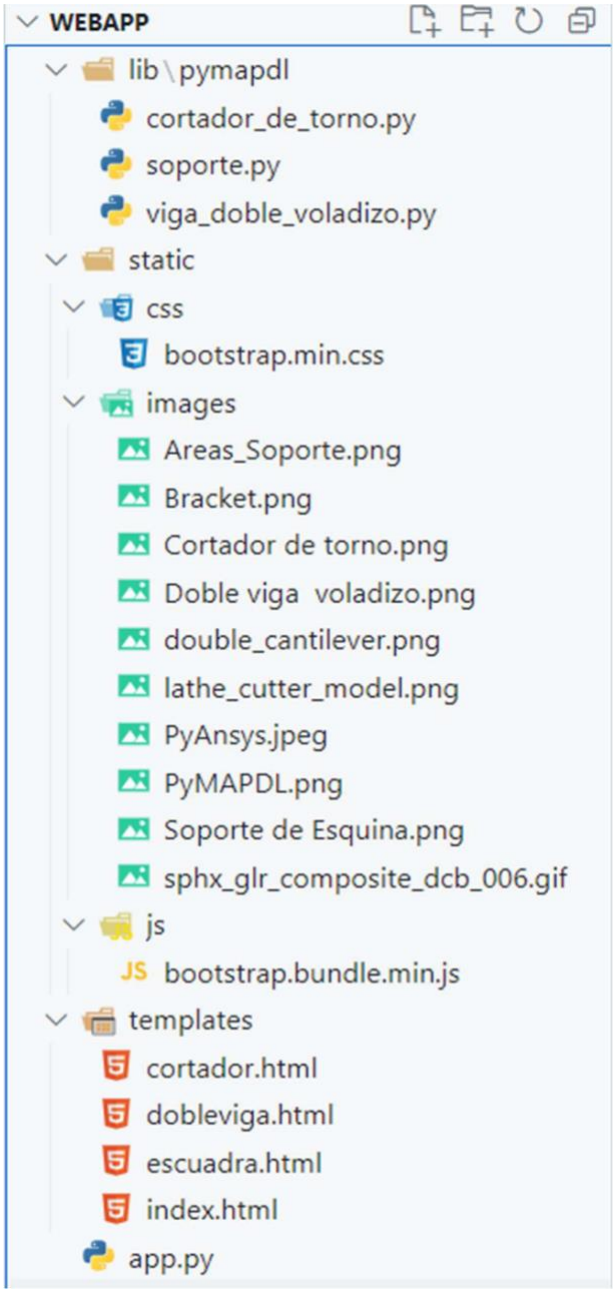
Salir de MAPDL

```python
mapdl.exit()
```

