ULPGC
Universidad de
Las Palmas de
Gran Canaria

TESIS DOCTORAL

Metodología para la optimización de parámetros geométricos en el proceso de fabricación aditiva basada en extrusión

Doctorado en Ingenierías Química, Mecánica y de Fabricación

Autora:

Gisela del Carmen Vega Rodríguez

Director 1:

Dr. Mario D. Monzón Verona

Director 2:

Dr. Rubén Paz Hernández

Las Palmas de Gran Canaria. Julio 2023

Agradecimientos

Este trabajo ha sido financiado a través del contrato predoctoral de la convocatoria de la Universidad de Las Palmas de Gran Canaria de 2019, con el código PIFULPGC-2019-ING-ARQ-1.

Además, ha recibido el apoyo para la movilidad y el desarrollo de la investigación por parte del proyecto BAMOS (Biomaterials and Additive Manufacturing: Osteochondral Scaffold innovation applied to osteoarthritis, H2020-MSCA-RISE-2016-734156) del programa de investigación e innovación Horizonte 2020 de la Unión Europea, el proyecto BioAM (Mejora de la biofuncionalidad de scaffolds poliméricos obtenidos por fabricación aditiva, DPI2017-88465-R) financiado por el Ministerio de Ciencia, Innovación y Universidades, y el proyecto PIZAM (PID2020-117648RB-I00) financiado por MCIN/AEI/10.13039/501100011033.

Deseo expresar mi agradecimiento a mis directores de tesis, Dr. Mario Monzón y Dr. Rubén Paz. Su valiosa orientación y conocimiento han sido fundamentales para el desarrollo de esta investigación. Agradezco también su paciencia y valiosas sugerencias que han enriquecido este trabajo.

My sincere gratitude to all the researchers who have contributed to the development of this research. Special thanks to Dr Andy Gleadall for his invaluable support and contributions to this project. Your resources and expertise have greatly enhanced the progress and outcomes of this study. I am truly grateful for your dedication and collaboration.

También quiero agradecer al Grupo de Investigación en Fabricación Integrada y Avanzada por brindarme los recursos necesarios para llevar a cabo esta investigación. Y, en especial, a mis compañeros, no sólo por su colaboración e intercambio de ideas, sino también por su apoyo y el ambiente lleno de calidez. A pesar de los obstáculos, siempre han sido el motivo y la inspiración para levantar mi ánimo y seguir adelante.

Por último, deseo agradecer a mi familia y amigos por su apoyo incondicional. A mis padres, por haber mostrado siempre su confianza en mí y haberme impulsado a alcanzar mis metas. A mi marido, por su paciencia, comprensión y necesaria presencia ante las dificultades. Me siento increíblemente afortunada.

Resumen

Las tecnologías de fabricación aditiva son cada vez más relevantes en los sectores de ingeniería y fabricación. Su uso se ha extendido en el desarrollo de nuevos productos, abarcando desde prototipos hasta piezas funcionales y herramientas. Las ventajas de la fabricación aditiva están asociadas con su capacidad para fabricar piezas complejas desde el punto de vista geométrico con un menor esfuerzo, en comparación con los métodos tradicionales. Además, estas piezas son altamente personalizables y rentables para producciones de bajo volumen.

Una de las tecnologías de fabricación aditiva que ha ganado relevancia en los últimos años es la fabricación por extrusión de material. En este proceso, se utiliza un material termoplástico en forma de filamento que se alimenta a través de una boquilla calentada. El material se funde y deposita capa por capa sobre una plataforma de construcción, siguiendo las instrucciones de un software de laminado específico, que se almacenan en un archivo de código G. Este software divide el modelo 3D en secciones transversales y genera los movimientos necesarios para que la boquilla deposite el material de manera precisa y controlada. De esta manera, la fabricación por extrusión de material permite la fabricación de piezas con geometrías complejas y una amplia gama de materiales termoplásticos.

Aunque la fabricación por extrusión de material es altamente versátil, presenta ciertas limitaciones en cuanto a la capacidad de predecir el comportamiento mecánico de las piezas fabricadas. La diferencia entre la pieza impresa en 3D y su modelo original radica en la mesoestructura, resistencia y propiedades del material, ya que la pieza impresa tiene una estructura interna formada por filamentos depositados. En la actualidad, no existen métodos ampliamente establecidos para modelar este tipo de piezas, lo que lleva a recurrir a modelos simplificados geométricamente y al análisis mediante el análisis de elementos finitos. Sin embargo, esta simplificación conduce a resultados que no se asemejan adecuadamente al comportamiento real de las piezas impresas. Por lo tanto, en esta tesis se ha propuesto desarrollar metodologías que permitan simular y analizar de manera más precisa las piezas fabricadas mediante esta tecnología, modelándolas a medida que se construyen capa por capa. Además, los modelos y resultados obtenidos se han utilizado para optimizar los parámetros geométricos de impresión (porcentaje de relleno, altura de capa, patrón de relleno, etc.), para lograr las propiedades deseadas en términos de comportamiento mecánico,

peso, tiempo de fabricación, etc.

Para conseguir el fin descrito, se ha creado una nueva metodología para el modelado de piezas impresas en 3D, basada en la geometría (operaciones de barrido que emulan la deposición de filamentos durante la impresión 3D, todo ello a partir del código G). Además, se ha comparado con otras técnicas de modelado desarrolladas por otros investigadores, con el fin de establecer la mejor metodología de modelado para cada aplicación y definir los pasos para obtener los resultados derivados del análisis de elementos finitos. Por otro lado, se ha diseñado una metodología de optimización, basada en el diseño de experimentos factorial y la aplicación del algoritmos genéticos y metamodelos Kriging.

Por último, se han validado las metodologías aplicándolas a dos casos prácticos. Por un lado, una geometría simple, una probeta de tamaño reducido sometida a flexión. Y, por otro, una geometría compleja, una estructura porosa con aplicación en la ingeniería tisular, para la regeneración de tejido óseo trabecular.

Índice

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

En este capítulo se describen los objetivos principales de la tesis titulada "Metodología para la optimización de parámetros geométricos en el proceso de fabricación aditiva basada en extrusión", y se detalla la estructura del presente documento.

1.1. OBJETIVOS

En la presente tesis se pretende optimizar los parámetros geométricos de fabricación (altura de capa, patrón y densidad de relleno, etc.) que se establecen en la fase de obtención del archivo de código G, utilizado para la posterior impresión en 3D de la pieza mediante extrusión de material. Esto se consigue con la retroalimentación de los resultados de simulación de un modelo 3D. El objetivo de la optimización es conseguir mejorar las propiedades mecánicas para garantizar la funcionalidad de las piezas impresas.

Para conseguir el objetivo principal, ha sido necesario el desarrollo de dos metodologías:

- Metodología de modelado

Según la hipótesis inicial, se requiere una metodología para obtener un modelo 3D de la pieza impresa, antes de ser fabricada mediante extrusión de material. Deberá ser una representación de la disposición de los filamentos según los parámetros geométricos de fabricación utilizados en la generación del código G, archivo inicial para impresión 3D. Asimismo, tomará como referencia este archivo para la obtención de coordenadas y realizar el modelado.

Además, dicho modelo se someterá a simulaciones para obtener sus características mecánicas, en concreto, a un análisis por elementos finitos (AEF).

- Metodología de optimización

Esta metodología tomará datos dimensionales como variables de diseño, y principalmente resultados mecánicos derivados del AEF como variables de respuesta. Con esta información, se aplicará una optimización por algoritmos genéticos, de acuerdo al diseño de experimentos, y proporcionará el resultado de los parámetros geométricos de fabricación óptimos (en base a las estimaciones del algoritmo) para obtener una pieza impresa con las características requeridas.

Ambas metodologías se aplicarán a distintas piezas para verificar el cumplimiento del objetivo principal. En este sentido, cabe destacar que, a pesar de que se ha planteado una metodología genérica para todo tipo de piezas de impresión 3D (con aplicaciones muy variadas), durante el desarrollo del presente trabajo se ha hecho especial hincapié en las aplicaciones biomédicas, concretamente en el desarrollo de estructuras porosas, denominadas scaffolds, para regeneración de tejidos, donde la impresión 3D cobra especial interés. Los scaffolds son estructuras tridimensionales utilizadas en la ingeniería tisular para favorecer el crecimiento celular y regeneración de tejidos biológicos como huesos, cartílagos, piel, vasos sanguíneos u órganos, imitando las propiedades de dicho tejido. Estos elementos son provisionales y están fabricados con materiales biocompatibles, capaces de degradarse gradualmente mientras se desarrolla el tejido natural, que finalmente ocupará su lugar.

1.2. ESTRUCTURA DEL DOCUMENTO

Este documento se divide en ocho capítulos. Como introducción a esta investigación, el Capítulo 1 recoge los objetivos principales de la tesis y el Capítulo 2, el estudio del estado del arte en el campo de la fabricación aditiva, el modelado, la predicción y la optimización de parámetros geométricos y, de forma específica, se recoge la bibliografía disponible relativa a la ingeniería tisular y estructuras porosas para regeneración celular.

En los Capítulos 3, 4 y 5 se recoge el trabajo realizado en el ámbito del modelado de piezas 3D. Concretamente, en el Capítulo 3, se describe la metodología de modelado desarrollada en esta investigación; en el Capítulo 4, se analizan otras alternativas disponibles para el modelado; y, por último, en el Capítulo 5, se compara la metodología de modelado desarrollada con la alternativa elegida.

Por otro lado, la explicación de la metodología de optimización utilizada en este estudio se desarrolla en el Capítulo 6. Mientras que, en el Capítulo 7, se aplican las metodologías de modelado y optimización a dos casos prácticos, partiendo de las conclusiones obtenidas en capítulos anteriores.

Por último, este documento se cierra con el Capítulo 8, en el que se presentan las conclusiones, difusión de resultados y líneas de investigación futuras.

Capítulo 2. Estudio del estado del arte

En este capítulo se expone la información disponible sobre la predicción de la geometría de piezas impresas, las opciones de modelado, simulación y comparación de resultados, y optimización de parámetros geométricos para la fabricación. Además, se analiza la información particular para la predicción de scaffolds impresos.

2.1. FABRICACIÓN ADITIVA

La fabricación aditiva (FA) ha conseguido captar el interés de los sectores de ingeniería y fabricación y cada vez se utiliza más en el desarrollo de nuevos productos, desde prototipos hasta piezas funcionales y herramientas. Las ventajas de la FA se pueden relacionar con la capacidad de fabricar y desarrollar una pieza geométricamente compleja con menos esfuerzo, lo cual resulta tedioso con métodos tradicionales. Además de esto, estas piezas pueden personalizarse para producciones de bajo volumen con factibilidad económica y produciendo menos residuo [1,2]. Debido a sus ventajas y a la gran variedad de materiales que pueden procesarse, desde metales, polímeros, cerámicas y materiales compuesto hasta materiales híbridos, la FA es utilizada en numerosos campos como el automovilístico, aeroespacial, médico y sanitario, electrónico, alimentario o de la construcción, entre otros [3].

Las técnicas de fabricación aditiva pueden agruparse en siete categorías diferentes según la norma ISO/ASTM 52900:2021: proyección de aglutinante, deposición de energía focalizada, extrusión de material, proyección de material, fusión de lecho de polvo, laminado de hojas y fotopolimerización en cuba [2,3]. Entre ellas, la tecnología más utilizada para la fabricación de scaffolds es la extrusión de material (MEX), comúnmente conocida como impresión 3D. El proceso de MEX presenta facilidad y flexibilidad en el procesamiento y en la selección de materiales, así como en el proceso de fabricación [2,4–11].

El proceso genérico, para la fabricación de una pieza por MEX, comienza con la generación de un modelo 3D CAD (Diseño Asistido por Computadora). Este modelo se convertirá en un archivo STL (*Standard Triangle Language*) o similar (*Additive Manufacturing File, AMF*), que transforma la geometría CAD en un formato de malla triangulada. A continuación, se utilizan los programas informáticos de laminado, que

dividen el modelo en capas horizontales. Estos softwares también determinan la trayectoria para que el extrusor de la impresora genere el perímetro y el relleno del modelo, asignando comandos de control numérico (CNC). Estos comandos CNC se almacenan en un formato de archivo G-code. Este archivo permite que la máquina de impresión 3D imprima la pieza final [11,12].

La pieza es fabricada de abajo hacia arriba, capa por capa. El filamento se alimenta a través de un extrusor que lo calienta hasta fundir. Luego, el filamento se extruye a través de una boquilla y se deposita a lo largo de la geometría de la pieza. Dado que el filamento se extruye en un estado fundido, el material recién depositado se fusiona con el adyacente depositado anteriormente. Además, el filamento recién extruido tiene una sección circular, mientras que, al ser depositado y presionado por la boquilla y las siguientes capas depositadas, pasa a tener una sección elíptica [11].

2.2. MODELADO DE PIEZAS 3D

Aunque como tecnología de impresión, la MEX es muy flexible, tiene algunas restricciones a la hora de predecir el comportamiento mecánico. La pieza fabricada por FA difiere con respecto a su modelo original en términos de mesoestructura, resistencia y propiedades del material, puesto que el modelo original es completamente sólido, mientras que la pieza fabricada tendrá una estructura interior formada por filamentos depositados. Actualmente, no existen métodos extendidos para el modelado de este tipo de piezas, por lo que normalmente se recurre a modelos geométricamente simplificados junto con métodos de análisis como el de por elementos finitos. Sin embargo, esta simplificación lleva a resultados poco asimilables al comportamiento real de las piezas impresas. Por ello, es necesario el desarrollo de metodologías para modelar, simular y analizar piezas de fabricación aditiva que se adapten con mayor precisión a los resultados de la pieza real, modelando las piezas a medida que se construyen, capa por capa. Además, esto último permitiría optimizar los parámetros geométricos asociados a la fabricación (como por ejemplo porcentaje de relleno, altura de capa, etc.) para conseguir las propiedades mecánicas requeridas con el mínimo peso.

En el caso de los scaffolds, no siempre se diseñan de la misma manera. Algunos autores definen la pieza sólida y luego establecen su porosidad, así como otros parámetros geométricos como el patrón de relleno o la altura de capa en la fase de laminado; por consiguiente, pueden obtenerse varias configuraciones de scaffolds a partir del mismo sólido en el software de laminado [13–15]. En este caso, la ruta de

impresión 3D se determina de acuerdo con los ajustes de impresión, y la geometría resultante se muestra en el software. Sin embargo, esta información geométrica no se puede exportar y, por lo tanto, no se puede utilizar para predecir el comportamiento mecánico mediante técnicas de simulación como el análisis de elementos finitos (AEF). Además, en caso de querer obtener estructuras internas que presenten superficies curvas, en los programas de laminado ordinarios, el giroide es el único patrón que cumple los requisitos de curvatura.

Algunos investigadores han analizado las limitaciones de los programas informáticos de laminación disponibles [16], y otros han desarrollado un algoritmo para obtener el código G para sus aplicaciones específicas [17–19]. En este campo de aplicación concreto, las principales limitaciones son la reducida gama de patrones de relleno y la imposibilidad de modificar sus parámetros. Para superar estas limitaciones cabe destacar un novedoso software de código abierto llamado FullControl GCode Designer [20]. Se creó para cubrir una amplia gama de requisitos de impresión y permite diseñar cualquier patrón mediante ecuaciones matemáticas sin restricciones.

Sin embargo, otros investigadores prefieren primero diseñar el modelo mediante software CAD, representándolo con filamentos en lugar de como una pieza sólida, y luego imprimir el scaffold [6,10,21–26]. En esos casos, pudieron simular mecánicamente el comportamiento del modelo y comparar los resultados con los experimentales u optimizar la pieza antes de imprimirla. A pesar de ello, en las referencias anteriores el proceso de impresión de este método no está adecuadamente definido; podría presentar problemas dependiendo del software de laminado elegido, ya que tienden a interpretar cada filamento como una pieza sólida y a representarlo como varios filamentos, en lugar de como una única pasada.