Apéndice F

Estructura y archivos fuente de la WebApp

https://github.com/IvanGMtz/SimuLab--PyMAPDL-Simulation-Analytics-Platform



ARCHIVO APP.PY

```
from flask import Flask, render_template, redirect, url_for, request, send_from_directory
import os
# Importaciones específicas del proyecto
```

```
from lib.pymapdl.soporte import *
from lib.pymapdl.viga_doble_voladizo import *
from lib.pymapdl.cortador_de_torno import *
app = Flask(__name__)
# Configuración del directorio de imágenes
IMAGE_DIR = os.path.join(os.getcwd(), 'static', 'images')
os.makedirs(IMAGE_DIR, exist_ok=True)
@app.route('/')
def home():
return render_template('index.html')
@app.route('/escuadra', methods=['GET', 'POST'])
def escuadra():
if request.method == 'POST':
# Extraer los valores enviados desde el formulario
box1 = request.form.get('box1', '').split(',')
box2 = request.form.get('box2', '').split(',')
radio_perno = float(request.form.get('pinhole_radius', 0))
tamaño_elemento = float(request.form.get('element_size', 0))
carga_presion = request.form.get('pressure_load', '').split(',')
# Convertir los valores de string a los tipos correctos
box1 = [float(value) for value in box1]
box2 = [float(value) for value in box2]
carga_presion = [float(value) for value in carga_presion]
```

```
        # Llamar a la función analizar_soporte
        ruta_imagen_deformacion = analizar_soporte(
        box1, box2, radio_perno, tamaño_elemento, carga_presion
        )
        # Rutas de las imágenes para usar en HTML
        url_imagen_deformacion = os.path.join('images',
os.path.basename(ruta_imagen_deformacion))
        return render_template('escuadra.html', solve_status="Solved",
        deformation_image_url=url_imagen_deformacion)
        return render_template('escuadra.html')
        @app.route('/dobleviga', methods=['GET', 'POST'])
        def doble_viga():
        if request.method == 'POST':
        # Extraer los valores enviados desde el formulario
        longitud = float(request.form.get('length', 0))
        precorte = float(request.form.get('pre_crack', 0))
        ancho = float(request.form.get('width', 0))
        altura = float(request.form.get('height', 0))
        d = float(request.form.get('d', 0))
        # Llamar a la función analizar_viga_doble_voladizo
        ruta_imagen_fuerza_desplazamiento = analizar_viga_doble_voladizo(longitud, precorte,
ancho, altura, d)
        # Rutas de las imágenes para usar en HTML
```

```
url_imagen_fuerza_desplazamiento = os.path.join('images',
os.path.basename(ruta_imagen_fuerza_desplazamiento))
return render_template('dobleviga.html', solve_status="Solved",
ForcevsDisplacement_image_url=url_imagen_fuerza_desplazamiento)
return render_template('dobleviga.html')
@app.route('/cortador_de_torno', methods=['GET', 'POST'])
def cortador_de_torno():
if request.method == 'POST':
# Extraer los valores enviados desde el formulario
EXX = float(request.form.get('EXX', 1.0e7))
NU = float(request.form.get('NU', 0.27))
Fuerza = float(request.form.get('Fuerza', 10000))
# Llamar a la función analizar_cortador_de_torno
resultados = analizar_cortador_de_torno(EXX, NU, Fuerza)
# Rutas de las imágenes para usar en HTML
load_image_url, stress_image_url, xy_stress_image_url = resultados
return render_template('cortador.html', solve_status="Solved",
load_image_url=load_image_url,
stress_image_url=stress_image_url,
xy_stress_image_url=xy_stress_image_url)
return render_template('cortador.html')
@app.route('/images/<filename>')
def uploaded_file(filename):
```

```
return send_from_directory(IMAGE_DIR, filename)

if __name__ == '__main__':

app.run(debug=True)
```

ARCHIVO INDEX.HTML

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link rel="stylesheet" href="../static/css/bootstrap.min.css" />

<script src="../static/js/bootstrap.bundle.min.js"></script>

<title>FEA Learning Platform</title>

</head>

<body>

<nav class="navbar navbar-expand-lg bg-dark" data-bs-theme="dark">

<div class="container-fluid">

<a class="navbar-brand" href="/">

<img src="../static/images/PyAnsys.jpeg" alt="" width="80" height="24" class="d-inline-
block aligntext-

top">

</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bstarget="#
```

```
navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false"
arialabel="
Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNavDropdown">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="/">Inicio</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Instalación</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Comandos MAPDL</a>
</li>
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown"
ariaexpanded="
false">
Ejemplos
</a>
<ul class="dropdown-menu">
```

```
<li><a class="dropdown-item" href="/escuadra">Escuadra de Esquina</a></li>
<li><a class="dropdown-item" href="/cortador_de_torno">Cortador de Torno</a></li>
<li><a class="dropdown-item" href="/dobleviga">Ensayo de Doble Viga en Voladizo</a></li>
</ul>
</li>
</ul>
</div>
</div>
</nav>
<div class="container shadow-lg p-3 mb-5 bg-body-tertiary rounded ">
<h1 class="text-center ">Bienvenidos a la Plataforma de Aprendizaje FEA</h1>
<p class="lead text-center">Explora el poder de Python y ANSYS para el análisis por elementos
finitos.</p>
<img src="../static/images/sphx_glr_composite_dcb_006.gif" class="img-fluid rounded mx-auto d-block"
height="200" width="400" alt="">
<section>
<h2>¿Qué es PyAnsys y PyMAPDL?</h2>
<p>PyAnsys es un conjunto de bibliotecas Python que proporciona una interfaz para interactuar con el
```

software de simulación ANSYS, permitiendo a los usuarios automatizar tareas, acceder a capacidades de postprocesamiento

y mucho más. PyMAPDL, específicamente, es una biblioteca que conecta Python con ANSYS

Mechanical APDL, ofreciendo un acceso programático completo a todas las funcionalidades de MAPDL.

</section>

<section>

<h2>Instalación</h2>

Comienza tu viaje con PyAnsys y PyMAPDL con nuestra guía de instalación paso a paso. Asegúrate

de cumplir con los requisitos previos y sigue las instrucciones detalladas para configurar tu entorno.

Aprende a instalar

</section>

<section>

<h2>Comenzando con PyMAPDL</h2>

Descubre cómo utilizar PyMAPDL para crear y analizar modelos de elementos finitos. Desde la

generación de mallas hasta la ejecución de simulaciones, aprende todo lo que necesitas saber para

comenzar.

Ver guía de comandos