Por otro lado, existen varias opciones válidas para simular el comportamiento mecánico del scaffold impreso a partir de su modelo, como la homogeneización [27–30], las técnicas de modelado basadas en vóxeles [8,31,32] o las basadas en CAD [6,10,21–26,33,34]. Entre estas alternativas, el método de modelado basado en CAD es el más utilizado. Existen varias formas de obtener el modelo geométrico. Una de ellas es la definición de una capa, que se repite a lo largo de la altura, combinando la dirección de los filamentos. Este método tiene la limitación de que sólo es aplicable a prismas cuadrangulares o cilindros con las mismas capas mesoestructuradas (cuando la figura base se gira 90 grados, mantiene su forma). Otra forma consiste en modelar la pieza depositada mediante un software asistido por ordenador basado en la información del código G. Este método puede reproducir objetos geométricos simples

que se definen por sus límites (vértices, aristas y bucles). Sin embargo, podría plantear problemas si se aplica a sólidos no eulerianos o a estructuras porosas biomorfas. Además, también podría fallar si hay huecos o solapamientos, y su manipulación puede resultar difícil cuando los objetos tienen arquitecturas internas finas o si son demasiado grandes [35,36].

Para resolver las limitaciones computacionales de los modelos CAD, se utilizan celdas unitarias. Se trata de bloques de construcción básicos que representan las microarquitecturas de los scaffolds. Existen bibliotecas de celdas unitarias que, combinadas con la técnica de homogeneización (aplicación de las propiedades de la celda unitaria a todo el sólido), hacen que el método computacional sea más eficiente [35,36].

Por otro lado, también se utiliza el método de modelado basado en imágenes, que es compatible con la tomografía computarizada (TC) y la resonancia magnética (RM). Permite reproducir la arquitectura interna de las estructuras porosas a partir de imágenes binarias 3D que, a partir de los datos de densidad, definen los valores de los vóxeles (valores booleanos, sólido o vacío). Sin embargo, este método presenta limitaciones de resolución, además de una gran cantidad de datos y la necesidad de una imagen de referencia que se obtiene a partir de la pieza ya fabricada [35,36].

Estudios recientes han introducido otra forma de modelado de estructuras porosas tomando como referencia geometrías naturales (por ejemplo, huesos). Esta técnica de modelado tiene en cuenta las irregularidades y la evolución espacial de la estructura porosa de referencia. Se trata del diseño *top-down* mediante el método de Teselación de Voronoi. Este considera la irregularidad de los huesos y controla la distribución y la forma de los poros, además del gradiente de porosidad interconectada. Se basa, principalmente, en la representación tridimensional de puntos semilla, que definirán el diagrama de Voronoi o polígonos de Thiessen. En primer lugar, se distribuyen los puntos en una malla regular dentro de una región que representa el volumen exterior de la estructura. A continuación, a estos puntos se les atribuye un volumen, correspondiente al de los poros, para luego ser redistribuidos de forma irregular. Por último, se generan las celdas de Voronoi, definiendo la estructura alrededor de los poros. El principal reto de este método es encontrar una tecnología de fabricación adecuada, que parece ser la de fusión de lecho de polvo con láser [37,38].

2.3. PREDICCIÓN DE RESULTADOS MEDIANTE SIMULACIÓN

A la hora de predecir el comportamiento de las piezas impresas mediante modelos que imitan a la pieza real, varios autores han considerado el uso del AEF [25,39–46]. Sin embargo, en algunos casos no han conseguido que los resultados de la simulación se asemejen a los experimentales [39,46]. Esto se debe a que la representación del modelo no se ajusta a la realidad de una pieza impresa, considerando la interacción de filamentos.

Es por ello que el desarrollo de un método modelado, combinado con una herramienta de simulación (para analizar el comportamiento mecánico de diferentes mesoestructuras de poros) permitiría reducir el trabajo experimental necesario para desarrollar un nuevo diseño (por ejemplo, nuevos scaffolds para su uso en la ingeniería tisular). Por lo tanto, esta herramienta mejoraría la rentabilidad del proceso de desarrollo [6].

2.4. OPTIMIZACIÓN DE PARÁMETROS

En el proceso de impresión 3D de un modelo, es fundamental la correcta elección de los parámetros de impresión para conseguir las características mecánicas, dimensionales y de masa deseadas. Dichos parámetros pueden clasificarse en parámetros de fabricación, que incluiría la temperatura de extrusión, la velocidad de impresión, la temperatura de la cama o condiciones ambientales; y parámetros estructurales o geométricos, como la densidad de relleno, la orientación de impresión, el ángulo de trama, la altura de capa, el número de perímetros o el patrón de relleno [11,41].

Varios autores han realizado estudios para seleccionar los parámetros más relevantes en la fabricación de piezas. Por ejemplo, O. Lužanin et al. [47] estudiaron la influencia de la altura de capa, el ángulo de trama y el relleno sobre la fuerza de flexión máxima en probetas de MEX. De la misma manera, F. Rayegani et al. [48] determinaron que la orientación de la pieza, el ángulo de trama, el ancho del filamento y los huecos (air gaps) afectan a la resistencia a la tracción. O.A. Mohamed et al. [49] añaden que el número de perímetros, la altura de capa y los huecos son los parámetros de proceso con más influencia en las propiedades dinamo-mecánicas.

En relación a la densidad de relleno y huecos, se tratarán como el mismo parámetro, aunque inversos. El porcentaje de relleno se refiere a la cantidad de material sólido depositado y, el otro, relacionado con la porosidad, a los huecos

presentes en el interior de la pieza. Así que cuanto mayor es uno, menor es el otro. Por un lado, un porcentaje de relleno más alto y, por tanto, menor tamaño de huecos, mejorará la rigidez del material, pero también supondrá un aumento del peso de la pieza. Lo ideal es encontrar el equilibrio entre la porosidad de la pieza y sus características mecánicas y de masa.

Con respecto al resto de parámetros nombrados anteriormente, para entender mejor qué representa cada uno y, por tanto, cómo pueden afectar al comportamiento mecánico, masa y dimensiones de la pieza impresa, se describen y representan en la Figura 1 [11].

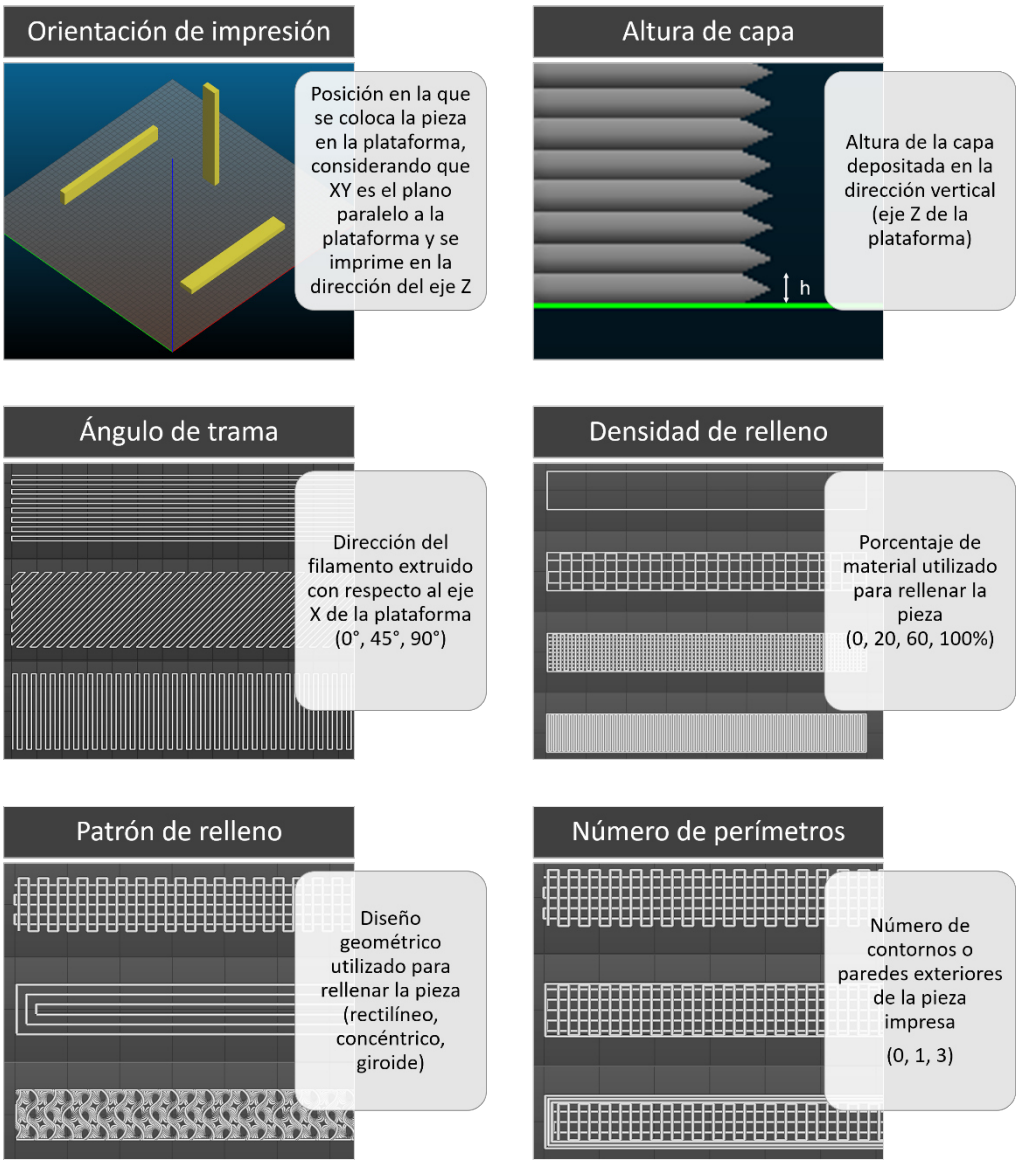


Figura 1. Definición de parámetros de impresión geométricos

La influencia de algunos de estos parámetros en el comportamiento mecánico y el peso de la pieza es predecible. Por ejemplo, cuanto mayor sea el porcentaje de

relleno, la rigidez de la pieza impresa será mayor, pero al utilizar más material, también aumentará el peso [40]. Pero, cuando se modifica más de un parámetro geométrico, es necesario estudiar cómo afecta la combinación y qué parámetros son más relevantes.

Varios autores han estudiado la influencia de los parámetros de impresión en las propiedades de la pieza impresa y han aplicado métodos de optimización para obtener la mejor configuración, según la pieza, la aplicación y el material utilizado. Entre los métodos más utilizados se encuentran los algoritmos genéticos [50,51], el método Taguchi o similar [49,52–54], el análisis paramétrico [55], el diseño compuesto central (CCD) [56], el método de tratamiento de datos en grupo (GMDH) combinado con evolución diferencial [48], la lógica difusa y la red neuronal artificial (ANN) [57,58]. Aunque, otros autores también han hecho uso de un experimento factorial 2^3 sin replicación con tres puntos centrales, complementado con un análisis de varianza (ANOVA) [47].

2.5. APLICACIÓN DE SCAFFOLDS EN LA INGENIERÍA TISULAR

En las aplicaciones de ingeniería tisular, se buscan estructuras porosas que favorezcan el crecimiento celular y la regeneración de tejidos. La morfología, el tamaño y la distribución de los poros desempeñan un papel fundamental no sólo en las propiedades mecánicas de la estructura [59], sino también en su rendimiento biológico. El tamaño de los poros tiene un valor óptimo para cada tipo de tejido. Un tamaño de poro inferior al óptimo dificulta la vascularización de la estructura y la migración celular. Por otra parte, los poros de gran tamaño dan lugar a una superficie reducida (y, por tanto, a una adhesión celular limitada) y a estructuras débiles cuando se utilizan polímeros biodegradables para fabricar la estructura. Por ejemplo, el proceso de osteointegración mejora cuando se utilizan scaffolds con tamaños de poro de entre 150 y 500 µm [60–62]. Además, la vascularización que se logra con poros interconectados con un tamaño de poro suficiente favorece el proceso de osteogénesis [63].

Además de su papel en los procesos biológicos que tienen lugar durante la regeneración in vivo del tejido de interés, la porosidad es relevante durante el proceso de degradación de los scaffolds biodegradables [64], ya que está relacionada con la permeabilidad de la estructura y, en consecuencia, con la eliminación de los subproductos de la degradación. Estas sustancias tienen un efecto autocatalítico, por lo que su concentración local tiene un fuerte impacto en la velocidad y el mecanismo

de degradación del scaffold [65].

En la ingeniería tisular se utilizan varias técnicas para la fabricación de scaffolds, que son las convencionales (lixiviación de partículas, separación de fases, espumación de gases o liofilización de emulsiones), y también las técnicas de electrospinning y fabricación aditiva (FA). En los últimos años, este último método es el más seleccionado. La FA permite un alto grado de control de la porosidad y ésta es una de las razones que explican su popularidad en el campo de la ingeniería de tejido [4–7].

Por otra parte, para optimizar el crecimiento celular, se ha observado que la regeneración ósea aumenta linealmente con la curvatura de las superficies disponibles para el anclaje celular, preferentemente en superficies cóncavas [66–70]. Por tanto, los scaffolds con curvaturas cóncavas tienen mayor potencial para la regeneración ósea.

Capítulo 3. Propuesta de modelado 3D

En este capítulo, se describirá la metodología de modelado, basada en geometría, desarrollada para esta tesis.

La subrutina de modelado creada se ha incluido en el repositorio de MathWorks, File Exchange, con el nombre AutomateD SwEep CAD modeller Of Extrusion-based G-coDE (DECODE) [71]. Además, los códigos completos de las subrutinas y funciones creadas en Matlab R2021a (The MathWorks, Inc., Natick, EE.UU.), se recogen en los Anexo A, Anexo B, Anexo C y Anexo D.

3.1. MODELADO BASADO EN GEOMETRÍA

En este trabajo, se ha desarrollado una metodología de modelado, titulada AutomateD SwEep CAD modeller Of Extrusion-based G-coDE (DECODE), basada en geometría. Es decir, interpreta las coordenadas dadas en el fichero G-code (archivo requerido para la fabricación de las piezas) y genera un modelo CAD.

3.1.1. Representación del filamento

Como su nombre indica, el modelado DECODE está basado en barridos, es decir, la manera de representar filamentos consiste en desplazar un perfil a lo largo de una trayectoria para generar una forma tridimensional de sección constante. Con lo cual, cada barrido correspondería a un tramo de filamento depositado por el cabezal de la impresora de manera continua. En la Figura 2 se muestra un ejemplo de barrido.

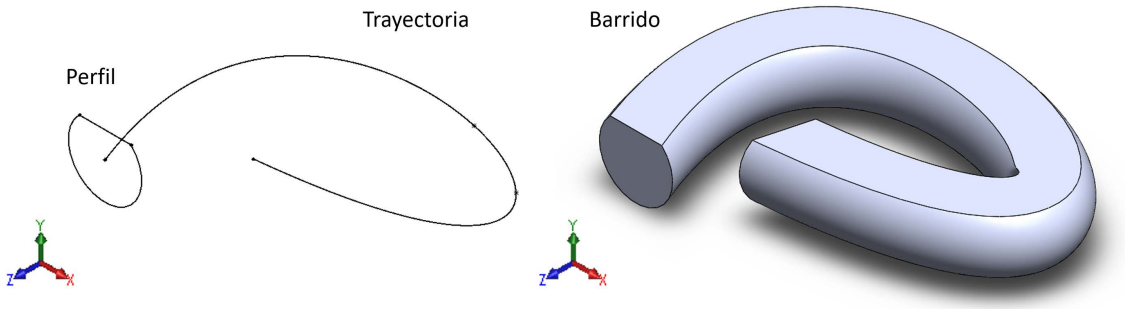


Figura 2. Ejemplo barrido

En este caso, se ha supuesto que el perfil del filamento tendrá una forma elíptica-rectangular y estará definido por los parámetros de extrusión, tal como se presenta en la Figura 3 [72].