```
</section>
<section>
<h2>Ejemplos prácticos</h2>
<p>Aplica tus conocimientos con ejemplos prácticos que muestran el uso de PyMAPDL en escenarios
reales. Cada ejemplo viene con código fuente completo y una explicación detallada de los procesos
involucrados.</p>
<a href="#" class="btn btn-warning ">Explorar ejemplos</a>
</section>
</div>
</body>
</html>
```

ARCHIVO ESCUADRA.HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Análisis Estático de un Soporte</title>
<link rel="stylesheet" href="../../static/css/bootstrap.min.css" />
<script src="../../static/js/bootstrap.bundle.min.js"></script>
<script language="javascript">
```

```
function fetchData() {
document.getElementsByName("SolveStatus")[0].value = "Solución en Progreso";
let b = document.getElementById("SolveStatus");
b.classList.remove("bg-light");
b.classList.add("bg-danger");
b.style.color = "white";
}
</script>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<div class="container-fluid">
<a class="navbar-brand" href="/">
<img
src="../static/images/PyAnsys.jpeg"
alt=""
width="80"
height="24"
class="d-inline-block align-text-top"
/>
</a>
<button
class="navbar-toggler"
```

```
type="button"
data-bs-toggle="collapse"
data-bs-target="#navbarNavDropdown"
aria-controls="navbarNavDropdown"
aria-expanded="false"
aria-label="Toggle navigation"
>
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNavDropdown">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="/">Inicio</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Instalación</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Comandos MAPDL</a>
</li>
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown"
ariaexpanded="
```

```
false">
Ejemplos
</a>
<ul class="dropdown-menu">
<li><a class="dropdown-item" href="/escuadra">Escuadra de Esquina</a></li>
<li><a class="dropdown-item" href="/cortador_de_torno">Cortador de Torno</a></li>
<li><a class="dropdown-item" href="/dobleviga">Ensayo de Doble Viga en Voladizo</a></li>
</ul>
</li>
</ul>
</div>
</div>
</nav>
<div class="container shadow-lg p-3 mb-5 bg-body-tertiary rounded">
<h1 class="mb-4 text-center">Análisis Estático de un Soporte</h1>
<div class="row">
<div class="col-md-12 fw-light">
<p class="text-justify">
Esta es una análisis estructural estático simple de un paso de carga de un soporte de esquina. El orificio
del perno superior izquierdo está restringido (soldado) alrededor de toda su circunferencia, y se aplica una
```

carga de presión decreciente en la parte inferior del orificio del perno inferior derecho. Se utiliza el sistema de

unidades estadounidense. El objetivo es demostrar cómo se utiliza típicamente Mechanical APDL en un

análisis.

<h3>Modelo del Soporte</h3>

Las dimensiones del soporte de esquina se muestran en la figura siguiente. El soporte está hecho de

acero A36 con un módulo de Young de 3*10^7 psi y un coeficiente de Poisson de 0.27.

<img src="../static/images/Soporte de Esquina.png" alt="Imagen del Modelo del Soporte" class="imgfluid"

style="width: 480px">

<img src="../static/images/Areas_Soporte.png" alt="Imagen del Areas del Soporte" class="img-fluid"

style="width: 480px">

<h3>Aproximación y suposiciones</h3>

Dado que el soporte es delgado en la dirección z (1/2 pulgada de espesor) en comparación con sus

dimensiones en x e y, y dado que la carga de presión actúa solo en el plano x-y, se asume un estado de tensión

plana para el análisis.

El enfoque es utilizar el modelado sólido para generar el modelo 2D y mallarlo automáticamente con

nodos y elementos. Una alternativa sería crear los nodos y elementos directamente.

<h2>ENTRADAS</h2>

<form onsubmit="fetchData()" method="POST">

<label for="box1" class="form-label">Coordenadas del Área 1 (x1, x2, y1, y2)</label>

<input type="text" class="form-control" id="box1" name="box1" required placeholder="0, 6, -1, 1"/>

```
<label for="box2" class="form-label">Coordenadas del Área 2 (x1, x2, y1, y2)</label>
<input type="text" class="form-control" id="box2" name="box2" required placeholder="4, 6, -1, -
3"/>
</div>
<div class="mb-3">
<label for="pinhole_radius" class="form-label">Radio del Orificio (in)</label>
<input type="number" class="form-control" step="0.01" id="pinhole_radius" name="pinhole_radius"
required/>
</div>
<div class="mb-3">
<label for="element_size" class="form-label">Tamaño del Elemento (in)</label>
<input type="number" class="form-control" step="0.01" id="element_size" name="element_size"
required/>
</div>
<div class="mb-3">
<label for="pressure_load" class="form-label">Carga de Presión (psi) en los Orificios</label>
<input type="text" class="form-control" id="pressure_load" name="pressure_load" required
placeholder="50, 500"/>
```

```
        </div>
        <button type="submit" class="btn btn-primary">Resolver</button>
        </form>
        </div>
        <div class="my-4 col">
        <h2>SALIDAS</h2>
        <div class="mb-3">
        <label for="SolveStatus" class="form-label">Estado de la Solución</label>
        {% if deformation_image_url %}
        <input type="text" class="form-control bg-success" id="SolveStatus"
name="SolveStatus"
        readonly="readonly" value="Resuelto"/>
        {% else %}
        <input type="text" class="form-control bg-light" id="SolveStatus" name="SolveStatus"
        readonly="readonly" value=""/>
        {% endif %}
        </div>
        <div class="Result-Image">
        {% if deformation_image_url %}
        <h2>Resultado de Deformación</h2>
        <img src="{{ deformation_image_url }}" alt="Imagen de Deformación" style="width:
480px"/>
        {% endif %}
```

```
</div>
</div>
</div>
</div>
</body>
</html>
```

ARCHIVO CORTADOR.HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Análisis Estructural del Cortador de Torno</title>
<link rel="stylesheet" href="../../static/css/bootstrap.min.css">
<script src="../../static/js/bootstrap.bundle.min.js"></script>
<script language="javascript">
function fetchData() {
document.getElementsByName("SolveStatus")[0].value = "Solución en Progreso";
let b = document.getElementById("SolveStatus");
b.classList.remove("bg-light");
b.classList.add("bg-danger");
b.style.color = "white";
}
```

```
</script>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<div class="container-fluid">
<a class="navbar-brand" href="/">
<img src="../static/images/PyAnsys.jpeg" alt="" width="80" height="24" class="d-inline-block
align-text-top">
</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bstarget="#
navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" arialabel="
Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNavDropdown">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="/">Inicio</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Instalación</a>
```

```
        </li>
        <li class="nav-item">
        <a class="nav-link" href="#">Comandos MAPDL</a>
        </li>
        <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown"
ariaexpanded="
        false">
        Ejemplos
        </a>
        <ul class="dropdown-menu">
        <li><a class="dropdown-item" href="/escuadra">Escuadra de Esquina</a></li>
        <li><a class="dropdown-item" href="/cortador_de_torno">Cortador de Torno</a></li>
        <li><a class="dropdown-item" href="/dobleviga">Ensayo de Doble Viga en
        Voladizo</a></li>
        </ul>
        </li>
        </ul>
        </div>
        </div>
        </nav>
        <div class="container shadow-lg p-3 mb-5 bg-body-tertiary rounded">
        <div class="row fw-light">
```

<h1 class="mb-4 text-center">Análisis Estructural del Cortador de Torno</h1>

Este ejemplo tiene como objetivo destacar algunas de las características habitualmente utilizadas de

PyMAPDL a través de un modelo de elemento finito de un cortador de torno. Los cortadores de torno tienen

múltiples vías de desgaste y falla, y los análisis que respaldan su diseño suelen ser térmico-estructurales

transitorios. Sin embargo, para simplificar, este ejemplo de simulación utiliza una carga no uniforme.

<h4 class="mt-4">Geometría y descripción de la carga del cortador de torno</h4>

<img src="../static/images/Cortador de torno.png" alt="Geometría y Carga del Cortador de Torno"

class="img-fluid" style="width: 550px;">

<h3 class="mt-4">Contenidos</h3>

Variables y lanzamiento: Definir las variables necesarias y lanzar MAPDL.

Geometría, malla y parámetros de MAPDL: Importar geometría e inspeccionar parámetros de MAPDL. Definir el modelo de material elástico lineal con variables de Python.