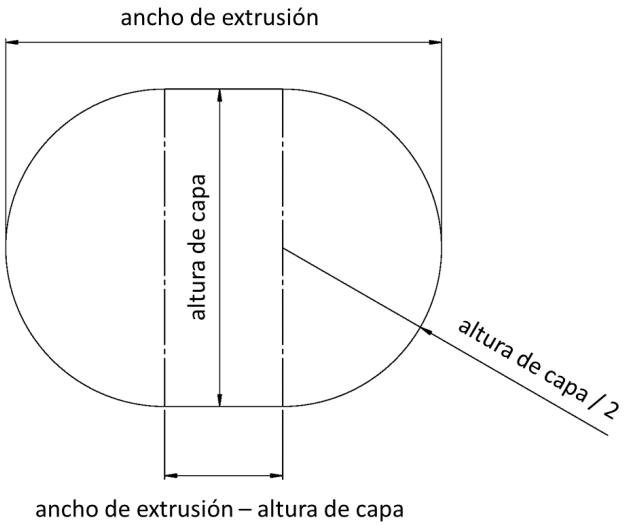


Figura 3. Perfil de filamento

3.1.2. Descripción del proceso

La generación del modelo CAD se lleva a cabo dentro del entorno del software de AEF, Abaqus/CAE 6.14-1, mediante la ejecución del archivo Python (.py). Este archivo contiene la información de los planos, trayectorias y perfiles que serán utilizados para generar los barridos de los filamentos.

Esta información se obtiene con el archivo de coordenadas, en formato .xlsx o .mat, que son interpretadas por una subrutina creada durante esta tesis, llamada *Abaqus_model* (su descripción completa se encuentra en el apartado 3.2). El volumen y dimensiones del modelo podrán obtenerse dentro del entorno de Abaqus/CAE 6.14-1. Asimismo, tras el AEF, mediante el postprocesado del análisis se pueden obtener las fuerzas de reacción u otros parámetros de interés para conocer el comportamiento mecánico del modelo.

3.1.3. Limitaciones y soluciones propuestas

Durante la creación de la metodología DECODE, se hallaron varias limitaciones tanto en la ejecución del modelo en Abaqus/CAE 6.14-1, como en el mallado del mismo.

En la etapa de modelado, después de importar el archivo Python, en algunas ocasiones (dependiendo de la geometría) el software informa de un error al detectar trayectorias cerradas, como las que presentan los perímetros, o auto-intersecciones. En la Figura 4, se muestra un ejemplo de una auto-intersección en una trayectoria cerrada, correspondiente a un perímetro, en comparación con una trayectoria cerrada sin intersección.

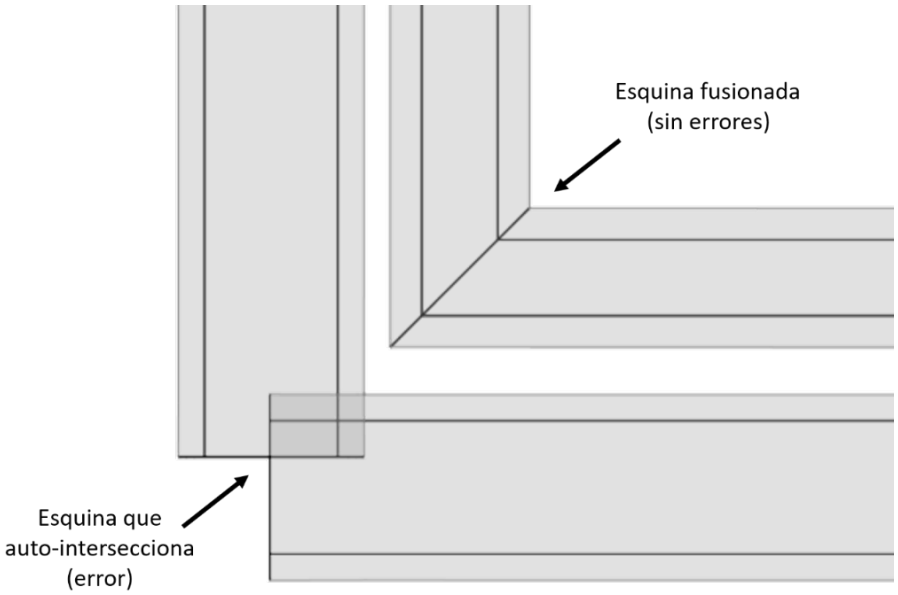


Figura 4. Ejemplo auto-intersección en trayectorias cerradas

Con el fin de solucionar estos errores en la etapa previa a la importación, se estudiaron varias subrutinas, cada una con distintas estrategias de modelado, para tratar de evitar los problemas de auto-intersecciones del modelo. Estas estrategias se aplicaron en un ejemplo concreto, para comparar los resultados y ver el comportamiento de cada estrategia. Dicho ejemplo de aplicación se corresponde con un ensayo a flexión de una miniprobeta (Figura 5), cuyas dimensiones son equivalentes a las de un octavo de probeta de flexión estándar según la norma ISO 178.

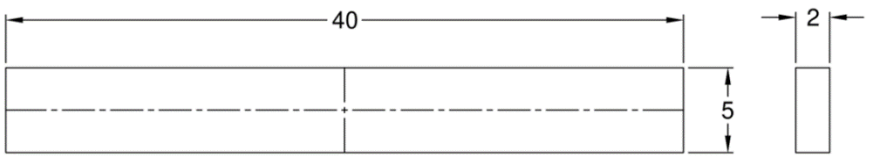


Figura 5. Dimensiones miniprobeta

En primer lugar, se estudió el tipo de operación de modelado a emplear. El modelado mediante extrusión de filamentos no llegó a llevarse a cabo y se descartó debido a la gran complejidad que presentaba. Este consistía en representar los filamentos mediante la operación de extrusión (cada tramo recto, correspondiente a un código G1, se representaría como una extrusión independiente). Para ello, haría falta crear planos perpendiculares a la trayectoria en el comienzo de cada tramo recto, para representar la sección, lo que se traduciría en una gran cantidad de planos y operaciones; al contrario que en el barrido, en el que sólo se necesitan los planos correspondientes a cada capa para dibujar el croquis de las trayectorias, puesto que el perfil que define la sección se ubica perpendicular al primer tramo automáticamente.

Por tanto, seleccionando el barrido como operación óptima, se barajaron distintas estrategias: el modelado de la pieza completa, el modelado línea a línea, el modelado línea a línea con revolución en las esquinas y el modelado por secciones.

3.1.3.1. Modelado de la pieza completa

La metodología más simple es en la que se modela la pieza entera según el G-code. En esta estrategia, cada trayectoria corresponde a una extrusión continua de filamento. En caso de existir una discontinuidad en la misma capa, cada tramo de la trayectoria es representada mediante un barrido. Para definir estas discontinuidades, en la matriz de coordenadas se añade una columna adicional (la cuarta, después de las tres correspondientes a las coordenadas X, Y y Z). Los croquis de las trayectorias son representados por líneas rectas que unen cada punto del archivo de coordenadas, de forma consecutiva, empezando en aquellos con valor "0" en la cuarta columna y continuando con cuyo valor es "1". Cada nueva coordenada indica un cambio de dirección del cabezal de impresión durante la extrusión y cada valor "0" en la cuarta columna, representa el posicionamiento de la boquilla en una coordenada sin extruir material desde su posición anterior.

Los modelos obtenidos mediante esta metodología presentan esquinas afiladas en ángulos muy agudos (Figura 6) y, en el caso de trayectorias cerradas, Abaqus/CAE 6.14-1 advierte de un error en el proceso de modelado.

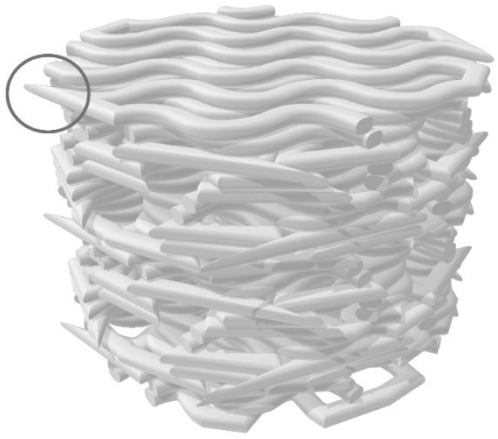


Figura 6. Ejemplo de esquinas afiladas

El problema de las trayectorias cerradas fue solucionado implementando en *Abaqus_model* la detección de este tipo de trayectorias. Si el primer y el último punto del barrido están demasiado cerca, en concreto, a una distancia inferior a la mitad del ancho de extrusión (distancia < ancho_extrusion /2), se considera que existe una trayectoria cerrada. Por tanto, se procede a separar el último tramo del barrido como

se muestra en la Figura 7. Si no se cumple la condición anterior, se considera una trayectoria abierta, así que no se realiza división.

	Columna	1	2	3	4
	Coordenada	X	Y	Z	-
Puntos	1	41.52	6.52	0.3	0
	2	2	6.52	0.3	1
	3	2	2	0.3	1
	4	41.52	2	0.3	-1
	5	41.52	6.46	0.3	1

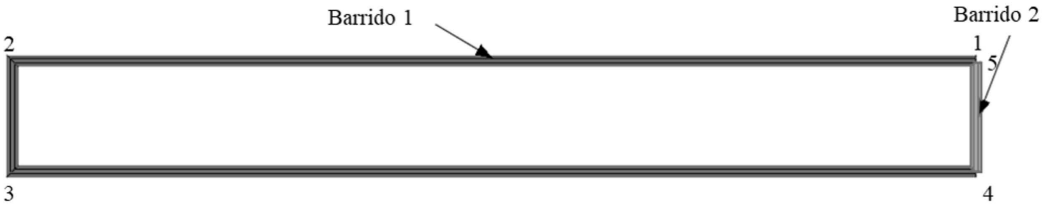


Figura 7. Solución trayectorias cerradas

Aunque esta actualización de la subrutina solucione los problemas encontrados sobre todo en piezas con perímetros, sigue sin aportar una solución al error de las auto-intersecciones, presentes en piezas más complejas como scaffolds con patrón de giroide. Estas intersecciones a menudo resultan en problemas de mallado en Abaqus/CAE 6.14-1 debido al solape de volúmenes. Se muestra un ejemplo de auto-intersección en la Figura 8.

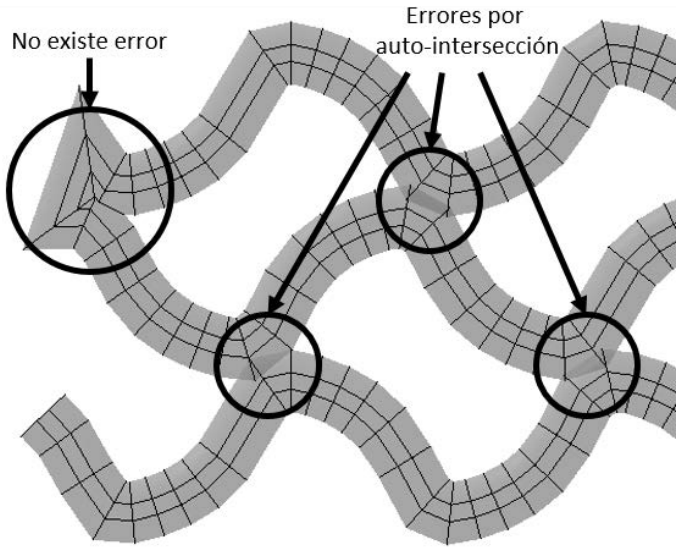


Figura 8. Ejemplo error por auto-intersección

3.1.3.2. Modelado línea a línea

Para evitar los problemas de la técnica de “modelado de la pieza completa”, se desarrolló otra estrategia denominada "modelado línea a línea", que consiste en crear un barrido por línea. Al realizase de esta manera, da lugar a esquinas vacías en las uniones (Figura 9).

	Columna	1	2	3	4
	Coordenada	X	Y	Z	-
Puntos	1	41.52	6.52	0.3	0
	2	2	6.52	0.3	1
	3	2	2	0.3	1
	4	41.52	2	0.3	1
	5	41.52	6.46	0.3	1

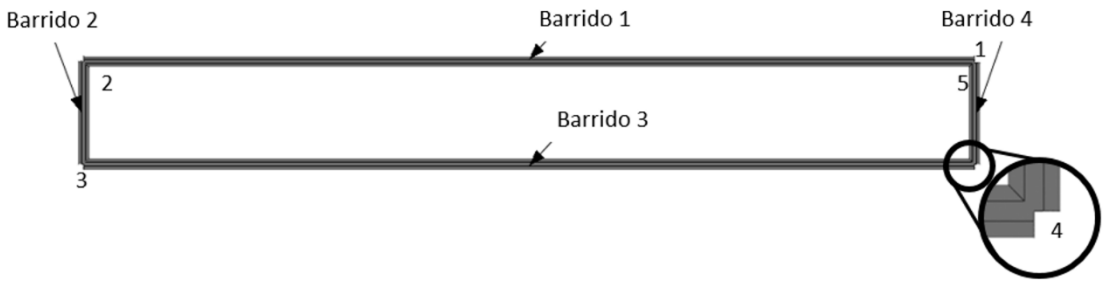


Figura 9. Resultado modelado línea a línea

Además, esta nueva estrategia dio lugar a nuevos errores que antes no se detectaban en Abaqus/CAE 6.14-1. Concretamente, el error en el proceso de modelado aparece al intentar realizar el barrido de ciertas líneas de la trayectoria que interfieren con puntos presentes en capas anteriores. Estos puntos corresponden a las esquinas generadas en las uniones. El error se muestra en la Figura 10.

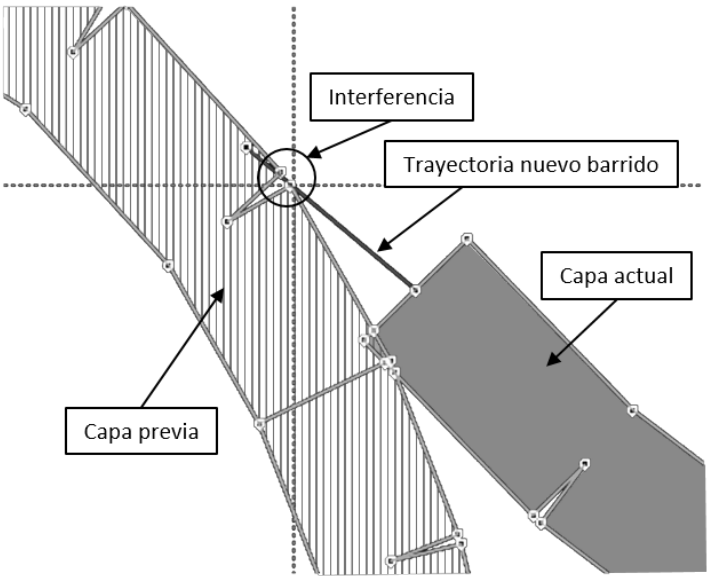


Figura 10. Error modelado línea a línea

3.1.3.3. Modelado línea a línea con revolución en las esquinas

El error identificado en el modelado línea a línea es solucionado al rellenar las esquinas vacías con revoluciones del perfil del filamento. Esta nueva estrategia es denominada "modelado línea a línea con revolución en las esquinas". El resultado de esta técnica se muestra en la Figura 11.