```
Malla y aplicar condiciones de contorno de simetría.</li>
<li><strong>Sistema de coordenadas y carga:</strong> Crear un sistema de coordenadas local
para la carga aplicada y verificar con un gráfico.</li>
<li><strong>Carga de presión:</strong> Definir la carga de presión como una función seno de la
longitud del área de aplicación usando arreglos de numpy. Importar el arreglo de presión a MAPDL como un
arreglo de tabla. Verificar la carga aplicada y resolver.</li>
<li><strong>Visualización de resultados:</strong> Mostrar visualización de resultados,
visualización con selección y manejo de la leyenda del gráfico.</li>
</ul>
</div>
</div>
<div class="row">
<form onsubmit="fetchData()" method="POST">
<div class="row">
<div class="col-md-3">
<h2>ENTRADAS</h2>
<div class="mb-3">
<label for="EXX" class="form-label">Módulo de Young (psi)</label>
<input type="number" class="form-control" id="EXX" name="EXX" required>
</div>
```

```
<div class="mb-3">
<label for="NU" class="form-label">Coeficiente de Poisson</label>
<input type="number" class="form-control" step="0.01" id="NU" name="NU" required>
</div>
<div class="mb-3">
<label for="Fuerza" class="form-label">Fuerza aplicada (lbf)</label>
<input type="number" class="form-control" id="Fuerza" name="Fuerza" required>
</div>
<button type="submit" class="btn btn-primary mt-4">Calcular</button>
</div>
<div class="col-md">
<h2>SALIDAS</h2>
<div class="mb-3">
<label for="SolveStatus" class="form-label">Estado de la Solución</label>
{% if load_image_url %}
<input type="text" class="form-control bg-success" id="SolveStatus"
name="SolveStatus"
readonly="readonly" value="Resuelto"/>
{% else %}
<input type="text" class="form-control bg-light" id="SolveStatus" name="SolveStatus"
readonly="readonly" value=""/>
{% endif %}
</div>
```

```
<div id="result_images" class="mb-4 text-center">
{% if load_image_url %}
```

CORTADOR_DE_TORNO.PY

```
import os
import numpy as np
from ansys.mapdl.core import launch_mapdl
from ansys.mapdl.core.examples.downloads import download_example_data
def analizar_cortador_de_torno(EXX, NU, Fuerza):
    # Directorio de trabajo actual
    path = os.getcwd()
    PI = np.pi
    mapdl = launch_mapdl(additional_switches="-smp")
    # Reiniciar la base de datos de MAPDL y configurar el entorno
    mapdl.clear()

    # Importar la geometría y definir propiedades del material y malla
    archivo_geometria = download_example_data("LatheCutter.anf", "geometry")
    mapdl.input(archivo_geometria)
    mapdl.finish()
    longitud_presion = mapdl.parameters["PRESS_LENGTH"]
    mapdl.units("Bin")
    mapdl.title("Cortador de Torno")
    mapdl.prep7()
```

```
mapdl.mp("EX", 1, EXX)

mapdl.mp("NUXY", 1, NU)

mapdl.et(1, 285)

mapdl.smrtsize(4)

mapdl.aesize(14, 0.0025)

mapdl.vmesh(1)

mapdl.da(11, "symm")

mapdl.da(16, "symm")

mapdl.da(9, "symm")

mapdl.da(10, "symm")


# Crear un sistema de coordenadas local y configurar la carga

mapdl.cskp(11, 0, 2, 1, 13)

mapdl.csys(1)

mapdl.view(1, -1, 1, 1)

mapdl.psymb("CS", 1)

# Definir y aplicar la carga de presión

puntos = 10

longitud_x = np.linspace(0, longitud_presion, puntos)

presion = Fuerza * np.sin(PI * longitud_x / longitud_presion)

presion = np.stack((longitud_x, presion), axis=-1)

mapdl.load_table("MY_PRESS", presion, "X", csysid=11)

mapdl.asel("S", "Area", "", 14)
```

```
mapdl.nsla("S", 1)
mapdl.sf("All", "Press", "%MY_PRESS%")
mapdl.allsel()
# Configurar y resolver el modelo
mapdl.finish()
mapdl.slashsolu()
mapdl.nlgeom("On")
mapdl.psf("PRES", "NORM", 3, 0, 1)
mapdl.view(1, -1, 1, 1)
ruta_imagen_carga = "static/images/Pressure_Load_plot.png"
mapdl.eplot(vtk=False, savefig=ruta_imagen_carga)
mapdl.solve()
mapdl.finish()
# Procesamiento y visualización de resultados
mapdl.post1()
mapdl.set("last")
mapdl.allsel()
argumentos_barra = dict(
    title_font_size=20,
    label_font_size=16,
    shadow=True,
    n_labels=9,
    italic=True,
```