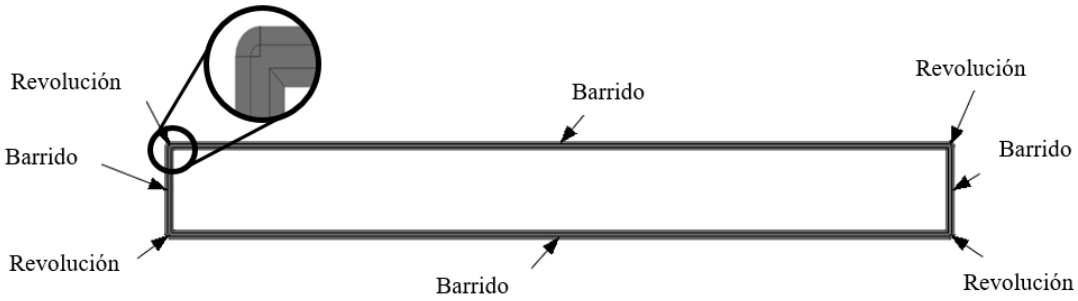


Figura 11. Modelado línea a línea con revolución en las esquinas

Esta técnica de modelado requiere la creación de planos perpendiculares a la trayectoria en cada esquina, para crear el croquis del perfil y generar la revolución con respecto al eje vertical. En consecuencia, cada línea y cada esquina requerirían varias operaciones, lo que ralentiza el proceso de modelado. Por este motivo, fue necesario diseñar otra estrategia que no requiriera separar cada línea de la trayectoria, pero evitara las auto-intersecciones. Esta nueva estrategia fue denominada "modelado por secciones".

3.1.3.4. Modelado por secciones

En la estrategia de "modelado por secciones" se dividen los barridos que presentan auto-intersección para evitar el error. Aunque el modelo presenta esquinas agudas como en la estrategia de "modelado de la pieza entera", se ha comprobado que su influencia no es demasiado relevante desde el punto de vista mecánico, al menos para la predicción de la rigidez en condiciones estáticas, ya que las fuerzas de reacción aparecen en los pilares de la pieza, es decir en las uniones entre filamentos, no en los filamentos en voladizo, como se muestra en la Figura 12.

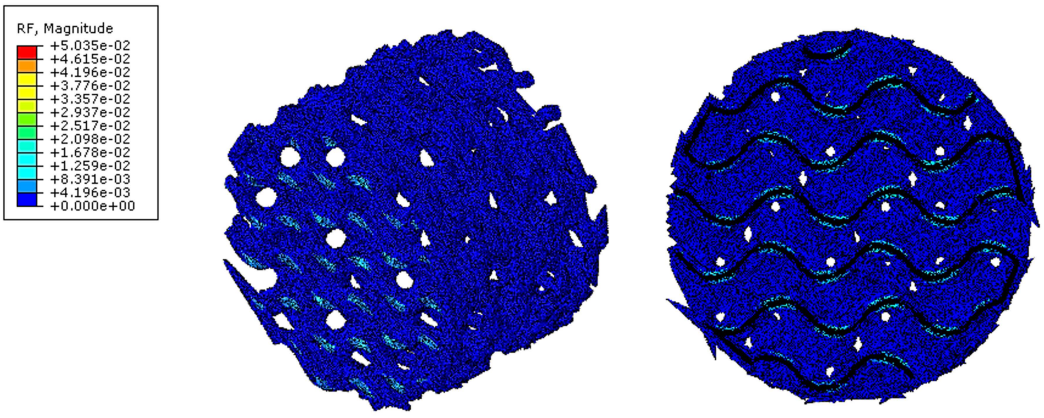


Figura 12. Representación de las fuerzas de reacción en un scaffold cilíndrico con patrón de relleno giroide al 50%

En esta estrategia, se sigue el proceso mostrado en la Figura 13 para la detección y resolución de problemas de auto-intersección. Las división automática de secciones mediante código de programación se expone en el apartado 3.2.3.

Primero, se identifican los picos de cada trayectoria (máximos y mínimos relativos) en las coordenadas X e Y. A continuación, se compara la distancia entre todos los máximos y mínimos relativos de la coordenada X, por una parte, y, por otra, todos los máximos y mínimos relativos de la coordenada Y. Si alguna distancia es inferior al ancho de extrusión y existen al menos 3 puntos entre los dos que se están comparando, hay una auto-intersección en esa coordenada (X o Y).

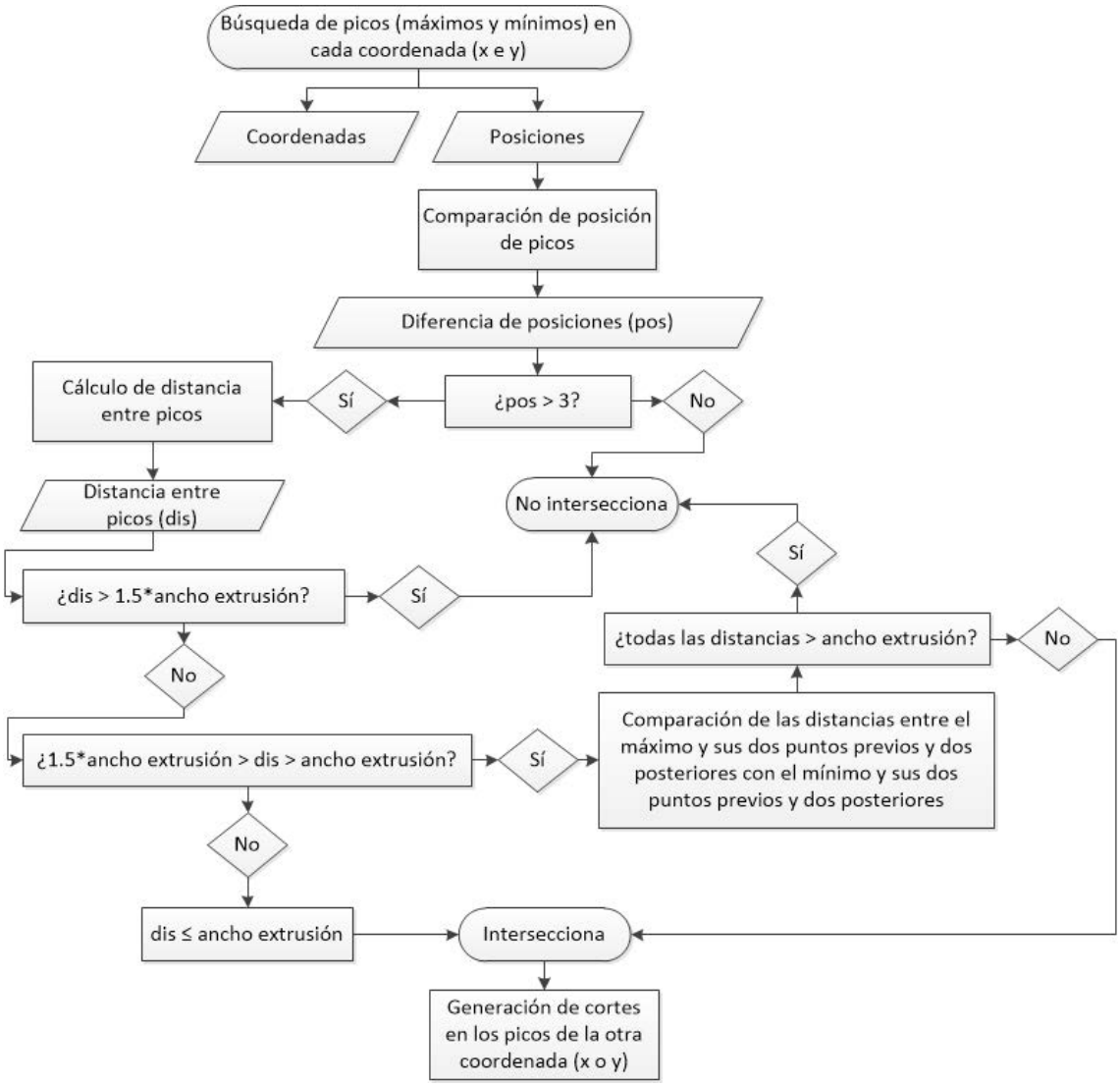


Figura 13. Diagrama resumido del modelado por secciones

La razón de excluir los puntos cercanos (menos de 3 puntos de distancia) es que en esos casos no habría auto-intersección en el modelado, ya que el problema de la auto-intersección sólo surge cuando la trayectoria se aleja del resto y se acerca de nuevo (Figura 8).

Si se detecta una auto-intersección en la coordenada X, la trayectoria se divide en los puntos correspondiente a picos de la coordenada Y, y viceversa. Este corte se representa en la matriz de coordenadas como un "-1" en la cuarta columna. Funciona de forma diferente a "0", ya que es el último punto del barrido anterior y el primero del siguiente.

En la Figura 14 se muestra un ejemplo de proceso de división, en el que las distancias entre picos son mayores que el ancho de extrusión, excepto para d1 y d2. Estas auto-intersecciones se encuentran en los picos de la coordenada Y, por lo tanto, se asigna un corte en el pico de la coordenada X mediante un "-1" en la cuarta columna.

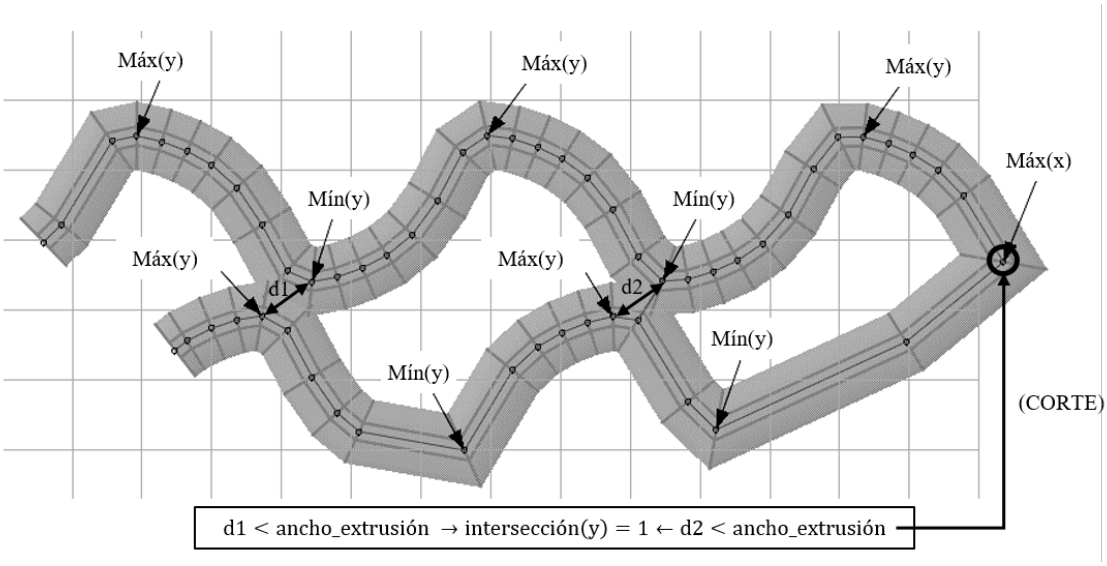


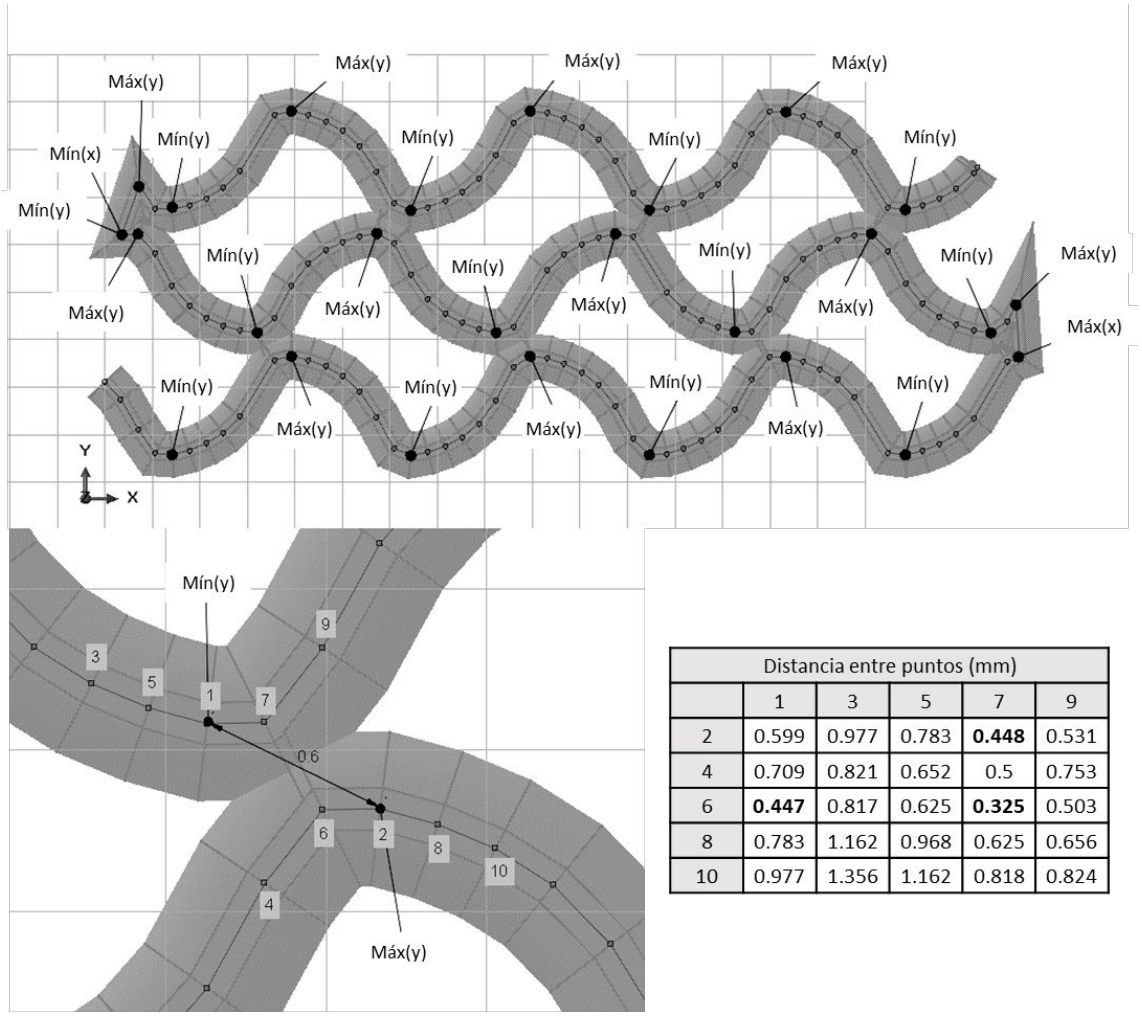
Figura 14. División del barrido en modelado por secciones

Sin embargo, existe la posibilidad de que exista una auto-intersección en el barrido, pero que no coincida exactamente con los picos. Por lo tanto, si dos puntos están próximos (la distancia es mayor que el ancho de extrusión, pero menor que el 150% de este ancho), se comparan también el punto máximo y sus dos puntos previos y posteriores con el punto mínimo y sus dos puntos previos y posteriores.

Como ejemplo, en la Figura 15 se muestra la identificación de los máximos y mínimos relativos de una trayectoria de barrido. Algunos de ellos cumplen la condición de estar a una distancia menor que el ancho de extrusión. Sin embargo, se muestra un ejemplo de un máximo y un mínimo que están cerca, pero los puntos que hacen que el barrido auto-intersecte no son los picos (los puntos 1 y 2), sino la distancia entre los puntos 2 y 7, 1 y 6 ó 6 y 7.

Además, en esta estrategia también se integra la identificación de trayectorias cerradas. Cabe señalar que, si no existen auto-intersecciones de trayectorias, el

modelo resultante será igual al obtenido con el "modelado de la pieza completa". Por lo tanto, la técnica de “modelado por secciones” es más adecuada para geometrías complejas, mientras que para piezas sencillas podría utilizarse la primera técnica.



Distancia entre puntos (mm)					
	1	3	5	7	9
2	0.599	0.977	0.783	**0.448**	0.531
4	0.709	0.821	0.652	0.5	0.753
6	**0.447**	0.817	0.625	**0.325**	0.503
8	0.783	1.162	0.968	0.625	0.656
10	0.977	1.356	1.162	0.818	0.824

$$\text{distancia}_{1-2} = 0.6 > \text{ancho\_extrusión} = 0.48$$

Figura 15. Auto-intersección entre puntos cercanos (no picos)

3.1.3.5. Caso práctico: comparación de estrategias de modelado

A continuación, se realiza la comparación de todas las estrategias de modelado para decidir cuál es la idónea para evitar errores, de forma automática y sencilla.

Las diferencias entre las geometrías resultantes al aplicar las distintas estrategias desarrolladas se muestran en la Figura 16, tanto para piezas simples como complejas (con auto-intersección).

Se observan esquinas vacías en la división de la última sección del "modelado de la pieza completa" y el "modelado por secciones"; en la división de trayectorias que auto-interceptan en el "modelado por secciones"; y entre cada tramo del "modelado

línea a línea". Y, por otro lado, tanto en el "modelado de la pieza completa" como en el "modelado por secciones" se pueden encontrar esquinas afiladas.

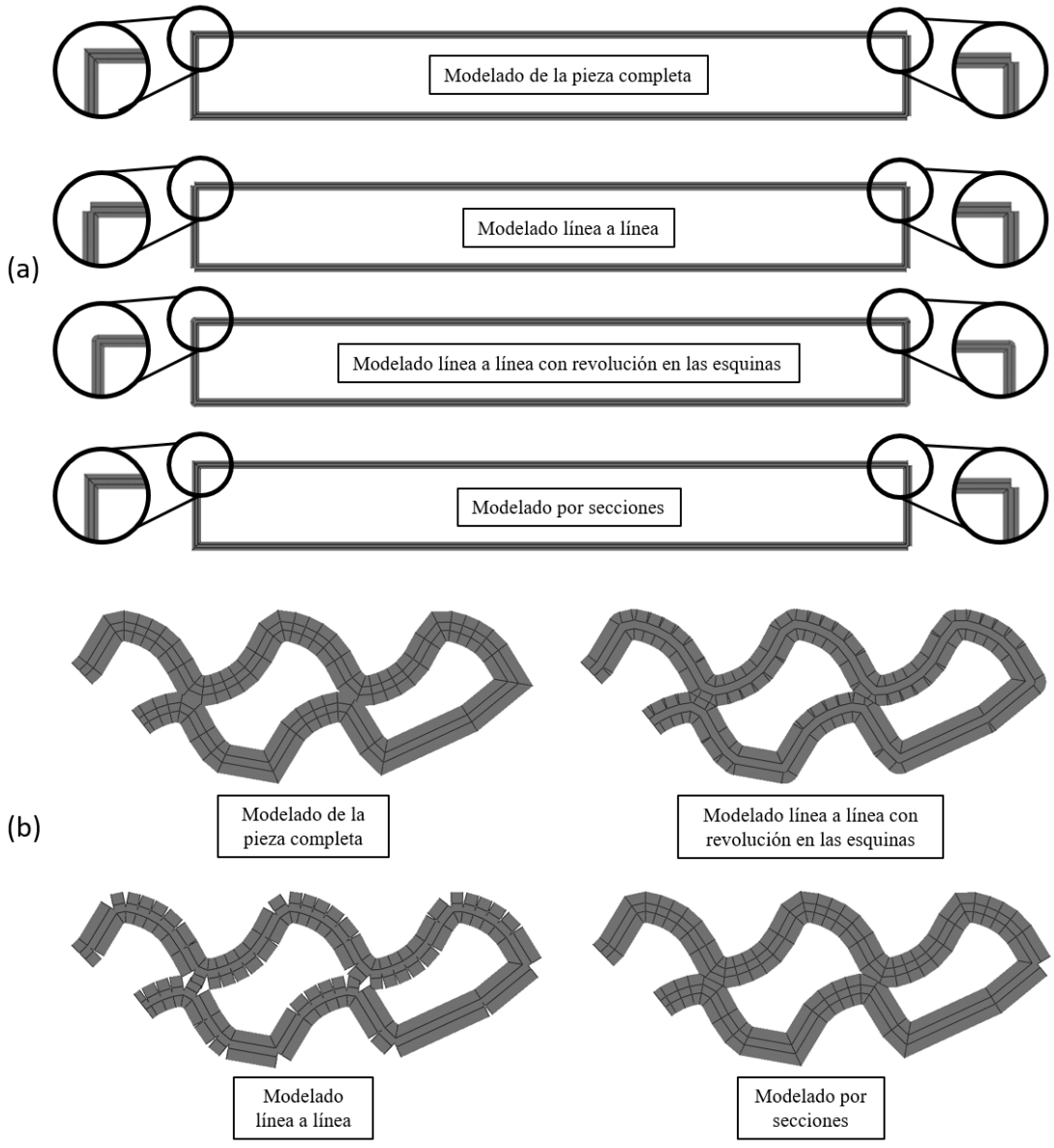


Figura 16. Comparación de estrategias de modelado en piezas simples (a) y complejas (b)

Por último, puede observarse que, en las piezas complejas, el "modelado línea a línea", el "modelado línea a línea con revolución en las esquinas" y el "modelado por secciones" evitan las auto-intersecciones.

Para validar las estrategias de modelado, se aplicaron a una probeta (40 x 5 x 2 mm) siguiendo los pasos de la metodología DECODE. Para obtener el archivo G-code, se introdujo el STL de la pieza sólida en Slic3r 1.3.0 y se fijaron los parámetros de impresión que figuran en la Tabla 1.

Tabla 1. Parámetros de impresión de la probeta

Parámetros de impresión	
Ancho de extrusión de perímetros externos	0.48 mm (1.87 mm^3/s)
Ancho de extrusión de perímetros	0.48 mm (3.74 mm^3/s)
Ancho de extrusión de relleno	0.48 mm (3.74 mm^3/s)
Diámetro de filamento	1.75 mm
Velocidad de relleno	30 mm/s
Diámetro de boquilla	0.4 mm
Temperatura	200 °C
Altura de capa	0.3 mm
Altura de primera capa	0.3 mm
No. de perímetros/capas sólidas	1
Ángulo de relleno	0°
Densidad de relleno	20%
Patrón de relleno	Concéntrico

Los modelos, obtenidos mediante cada estrategia de modelado, se simularon mediante un ensayo de flexión de tres puntos en Abaqus/CAE 6.14-1, como se muestra en la Figura 17. Para ello, se diseñaron tres piezas cilíndricas rígidas con un radio y una altura de 5 mm y se estableció una interacción superficie-superficie entre la probeta y cada cilindro. Se asignó un coeficiente de fricción de 0.15 al comportamiento tangencial de la interacción y el comportamiento normal se definió como un contacto rígido. Los modelos se mallaron con una malla tetraédrica de segundo orden y un tamaño de semilla de 1 mm. Los soportes se encastraron y se aplicó a la cruceta un desplazamiento vertical descendente de 1 mm.

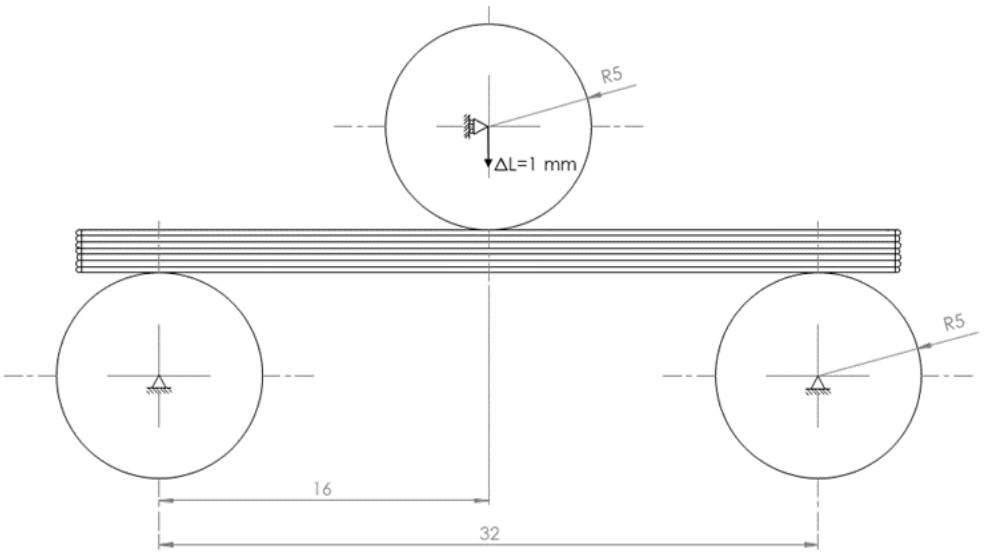


Figura 17. Ensayo a flexión de la probeta

Una vez definidas todas las condiciones anteriores, se obtuvieron las fuerzas de reacción mediante la aplicación del AEF. Los resultados obtenidos para las distintas estrategias se muestran en la Tabla 2.

Tabla 2. Comparación de resultados simulación de las estrategias de modelado

Estrategia de modelado	Fuerza de reacción (N)
Modelado de la pieza completa	13.49
Modelado línea a línea	13.47
Modelado línea a línea con revolución en las esquinas	13.47
Modelado por secciones	13.49

Las fuerzas de reacción resultantes no varían significativamente entre las distintas estrategias de modelado. Además, son iguales en el "modelado de la pieza completa" y en el "modelado por secciones", debido a que, en las piezas sin auto-intersección, sus geometrías son iguales.

Por este motivo, el "modelado por secciones" fue seleccionado como mejor estrategia, ya que se producen menos errores durante el proceso y fue posible automatizarlo.

3.2. CREACIÓN DE SUBRUTINA DE MODELADO

La subrutina creada en esta tesis para el modelado de piezas en 3D consta de una subrutina principal, denominada Abaqus_model, y varias funciones.

La subrutina principal es la encargada de crear, a partir de las coordenadas extraídas del código G, un archivo Python (.py), que generará el modelo al ser ejecutado en Abaqus/CAE 6.14-1.

En la Figura 18 se muestra el resumen del proceso de modelado propuesto, implementado en la subrutina Abaqus_model. En primer lugar, la subrutina requiere de un archivo de entrada en formato G-code, correspondiente a la pieza que se quiere modelar. Al introducir el G-code, la subrutina detecta en qué laminador se ha generado dicho archivo y llama a la función correspondiente para su interpretación y la obtención de la matriz de coordenadas y el diámetro del filamento. Con los datos devueltos, se escribe el archivo Python que, al ser ejecutado en Abaqus/CAE, genera los planos y ejes para la representación de trayectorias y perfiles y, finalmente, los barridos del modelo.

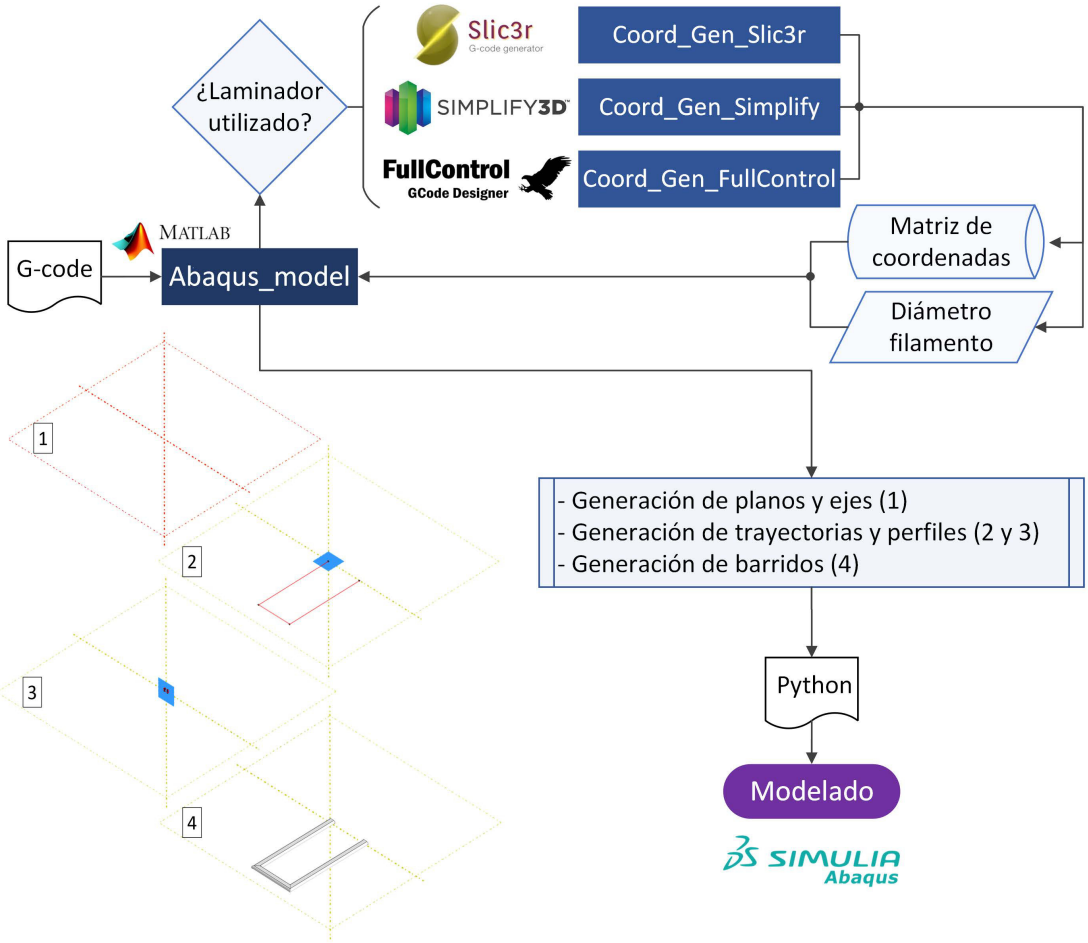


Figura 18. Resumen metodología de modelado DECODE

Para crear la matriz de coordenadas utilizada en la subrutina de modelado, se requieren funciones adaptadas a los diferentes lenguajes específicos de cada generador de código G. En este trabajo se han desarrollado tres funciones, Coord_Gen_Slic3r, Coord_Gen_Simplify y Coord_Gen_FullControl, para cada software de procedencia del archivo de entrada, Slic3r, Simplify 3D y FullControl G-code Designer, respectivamente.

En los siguientes subapartados se explicará el formato del archivo de entrada, la estructura del archivo de salida que permitirá modelar la pieza 3D, la programación de la subrutina principal encargada de generar dicho archivo de salida (Abaqus_model) y las tres funciones desarrolladas para leer las coordenadas según la procedencia del archivo de entrada (.gcode).

3.2.1. Formato archivo de entrada

El archivo de entrada tendrá la extensión *.gcode*. Además, para el correcto

funcionamiento de la programación, se requiere generar el archivo G-code con los comentarios activados, ya que estos son utilizados para interpretar los movimientos e identificar el comienzo de las trayectorias. En la Figura 19, se muestra un ejemplo de archivo G-code, resaltando las líneas claves para su interpretación.