```
    bold=True,
    fmt="%.1f",
    font_family="arial",
    title="Esfuerzo Principal 1 (psi)",
    color="black",
)
ruta_imagen_esfuerzo = "static/images/principal_stress_plot.png"
mapdl.post_processing.plot_nodal_principal_stress("1", smooth_shading=False,
    edge_color="black",
    background="beige",
    show_edges=True,
    scalar_bar_args=argumentos_barra,
    n_colors=256,
    cmap="jet",
    savefig=ruta_imagen_esfuerzo
    )

ruta_imagen_esfuerzo_xy = "static/images/XYprincipal_stress_plot.png"
mapdl.post_processing.plot_nodal_principal_stress(
    "1",
    cpos="xy",
    edge_color="black",
    background="beige",
```

```
        show_edges=True,
        scalar_bar_args=argumentos_barra,
        n_colors=256,
        cmap="jet",
        savefig=ruta_imagen_esfuerzo_xy
    )
    mapdl.exit()
    return ruta_imagen_carga, ruta_imagen_esfuerzo, ruta_imagen_esfuerzo_xy
# Ejemplo de uso de la función
#print(analizar_cortador_de_torno(1.0e7, 0.27, 10000))
```

SOPORTE.PY

```
from ansys.mapdl.core import launch_mapdl
def analizar_soporte(box1, box2, radio_perno, tamaño_elemento, carga_presion):
    nombre_trabajo = "analisis_soporte"
    mapdl = launch_mapdl(jobname=nombre_trabajo)
    mapdl.clear()
    mapdl.prep7()
    # Creación de la geometría
    mapdl.rectng(*box1)
    mapdl.rectng(*box2)

    # Crear y restar agujeros para los pernos
    radio = 1
```

```
centro1_X = box1[0]

centro1_Y = (box1[2] + box1[3]) / 2

mapdl.cyl4(centro1_X, centro1_Y, radio)

centro2_X = (box2[0] + box2[1]) / 2

centro2_Y = box2[3]

mapdl.cyl4(centro2_X, centro2_Y, radio)

mapdl.aadd("all")

# Crear el primer agujero de perno

perno1_X = box1[0]

perno1_Y = (box1[2] + box1[3]) / 2

perno1 = mapdl.cyl4(perno1_X, perno1_Y, radio_perno)

perno2_X = (box2[0] + box2[1]) / 2

perno2_Y = box2[3]

perno2 = mapdl.cyl4(perno2_X, perno2_Y, radio_perno)

# Eliminar áreas de agujeros de los pernos del soporte

mapdl.asba("all", perno1)

soporte = mapdl.asba("all", perno2)

# Definir propiedades del material

modulo_young = 30e6  # Módulo de Young

coef_poisson = 0.27  # Coeficiente de Poisson

mapdl.mp("EX", 1, modulo_young)

mapdl.mp("PRXY", 1, coef_poisson)

# Establecer el grosor del elemento
```

```
grosor = 0.5
mapdl.r(1, grosor)  # grosor de 0.5 unidades de longitud
# Tipo de elemento y malla
mapdl.et(1, "PLANE183", kop3=3)
mapdl.esize(tamaño_elemento)
mapdl.amesh(soporte)
mapdl.allsel()
mapdl.solution()
mapdl.antype("STATIC")
bc1 = mapdl.lsel(
    "S", "LOC", "X", perno1_X - radio_perno, perno1_X + radio_perno
)
fixNodes = mapdl.nsll(type_="S")
# Condiciones de frontera
mapdl.d("ALL", "ALL", 0)  # Se asume cero por defecto
# Aplicar carga de presión
mapdl.lsel("S", "LOC", "Y", perno2_Y - radio_perno, perno2_Y)
mapdl.lsel("R", "LOC", "X", 0, perno2_X)
mapdl.sf("ALL", "PRES", carga_presion[0], carga_presion[1])
mapdl.allsel()


mapdl.lsel("S", "LOC", "Y", perno2_Y - radio_perno, perno2_Y)
mapdl.lsel("R", "LOC", "X", perno2_X, perno2_X + radio_perno)
```