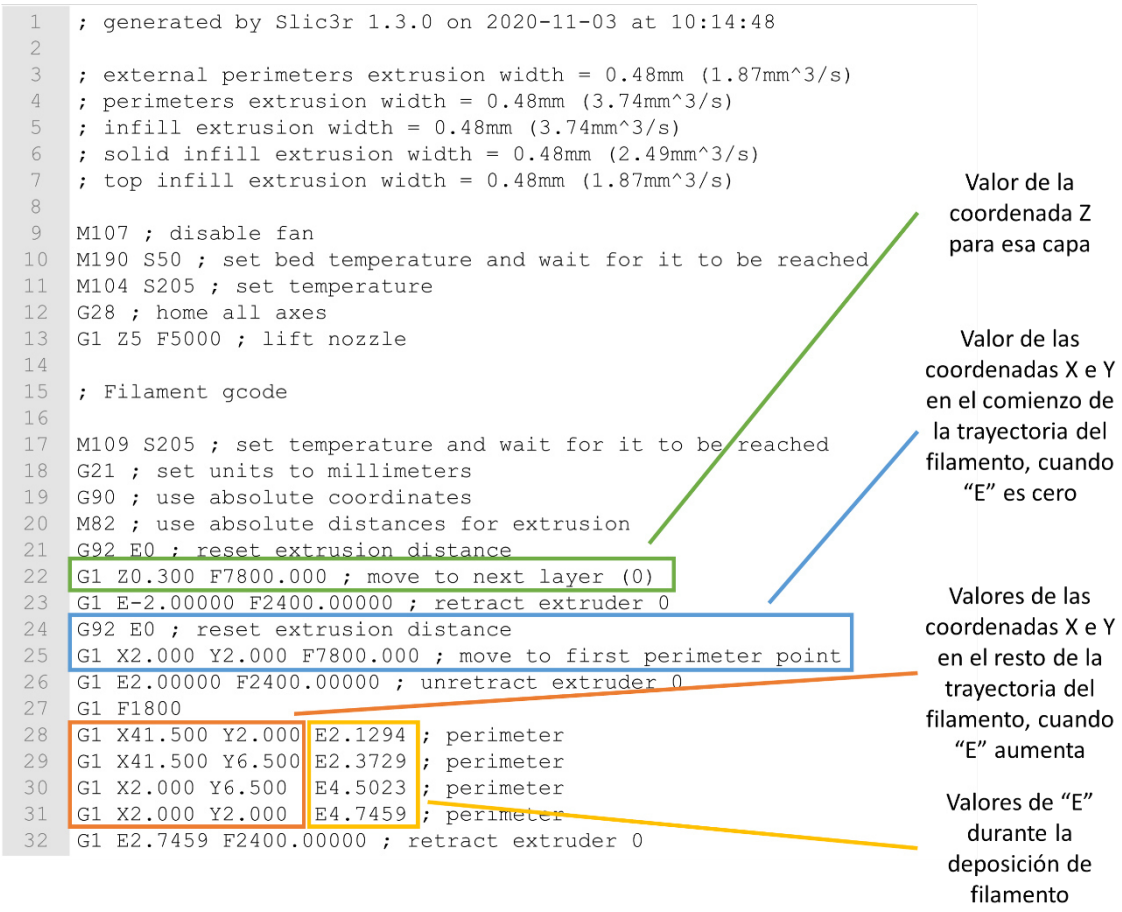


Figura 19. Formato del archivo G-code con indicaciones para detectar las trayectorias (generado en Slic3r)

3.2.2. Formato archivo de salida

El archivo de salida para modelar la pieza en 3D tendrá la extensión *.py* y contendrá la programación necesaria para que Abaqus/CAE 6.14-1 genere el modelo deseado de manera automática. Como se ha comentado anteriormente, este archivo de salida se generará directamente a partir de la ejecución de la subrutina Abaqus_model y la función correspondiente, y tendrá una estructura como la que se detalla a continuación.

En primer lugar, todos los archivos de salida presentan el siguiente encabezado para la importación de los diferentes módulos de Abaqus/CAE.

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
```

Concretamente, los módulos presentados permiten la creación de bocetos y piezas, la definición de propiedades de materiales, la asignación de estas propiedades a las piezas, la creación de ensamblajes, la definición de restricciones e interacciones, la asignación de propiedades de contacto, la definición de pasos de análisis de cargas o desplazamientos, la asignación de condiciones de contorno, la definición de cargas, la generación y manipulación de mallas, la optimización, la ejecución de análisis, y la gestión y visualización de resultados, entre otras funciones.

Tras el encabezado, se empieza a definir la primera trayectoria que seguirá el barrido. Las trayectorias se describirán mediante rectas; con lo cual, se deben detallar los puntos de inicio y final de cada una de ellas.

```
mdb.models['Model-1'].ConstrainedSketch(name='__sweep__',
sheetSize=20.000)
mdb.models['Model-1'].sketches['__sweep__'].Line(point1=(8.962,
2.482),  point2=(4.664, 2.482))
```

La subrutina *mdb.models['Model-1']* crea un nuevo objeto, denominado "Model-1". Además, en este modelo se crea el boceto llamado "sweep" mediante la función *ConstrainedSketch*, con un tamaño de hoja cuyas unidades son definidas mediante *sheetSize*. Seguidamente, dentro de este boceto, se dibuja la línea definida por los puntos de inicio y fin (*point1* y *point2*) mediante la función *Line*. Dichos puntos se especifican mediante las coordenadas (x, y).

Seguidamente, se define el perfil del barrido. En este caso, la forma del filamento es elíptica-rectangular (Figura 20), formado por dos líneas y dos arcos.

Además, se deberá tener en cuenta la orientación tridimensional del croquis, que deberá contenerse en un plano perpendicular a la primera recta de la trayectoria. La secuencia de números que se introducirán en el argumento *transform* es una matriz de transformación de 3x4 que especifica el eje de rotación (ROT) y el vector de traslación (v). El valor de esta matriz se halla como se muestra en la Ecuación (1).

$$T = \begin{pmatrix} ROT \\ v \end{pmatrix}_{3x4} \quad (1)$$

donde:

$$ROT_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-90°) & -\operatorname{sen}(-90°) \\ 0 & \operatorname{sen}(-90°) & \cos(-90°) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

$$ROT_y = \begin{pmatrix} \cos(0°) & 0 & \operatorname{sen}(0°) \\ 0 & 1 & 0 \\ -\operatorname{sen}(0°) & 0 & \cos(0°) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$ROT_z = \begin{pmatrix} \cos(\theta_z) & -\operatorname{sen}(\theta_z) & 0 \\ \operatorname{sen}(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\theta_z = \tan^{-1}\left(\frac{x_2 - x_1}{y_2 - y_1}\right)$$

$$ROT = ROT_x \cdot ROT_y \cdot ROT_z \cdot I$$

$$v = (x_1 \quad y_1 \quad (z_1 - altura\_capa))$$

En las ecuaciones, x_1, y_1, z_1 representan las coordenadas del punto de inicio y x_2, y_2 las coordenadas del punto final de una línea de la trayectoria. Además, se asume que siempre la rotación del eje X será de -90° y 0° del eje Y. El ángulo de rotación del eje Z dependerá del ángulo que forma la línea de la trayectoria con el sistema global de coordenadas.

Por otro lado, las dimensiones del perfil dependerán de la altura de capa y ancho de extrusión, tal como se muestra en la Figura 20.

```
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
sheetSize=20.000,transform=(0.000000,  1.000000,  0.000000,  0.000000,
0.000000, 1.000000, 1.000000, 0.000000, 0.000000, 8.962, 2.482,0.000))
mdb.models['Model-1'].sketches['__profile__'].ConstructionLine
(point1=(-20.000,0.0), point2=(20.000, 0.0))
mdb.models['Model-1'].sketches['__profile__'].ConstructionLine
(point1=(0.0,-20.000), point2=(0.0, 20.000))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-0.065,
0.350), point2=(0.065, 0.350))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-0.065,
0.000), point2=(0.065, 0.000))
mdb.models['Model-1'].sketches['__profile__'].ArcByStartEndTangent
(point1=(0.065,0.000), point2=(0.065,0.350), vector=(1.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].ArcByStartEndTangent
(point1=(-0.065,0.350), point2=(-0.065,0.000), vector=(-1.0, 0.0))
```

De la misma manera que se definió el boceto de la trayectoria de barrido, con *ConstrainedSketch* se genera el boceto del perfil ("profile"), pero aplicando una transformación afín que permite ajustar su posición y orientación en relación con el sistema de coordenadas. Para ello, antes se dibujan dos líneas de construcción mediante la función *ConstructionLine,* una horizontal y otra vertical, pasando por el origen. A continuación, se dibujan las líneas rectas que definen la parte alta y baja del rectángulo y los semicírculos que las unen. Estos semicírculos se bocetan mediante la función *ArcByStartEndTangent,* en la que se especifican el punto de inicio (*point1*), el punto final (*point2*) y el vector de cual tomará la dirección para hallar la tangente. Las coordenadas para la definición del perfil se muestran en la Figura 20.

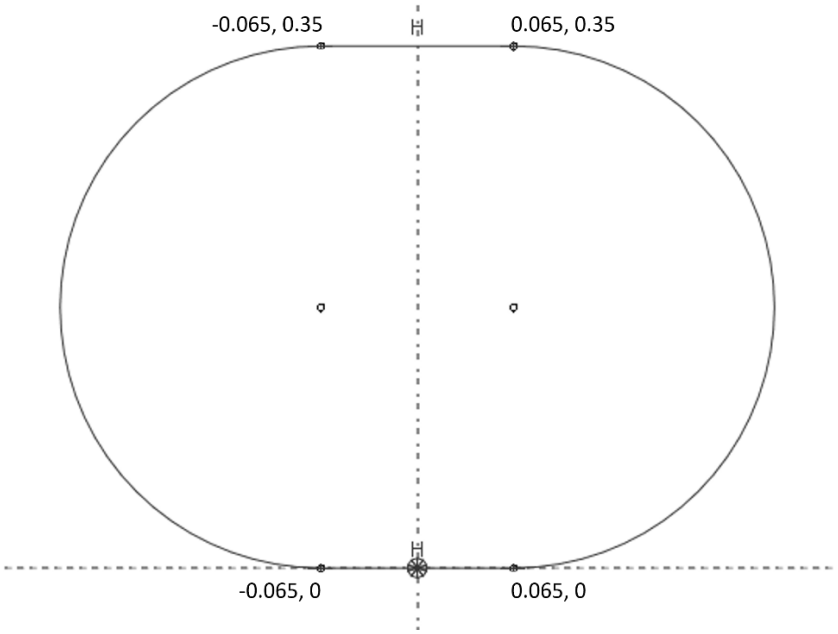


Figura 20. Definición del perfil del barrido con coordenadas

En este caso, en el que *point1* es (8.962, 2.482) y *point2* (4.664, 2.482) y, por tanto, θ_z es -90°, el valor de la matriz de transformación es la que se muestra en el Ecuación (2). En la Figura 21 se muestra los pasos de rotación y traslación para el correcto posicionamiento del perfil con respecto a la trayectoria y, así, generar el barrido correspondiente.

$$T = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 8.962 & 2.482 & 0 \end{pmatrix}_{3x4} \quad (2)$$

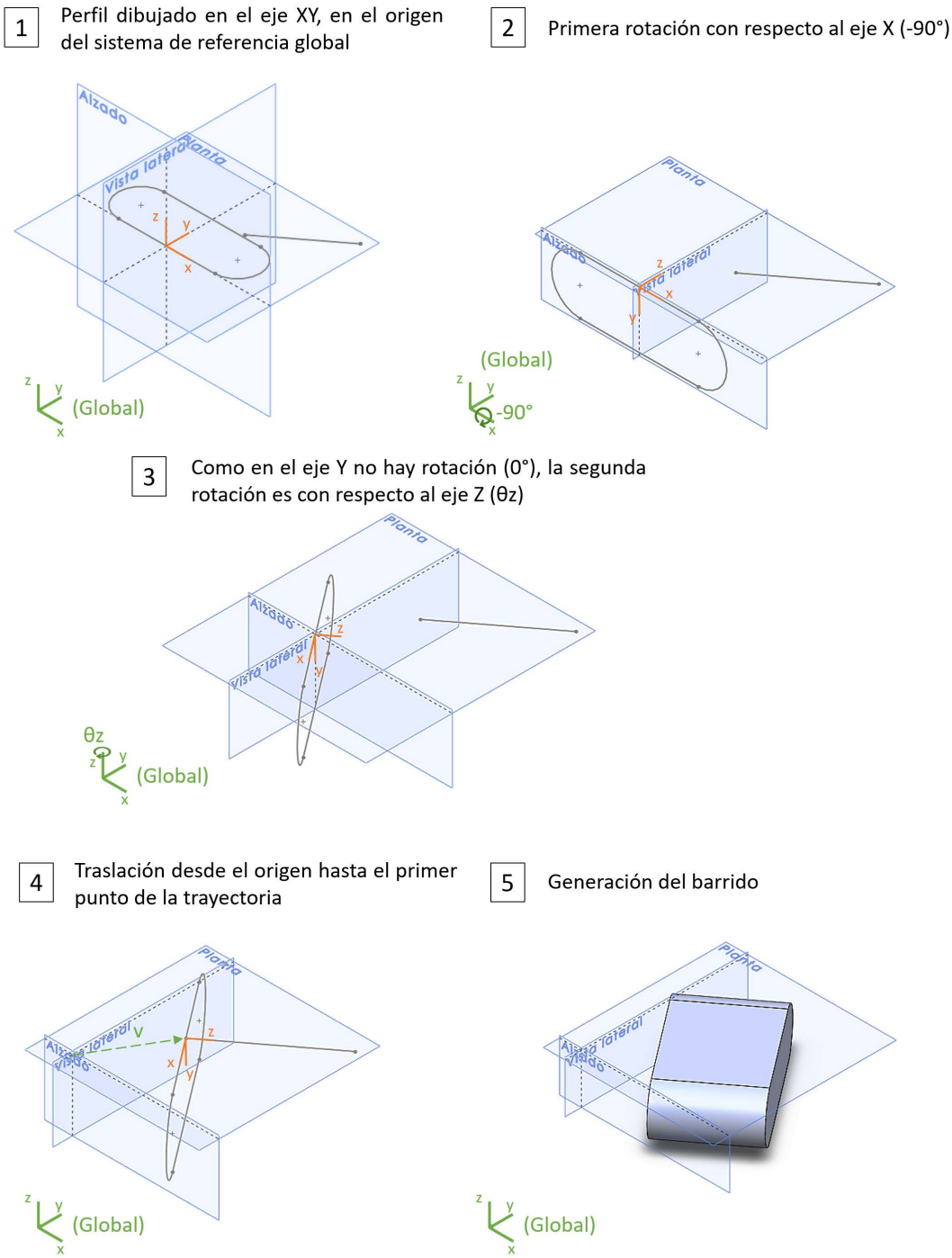


Figura 21. Proceso de rotación y traslación del perfil para el barrido

Para finalizar con el primer filamento, se ejecuta el barrido, asociándolo a la pieza correspondiente y especificando que es un cuerpo deformable.

```
mdb.models['Model-1'].Part(dimensionality=THREE_D,      name='Part-1',
type=DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-1'].BaseSolidSweep(path=
mdb.models['Model-1'].sketches['__sweep__'],sketch=
mdb.models['Model-1'].sketches['__profile__'])
```

Para crear la pieza dentro del modelo, se recurre a la función *Part*, la cual requiere que se definan los parámetros *dimensionality*, *name* y *type*, con los que se le atribuirá la condición tridimensional, el nombre y la característica de sólido deformable. La función *BaseSolidSweep* será la encargada de generar el barrido de sólidos, utilizando el boceto "sweep" como trayectoria (*path*) y el boceto "profile" como perfil (*sketch*).

A continuación, se definen los ejes Y y Z y los planos paralelos a XY, en los que se representarán las trayectorias de cada capa, por ello será necesario definir uno a la altura de cada capa.

```
mdb.models['Model-1'].parts['Part-1'].DatumAxisByPrincipalAxis
(principalAxis=YAXIS)
mdb.models['Model-1'].parts['Part-1'].DatumAxisByPrincipalAxis
(principalAxis=ZAXIS)
mdb.models['Model-1'].parts['Part-1'].DatumPlaneByPrincipalPlane
(offset=0.000,principalPlane=XYPLANE)
mdb.models['Model-1'].parts['Part-1'].DatumPlaneByPrincipalPlane
(offset=0.350,principalPlane=XYPLANE)
mdb.models['Model-1'].parts['Part-1'].DatumPlaneByPrincipalPlane
(offset=0.650,principalPlane=XYPLANE)
```

La función *DatumAxisByPrincipalAxis* crea un eje de referencia en la pieza “Part-1” del modelo “Model-1”, según el *principalAxis* que se le asigne, con *YAXIS* se indica que el eje se alinea con el eje Y principal y, con *ZAXIS*, que se alinea con el eje Z principal. Por otro lado, la función *DatumPlaneByPrincipalPlane* crea planos de referencia paralelos al plano principal seleccionado (*principalPlane*), con un desplazamiento del valor establecido en el *offset*.

Después de representar todos los planos, se continuará con la programación de los barridos para representar el resto de los filamentos. Hasta ahora sólo se había representado el primer tramo. En este caso, existen ligeras diferencias en el código de modelado, ya que es necesario indicar el origen, los ejes y el plano que se toman como referencia (identificados como *datum*) y su orientación con respecto a ellos.

```
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=0.740,
name='__sweep__',sheetSize=20.000,   transform=mdb.models['Model-1'].
parts['Part-1'].MakeSketchTransform(sketchPlane=
mdb.models['Model-1'].parts['Part-1'].datums[4],
sketchPlaneSide=SIDE1,sketchUpEdge=mdb.models['Model-1'].
parts['Part-1'].datums[2],sketchOrientation=LEFT,  origin=(0.0,  0.0,
0.000)))
mdb.models['Model-1'].sketches['__sweep__'].
Line(point1=(8.962,11.098), point2=(4.664, 11.098))
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
sheetSize=20.000,transform=(0.000000,  1.000000,  0.000000,  0.000000,
0.000000,    1.000000,    1.000000,    0.000000,    0.000000,    8.962,
11.098,0.000))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-0.065,
0.350), point2=(0.065, 0.350))
```

```
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-0.065,
0.000), point2=(0.065, 0.000))
mdb.models['Model-1'].sketches['__profile__'].
ArcByStartEndTangent(point1=(0.065,0.000),      point2=(0.065,0.350),
vector=(1.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].
ArcByStartEndTangent(point1=(-0.065,0.350),  point2=(-0.065,  0.000),
vector=(-1.0, 0.0))
mdb.models['Model-1'].parts['Part-1'].SolidSweep(path=
mdb.models['Model-1'].sketches['__sweep__'],    pathOrientation=LEFT,
pathPlane=mdb.models['Model-1'].parts['Part-1'].datums[4],
pathUpEdge=mdb.models['Model-1'].parts['Part-1'].datums[2],
profile=mdb.models['Model-1'].sketches['__profile__'],
sketchOrientation=LEFT,sketchUpEdge=mdb.models['Model-1'].
parts['Part-1'].datums[3])
```

La generación del resto de barridos es similar a lo descrito anteriormente, pero se añaden ciertas funciones como el espaciado de la cuadrícula del boceto (*gridSpacing*) o la transformación afín de la pieza "Part-1" mediante *MakeSketchTransform*. Esta transformación se aplica utilizando un plano de referencia, en este caso "datums [4]", como el plano del boceto (*sketchPlane*). A continuación, para poder orientarlo, se debe indicar el lado del plano del boceto, en este caso "SIDE1", que quedará orientado hacia arriba y el eje de referencia, en este caso "datums [2]", que quedará a la izquierda o derecha del boceto (*sketchOrientation*). El origen del boceto se establece en *origin*.

Además, en la generación del barrido se añade una parte de orientación de bocetos, tanto de la trayectoria como del perfil, mediante planos y ejes de referencia (*datum*).

3.2.3. Subrutina Abaqus_model

La subrutina Abaqus_model es la encargada de generar el archivo de salida .py que permitirá modelar la pieza en 3D. El código completo se adjunta en el Anexo A. En el primer bloque de esta subrutina, se definirán los nombres de los archivos de entrada y salida, en los formatos *.gcode* y *.py*, respectivamente:

```
%NOMBRE ARCHIVOS
filename_gcode=input('Introduzca el nombre del archivo G-code
(terminado en .gcode): ','s');
filename_py=input('Introduzca el nombre del archivo Python (terminado
en .py): ','s');
```

Seguidamente, se abre el archivo *.gcode* para leer el encabezado y determinar con qué software se ha generado. Tras conocer su origen, se obtiene la matriz de coordenadas mediante las funciones *Coord_Gen_Slic3r*, *Coord_Gen_Simplify3D* o *Coord_Gen_FullControl*. A partir de esta matriz, se determinará su dimensión para

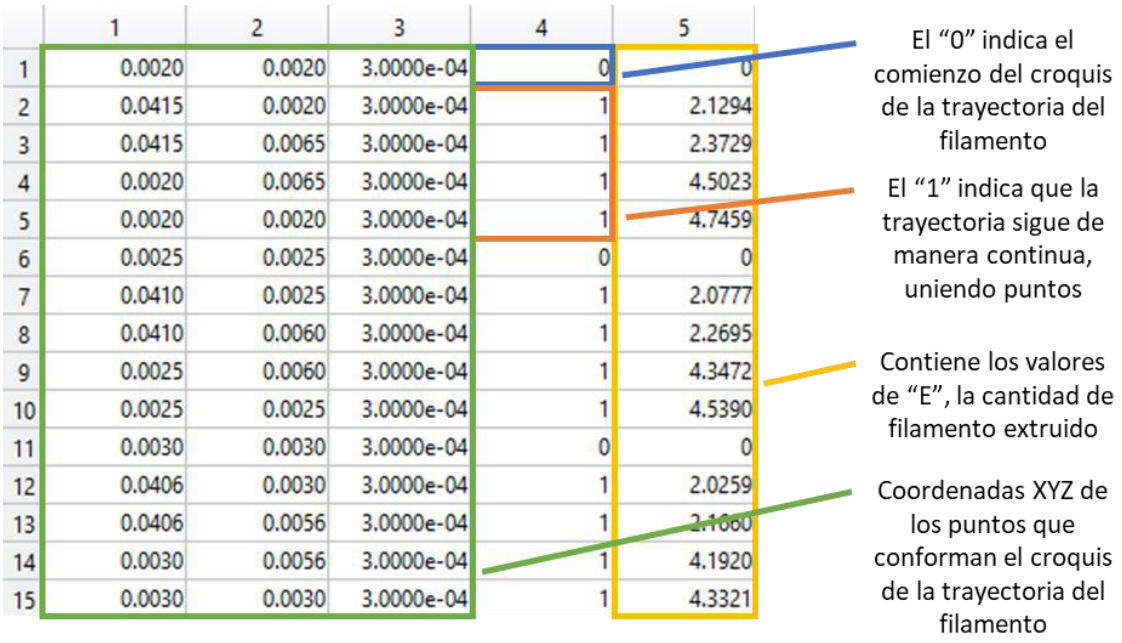
conocer el número total de coordenadas y se cambiará la unidad de sus valores a milímetros:

```
%LECTURA ENCABEZADO
ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
HEAD=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[1,20]);
fclose(ARCH); %Cierre del archivo

%LECTURA DE COORDENADAS
for i=1:1:20
   sim=cell2mat(strfind(HEAD{1,i},'Simplify3D(R)')); %Busca en cada celda 'Simplify3D'
   sli=cell2mat(strfind(HEAD{1,i},'Slic3r')); %Busca en cada celda 'Slic3r'
   fc=cell2mat(strfind(HEAD{1,i},'FLAVOR')); %Busca en cada celda el encabezado típico de FullControl
   if sim>0
      [coordinates,fil_D]=Coord_Gen_Simplify(filename_gcode);
   elseif sli>0
      [coordinates,fil_D]=Coord_Gen_Slic3r(filename_gcode);
   elseif fc>0
      [coordinates,fil_D]=Coord_Gen_FullControl(filename_gcode);
   end
end

[m,n]=size(coordenadas); %Dimensiones tabla; m = número de coordenadas
COORD_mm=coordenadas*1000; %Coordenadas en mm
```

El formato de la matriz de coordenadas es el siguiente:

- Las tres primeras columnas están destinadas a las coordenadas X, Y y Z, respectivamente, de cada punto que forma la trayectoria de extrusión.
- En la cuarta columna, el valor “0” indica el comienzo de una nueva trayectoria del filamento y el valor “1” la continuación de esta.
- En la quinta columna aparecen los valores del comando “E” del G-code. Este comando se corresponde con un cuarto eje de movimiento asociado a la longitud o posición de filamento extruido (impulsado por la rueda dentada del extrusor). Por tanto, al definir un movimiento en el G-code, además de las coordenadas X, Y y Z, se puede incorporar el comando “E” seguido de un valor que indica la posición de ese cuarto eje virtual. De este modo, si la E pasa de 2 a 3 en un movimiento, esto significa que se habrá impulsado 1 mm de filamento en ese movimiento.

En la Figura 22 se presenta un ejemplo de la disposición de esta información en la matriz.



	1	2	3	4	5
1	0.0020	0.0020	3.0000e-04	0	0
2	0.0415	0.0020	3.0000e-04	1	2.1294
3	0.0415	0.0065	3.0000e-04	1	2.3729
4	0.0020	0.0065	3.0000e-04	1	4.5023
5	0.0020	0.0020	3.0000e-04	1	4.7459
6	0.0025	0.0025	3.0000e-04	0	0
7	0.0410	0.0025	3.0000e-04	1	2.0777
8	0.0410	0.0060	3.0000e-04	1	2.2695
9	0.0025	0.0060	3.0000e-04	1	4.3472
10	0.0025	0.0025	3.0000e-04	1	4.5390
11	0.0030	0.0030	3.0000e-04	0	0
12	0.0406	0.0030	3.0000e-04	1	2.0259
13	0.0406	0.0056	3.0000e-04	1	2.1660
14	0.0030	0.0056	3.0000e-04	1	4.1920
15	0.0030	0.0030	3.0000e-04	1	4.3321

Figura 22. Disposición datos matriz de coordenadas

Debido a los errores de auto-intersecciones y trayectorias cerradas en la ejecución del modelado en Abaqus/CAE 6.14-1, que impiden el modelado de la pieza, ha sido necesario implementar un bloque para evitarlas. La solución a esta limitación del modelado ha requerido un extenso estudio que se ha desarrollado en el apartado 3.1.3, en el que se detallan las limitaciones encontradas en la aplicación de esta metodología de modelado.

Para comenzar el bloque, se localiza el comienzo de cada tramo, es decir, se buscan los valores “0”, para delimitar el principio y el fin de cada croquis. Además, se añade una última fila con un “0” en su cuarta columna para tomarla como el final del último tramo.

```
% GENERAR CORTES PARA EVITAR AUTO-INTERSECCIONES Y TRAYECTORIAS
CERRADAS
COORD_mm(m+1,4)=0; %Añadir una fila al final, cuya cuarta columna es
un “0”
[row,col]=find(COORD_mm(:,4) == 0); %Localizar el inicio de un tramo
+ la posición de la última fila de ceros
row_size=size(row,1); %Número de tramos + última fila de ceros
```

A continuación, comenzará el código que evita las auto-intersecciones. Para ello, *Abaqus_model* realiza un análisis de las trayectorias y las divide cuando las trayectorias de dos tramos no continuos discurren a poca distancia y, por lo tanto, se produce una intersección. Concretamente, se producirá cuando la distancia sea menor al ancho de extrusión. Del mismo modo, si no es superior al 150% del ancho, se analizará la distancia con los puntos cercanos (véase el apartado 3.1.3.4 para una

explicación más precisa).

El proceso a realizar para evitar la auto-intersección de filamentos se resume en el diagrama de la Figura 23.

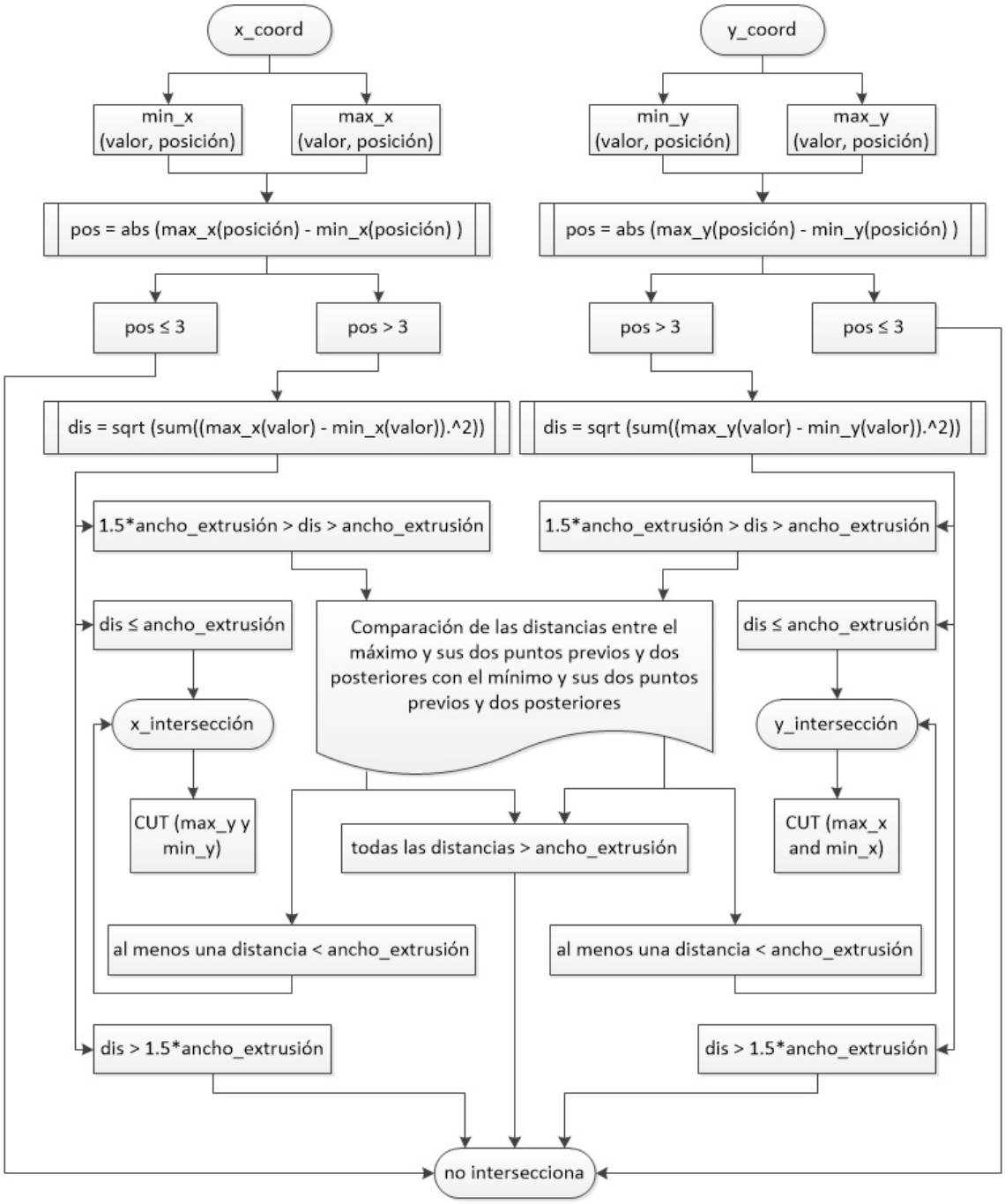


Figura 23. Diagrama proceso para evitar auto-intersecciones de filamentos

Todo este proceso se resuelve mediante código como se muestra a continuación, realizando un análisis tramo a tramo y comparándolo con los valores de ancho de extrusión, calculado mediante los datos de la quinta fila.