```
mapdl.sf("ALL", "PRES", carga_presion[1], carga_presion[0])
mapdl.allsel()
mapdl.solve()
mapdl.post1()
result = mapdl.result
# Guardar la imagen de desplazamiento
ruta_imagen_deformacion = "static/images/deformation_plot.png"
argumentos_barra = dict(
    title_font_size=20,
    label_font_size=16,
    shadow=True,
    n_labels=9,
    italic=True,
    bold=True,
    font_family="arial",
    title="Desplazamiento Normalizado (in)",
    color="black",
)
mapdl.post_processing.plot_nodal_displacement(
    'NORM',
    cpos="xy",
    edge_color="black",
    background="beige",
```

```
        show_edges=True,
        scalar_bar_args=argumentos_barra,
        n_colors=256,
        cmap="jet",
        savefig=ruta_imagen_deformacion
    )
    result.plot_principal_nodal_stress(
    0,
    "SEQV",
    cpos="xy",
    background="w",
    text_color="k",
    add_text=True,
    show_edges=True,
    )
    mapdl.exit()
    return ruta_imagen_deformacion
```

VIGA_DOBLE_VOLADIZO.PY

```
import os
import tempfile
from ansys.mapdl.core import launch_mapdl
```

```
from ansys.dpf import core as dpf
import numpy as np
from ansys.mapdl import core as pymapdl
import matplotlib.pyplot as plt
def analizar_viga_doble_voladizo(longitud, precorte, ancho, altura, desplazamiento):
    mapdl = pymapdl.launch_mapdl()
    # Una pequeña cantidad definida para evitar errores de redondeo al seleccionar entidades
geométricas
    eps = 1e-1
    # Configurar el modelo
    mapdl.prep7()
    mapdl.units("mpa")
    # Definir elementos y tamaños para simulación completamente tridimensional
    mapdl.et(1, 185)
    mapdl.et(2, 170)
    mapdl.et(3, 174)
    mapdl.esize(10.0)
    # Propiedades del material para las placas compuestas
    mapdl.mp("ex", 1, 61340)
    mapdl.mp("dens", 1, 1.42e-09)
    mapdl.mp("nuxy", 1, 0.1)
    # Ley cohesiva bilineal para elementos de contacto
    mapdl.mp("mu", 2, 0)
```

```
mapdl.tb("czm", 2, 1, "", "bili")
mapdl.tbtemp(25.0)
mapdl.tbdata(1, 50.0, 0.5, 50, 0.5, 0.01, 2)
# Creación y malla de la geometría
vnum0 = mapdl.block(0.0, longitud + precorte, 0.0, ancho, 0.0, altura)
vnum1 = mapdl.block(0.0, longitud + precorte, 0.0, ancho, altura, 2 * altura)
mapdl.mat(1)
mapdl.type(1)
mapdl.vmesh(vnum0)
mapdl.vmesh(vnum1)
mapdl.eplot()
# Generación de elementos cohesivos entre superficies de contacto
mapdl.allsel()
mapdl.asel("s", "loc", "z", 1.7)
areas = mapdl.geometry.anum
mapdl.geometry.area_select(areas[0], "r")
mapdl.nsla("r", 1)
mapdl.nsel("r", "loc", "x", precorte, longitud + precorte + eps)
mapdl.components["cm_1"] = "node"
mapdl.allsel()
mapdl.asel("s", "loc", "z", 1.7)
areas = mapdl.geometry.anum
mapdl.geometry.area_select(areas[1], "r")
```

```
mapdl.nsla("r", 1)

mapdl.nsel("r", "loc", "x", precorte, longitud + precorte + eps)

mapdl.components["cm_2"] = "node"

# Configuración de opciones reales y de elementos

mapdl.allsel()

mapdl.components["_elemcm"] = "elem"

mapdl.mat(2)

mapdl.r(3, "", "", 1.0, 0.1, 0, "")

mapdl.rmore("", "", 1.0e20, 0.0, 1.0, "")

mapdl.rmore(0.0, 0.0, 1.0, "", 1.0, 0.5)

mapdl.rmore(0.0, 1.0, 1.0, 0.0, "", 1.0)

mapdl.rmore("", "", "", "", "", 1.0)

mapdl.keyopt(3, 4, 0)

mapdl.keyopt(3, 5, 0)

mapdl.keyopt(3, 7, 0)

mapdl.keyopt(3, 8, 0)

mapdl.keyopt(3, 9, 0)

mapdl.keyopt(3, 10, 0)

mapdl.keyopt(3, 11, 0)

mapdl.keyopt(3, 12, 3)

mapdl.keyopt(3, 14, 0)

mapdl.keyopt(3, 18, 0)

mapdl.keyopt(3, 2, 0)
```

```
mapdl.keyopt(2, 5, 0)
# Generate TARGE170 elements on top of cm_1
mapdl.nsel("s", "", "", "cm_1")
mapdl.components["_target"] = "node"
mapdl.type(2)
mapdl.esln("s", 0)
mapdl.esurf()
# Generate CONTA174 elements on top of cm_2
mapdl.cmsel("s", "_elemcm")
mapdl.nsel("s", "", "", "cm_2")
mapdl.components["_contact"] = "node"
mapdl.type(3)
mapdl.esln("s", 0)
mapdl.esurf()
# Condiciones de contorno y resolución del modelo
mapdl.allsel()
mapdl.nsel(type_="s", item="loc", comp="x", vmin=0.0, vmax=0.0)
mapdl.nsel(type_="r", item="loc", comp="z", vmin=2 * altura, vmax=2 * altura)
mapdl.d(node="all", lab="uz", value=desplazamiento)
mapdl.components["top_nod"] = "node"
mapdl.allsel()
mapdl.nsel(type_="s", item="loc", comp="x", vmin=0.0, vmax=0.0)
mapdl.nsel(type_="r", item="loc", comp="z", vmin=0.0, vmax=0.0)
```

```
mapdl.d(node="all", lab="uz", value=-10)
mapdl.components["bot_nod"] = "node"
# Fijar el modelo y resolver
mapdl.allsel()
mapdl.nsel(
    type_="s",
    item="loc",
    comp="x",
    vmin=longitud + precorte,
    vmax=longitud + precorte,
)
mapdl.d(node="all", lab="ux", value=0.0)
mapdl.d(node="all", lab="uy", value=0.0)
mapdl.d(node="all", lab="uz", value=0.0)
mapdl.eplot(plot_bc=True, bc_glyph_size=3, title="")
mapdl.run("/SOLU")
mapdl.antype("static")
mapdl.nlgeom("on")
mapdl.autots(key="on")
mapdl.nsubst(nsbstp=100, nsbmx=100, nsbmn=100)
mapdl.kbc(key=0)
mapdl.outres("all", "all")
mapdl.solve()
```

```
        # Postprocesamiento
        mapdl.post1()
        mapdl.set(1, 100)
        mapdl.allsel()
        mapdl.esel("s", "ename", "", 174)
        mapdl.post_processing.plot_element_values("nmisc",  70,  scalar_bar_args={"title":
"Daño Cohesivo"})
        mapdl.allsel()
        mapdl.esel("s", "ename", "", 174)
        mapdl.etable("damage", "nmisc", 70)
        damage_df = mapdl.pretab("damage").to_dataframe()
        directorio_temporal = tempfile.gettempdir()
        ruta_resultados = mapdl.download_result(directorio_temporal)
        dpf.core.make_tmp_dir_server(dpf.SERVER)
        if dpf.SERVER.local_server:
            fuente_path = ruta_resultados
        else:
            fuente_path = dpf.upload_file_in_tmp_folder(ruta_resultados)
        modelo = dpf.Model(fuente_path)
        seleccion_malla = modelo.metadata.named_selection("BOT_NOD")
        mapdl.exit()
        f_totales = []
        d_totales = []
```

```
        for i in range(100):
            evaluacion_fuerza = modelo.results.element_nodal_forces(time_scoping=i,
mesh_scoping=seleccion_malla).eval()
            fuerza = evaluacion_fuerza[0].data
            f_totales.append(np.sum(fuerza[:, 2]))
            d = abs(modelo.results.displacement(time_scoping=i,
mesh_scoping=seleccion_malla).eval()[0].data[0])
            d_totales.append(d[2])
        d_totales[0] = 0
        f_totales[0] = 0

        fig, ax = plt.subplots()
        plt.plot(d_totales, f_totales, "b")
        plt.title('Fuerza vs Desplazamiento')
        plt.xlabel('Desplazamiento [mm]')
        plt.ylabel('Fuerza [N]')
        ruta_imagen_fuerza = os.path.join('static/images', 'ForcevsDisplacement.png')
        plt.savefig(ruta_imagen_fuerza)
        plt.close(fig)
        return ruta_imagen_fuerza
    # Uso de la función para análisis de una viga de doble voladizo
    # print(analizar_viga_doble_voladizo(75.0, 10.0, 25.0, 1.7, 10.0))
```