```
for j=1:1:row_size-1 %Recorre tramo a tramo

  %Determinación de width
  height=COORD_mm(row(j,1),3); %Determinación altura de capa
  %Sumatoria distancias
  dist=0; %Inicialización de variable dist
  for i=row(j,1):1:row(j+1,1)-2 %Recorre el tramo
    p1=COORD_mm(i,1:2);
    p2=COORD_mm(i+1,1:2);
    dist_n=sqrt(sum((p1-p2).^2)); %Calcula distancia entre puntos
    dist=dist+dist_n; %Sumatoria distancias
  end
  %Diferencia E
  AE=(COORD_mm(row(j+1,1)-1,5)-COORD_mm(row(j,1),5))/1000;
  %Ancho de extrusión
  width=height+(AE*pi*0.875^2/dist-pi*height^2/4)/height;

  %INTERSECCIONES
  %Inicialización de variables
  inter_x=0;
  inter_y=0;
  if row(j+1,1)-row(j,1)>2 %Si hay más de dos puntos en el tramo

    %Se analizan las coordenadas de la matriz entre el inicio de un
    tramo y el fin de este (el inicio del siguiente menos uno)
    datax=COORD_mm(row(j,1):row(j+1,1)-1,1);
    datay=COORD_mm(row(j,1):row(j+1,1)-1,2);

    %Busca los máximos y mínimos de las coordenadas del tramo (valor
    y localización)tanto en X como en Y
    [max_x,locs_max_x] = findpeaks(datax);
    [max_y,locs_max_y] = findpeaks(datay);
    [min_x,locs_min_x] = findpeaks(-datax);
    [min_y,locs_min_y] = findpeaks(-datay);

    %Se determina el número de máximos y mínimos
    size_max_x=size(max_x);
    size_min_x=size(min_x);
    size_max_y=size(max_y);
    size_min_y=size(min_y);

    %Comparación de cada punto máximo con los puntos mínimos en X
    for i=1:1:size_max_x
      p1=COORD_mm(row(j,1)+locs_max_x(i)-1,1:2);

      %Coordenadas del máximo
      for w=1:1:size_min_x
        p2=COORD_mm(row(j,1)+locs_min_x(w)-1,1:2); %Coordenadas del
        mínimo
        dist=sqrt(sum((p1-p2).^2)); %Distancia entre el máximo y el
        mínimo

        %Si dicha distancia es inferior al ancho de extrusión y hay
        más de 3 puntos entre ellos, existe una auto-intersección
        if (dist<width)&&(abs(locs_max_x(i)-locs_min_x(w))>3)
          inter_x=1;

        %Si dicha distancia es superior al ancho de extrusión, pero
        inferior al 150% de este ancho y hay más de 3 puntos entre
```

```
        ellos; se comparan las distancias entre el punto mínimo y
        los dos anteriores y dos posteriores con el punto máximo y
        los dos anteriores y dos posteriores
        elseif (dist<1.5*width)&&(abs(locs_max_x(i)-
        locs_min_x(w))>3)
          Pmin=[COORD_mm(row(j,1)+locs_min_x(w)-3,1:2);
          COORD_mm(row(j,1)+locs_min_x(w)-2,1:2);
          COORD_mm(row(j,1)+locs_min_x(w)-1,1:2);
          COORD_mm(row(j,1)+locs_min_x(w),1:2);
          COORD_mm(row(j,1)+locs_min_x(w)+1,1:2)];

          Pmax=[COORD_mm(row(j,1)+locs_max_x(i)-3,1:2);
          COORD_mm(row(j,1)+locs_max_x(i)-2,1:2);
          COORD_mm(row(j,1)+locs_max_x(i)-1,1:2);
          COORD_mm(row(j,1)+locs_max_x(i),1:2);
          COORD_mm(row(j,1)+locs_max_x(i)+1,1:2)];

          for k=1:1:5
            for w=1:1:5
              dist=sqrt(sum((Pmax(w,1:2)-Pmin(k,1:2)).^2));
              if (dist<width)
                inter_x=1;
              end
            end
          end
        end
      end

      %Comparación de cada punto máximo con los puntos mínimos en Y (se
      repiten los pasos anteriores)
      for i=1:1:size_max_y
       p1=COORD_mm(row(j,1)+locs_max_y(i)-1,1:2);
        for w=1:1:size_min_y
          p2=COORD_mm(row(j,1)+locs_min_y(w)-1,1:2);
          dist=sqrt(sum((p1-p2).^2));
          if (dist<width)&&(abs(locs_max_y(i)-locs_min_y(w))>3)
            inter_y=1;
          elseif (dist<1.5*width)&&(abs(locs_max_y(i)-
          locs_min_y(w))>3)
            Pmin=[COORD_mm(row(j,1)+locs_min_y(w)-3,1:2);
            COORD_mm(row(j,1)+locs_min_y(w)-2,1:2);
            COORD_mm(row(j,1)+locs_min_y(w)-1,1:2);
            COORD_mm(row(j,1)+locs_min_y(w),1:2);
            COORD_mm(row(j,1)+locs_min_y(w)+1,1:2)];

            Pmax=[COORD_mm(row(j,1)+locs_max_y(i)-3,1:2);
            COORD_mm(row(j,1)+locs_max_y(i)-2,1:2);
            COORD_mm(row(j,1)+locs_max_y(i)-1,1:2);
            COORD_mm(row(j,1)+locs_max_y(i),1:2);
            COORD_mm(row(j,1)+locs_max_y(i)+1,1:2)];

            for k=1:1:5
              for w=1:1:5
                dist=sqrt(sum((Pmax(w,1:2)-Pmin(k,1:2)).^2));
                if (dist<width)
                  inter_y=1;
                end
              end
            end
          end
```

```
        end
      end

      %Si hay una intersección en Y, se genera el corte en los máximos
      y mínimos de X (dándole valor -1 a la cuarta columna en la fila
      de esa coordenada); y viceversa, si hay interacción en X, el corte
      se establece en los máximos y mínimos de Y
      if inter_y==1
        if size_max_x(1,1)>0
          for i=1:1:size_max_x(1,1)
            COORD_mm(row(j,1)+locs_max_x(i)-1,4)=-1;
          end
        end
        if size_min_x(1,1)>0
          for i=1:1:size_min_x(1,1)
            COORD_mm(row(j,1)+locs_min_x(i)-1,4)=-1;
          end
        end
      elseif inter_x==1
        if size_max_y(1,1)>0
          for i=1:1:size_max_y(1,1)
            COORD_mm(row(j,1)+locs_max_y(i)-1,4)=-1;
          end
        end
        if size_min_y(1,1)>0
          for i=1:1:size_min_y(1,1)
            COORD_mm(row(j,1)+locs_min_y(i)-1,4)=-1;
          end
        end
      end
    end

    %TRAYECTORIAS CERRADAS
    %Si la distancia entre el primer y el último punto del plano es
    menor a la mitad del ancho de extrusión, se considera que es una
    trayectoria cerrada. Cuando ocurre esto, se separa la última línea
    recta del tramo, asignando a la cuarta columna de la penúltima
    coordenada de la trayectoria el valor -1
    dist_ext=COORD_mm(row(j),1:2)-COORD_mm(row(j+1)-1,1:2);
    if sqrt(sum(dist_ext.^2))<(width/2)
      COORD_mm(row(j+1)-2,4)=-1;
    end
  end
```

A continuación, se volverán a localizar los inicios de cada tramo. En este caso, serán aquellas que presenten un "0" y un "-1", para que sean efectivos los cortes:

```
[row,col]=find(COORD_mm(:,4) <= 0);
row=[row;m+1];
row_size=size(row,1);
```

Una vez obtenidas las coordenadas de la trayectoria del filamento, se comienza a generar el archivo *.py* que será ejecutado en Abaqus/CAE 6.14-1. Para ello, se creará un archivo de escritura con el nombre indicado por el usuario al principio de la subrutina, se define el encabezado del documento y la denominación de los diferentes elementos. En este caso, el modelo se llamará *Model-1*, la pieza *Part-1* y la técnica de

modelado es el barrido (*sweep*).

```
%CREACIÓN ARCHIVO PHYTON
FILE=fopen(filename_py,'w');
head='from part import *\nfrom material import *\nfrom section import
*\nfrom assembly import *\nfrom step import *\nfrom interaction import
*\nfrom load import *\nfrom mesh import *\nfrom optimization import
*\nfrom job import *\nfrom sketch import *\nfrom visualization import
*\nfrom connectorBehavior import *\n';

sheetSize=20;
Model_name='''Model-1''';
Sketch_type='''__sweep__''';
Part_name='''Part-1''';
gridSpacing=0.74;
Profile='''__profile__''';
```

Se realiza un primer barrido independiente, dividido en cabecera, definición del croquis de la trayectoria (que incluye una rotación de ejes para que coincida con la orientación de barrido deseada), creación del croquis del perfil de los filamentos y acción de barrido.

```
%PRIMER BARRIDO
%Croquis
%Escribe la cabecera
fprintf(FILE,head);
fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
sheetSize=%3.3f)\n',Model_name,Sketch_type,sheetSize);

j=1; %Inicialización variable

%Define la fila de la matriz coordenadas en la que empieza la nueva
trayectoria y en la fila en la que acaba, que dependerá de si son
trayectorias que no auto-intersectan y son abiertas, donde el final
sería en la fila anterior al comienzo de la siguiente (cuyo valor en
la cuarta columna es un "0") o si es una de las trayectorias cortadas
por auto-intersección o ser cerrada, en la cual llegaría hasta el punto
donde comienza la siguiente (su valor en la cuarta columna es "-1".
START=row(j,1);
if COORD_mm(row(j+1,1),4)==0
  END=row(j+1,1)-1;
else
  END=row(j+1,1);
end

%Recorre toda la trayectoria, escribiendo en el archivo Python la línea
de programación correspondiente a la creación de una nueva línea del
croquis (definida por dos puntos).
for i=START:1:END-1
  point=COORD_mm(i:i+1,1:2);
  point1=point(1,1:2);
  point2=point(2,1:2);
  fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,%3.3f
  ), point2=(%3.3f, %3.3f))\n',Model_name,Sketch_type,point1,point2);
end

%Rotación de ejes para la correcta orientación del croquis con respecto
al plano y escritura de dichas líneas de programación en el Python.
```

```
var=COORD_mm(2,1:2)-COORD_mm(1,1:2);
tx=-90;
ty=0;
tz=atand(var(1)/var(2));
rot_x=[1 0 0;0 cosd(tx) -sind(tx);0 sind(tx) cosd(tx)];
rot_y=[cosd(ty) 0 sind(ty);0 1 0;-sind(ty) 0 cosd(ty)];
rot_z=[cosd(tz) -sind(tz) 0;sind(tz) cosd(tz) 0;0 0 1];
I=[1 0 0;0 1 0;0 0 1];
ROT=I*rot_x*rot_y*rot_z;

fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
%1.6f, %1.6f, %1.6f, %3.3f, %3.3f,%3.3f))\n',Model_name,Profile,
sheetSize,ROT(1,1:3),ROT(2,1:3),ROT(3,1:3),COORD_mm(1,1:2),
COORD_mm(1,3)-COORD_mm(1,3));

fprintf(FILE,'mdb.models[%s].sketches[%s].ConstructionLine(point1=
(%3.3f,0.0), point2=(%3.3f, 0.0))\n
mdb.models[%s].sketches[%s].ConstructionLine(point1=(0.0,%3.3f),
point2=(0.0, %3.3f))\n',Model_name,Profile,-sheetSize,sheetSize,
Model_name,Profile,-sheetSize,sheetSize);

%Perfil
height=COORD_mm(1,3); %Determinación alto de capa
dist=0; %Inicialización variable
%Determinación de la distancia total recorrida en la trayectoria
for i=START:1:END-1
   p1=COORD_mm(i,1:2);
   p2=COORD_mm(i+1,1:2);
   dist_n=sqrt(sum((p1-p2).^2)); %Cálculo de distancia
   dist=dist+dist_n; %Sumatoria de distancia
end
%Diferencia E
AE=(COORD_mm(END,5)-COORD_mm(START,5))/1000;
%Ancho de extrusión
width=height+(AE*pi*0.875^2/dist-pi*height^2/4)/height;

%Definición de perfil en PY
fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f))\n
mdb.models[%s].sketches[%s].Line(point1=(%3.3f, %3.3f),
point2=(%3.3f, %3.3f))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f,%3.3f), vector=(1.0, 0.0))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f), vector=(-1.0, 0.0))\n',Model_name,
Profile,-(width-height)/2,height,(width-height)/2,height,Model_name,
Profile,-(width-height)/2,0,(width-height)/2,0,Model_name,Profile,
(width-height)/2,0,(width-height)/2,height,Model_name,Profile,
-(width-height)/2,height,-(width-height)/2,0);

%Definición de barrido en PY
fprintf(FILE,'mdb.models[%s].Part(dimensionality=THREE_D,name=%s,
type=DEFORMABLE_BODY)\nmdb.models[%s].parts[%s].BaseSolidSweep(path=
mdb.models[%s].sketches[%s], sketch=mdb.models[%s].sketches[%s])\n',
Model_name,Part_name,Model_name,Part_name,Model_name,Sketch_type,
Model_name,Profile);

fprintf(FILE,'\n');
```

A continuación, se crean los ejes de coordenadas Y y Z y los planos en los que se crearán los croquis. Cada plano corresponderá a una capa de la pieza.

```
%Definición de ejes en PY
fprintf(FILE,'mdb.models[%s].parts[%s].DatumAxisByPrincipalAxis(prin
cipalAxis=YAXIS)\nmdb.models[%s].parts[%s].DatumAxisByPrincipalAxis(
principalAxis=ZAXIS)\n',Model_name,Part_name,Model_name,Part_name);

%Definición de Plano 0 en PY
fprintf(FILE,'mdb.models[%s].parts[%s].DatumPlaneByPrincipalPlane(
offset=%3.3f,principalPlane=XYPLANE)\n',Model_name,Part_name,0);

%Definición de resto de planos en PY
fprintf(FILE,'mdb.models[%s].parts[%s].DatumPlaneByPrincipalPlane(
offset=%3.3f,principalPlane=XYPLANE)\n',Model_name,Part_name,
COORD_mm(row(1),3)); %Para r=1

for r=2:1:row_size-2
   if COORD_mm(row(r),3)==COORD_mm(row(r-1),3)
   else %Si la coordenada Z cambia
     fprintf(FILE,'mdb.models[%s].parts[%s].DatumPlaneByPrincipalPlane
     (offset=%3.3f,principalPlane=XYPLANE)\n',Model_name,Part_name,
     COORD_mm(row(r),3));
   end
end

fprintf(FILE,'\n');
```

Se realizará el resto de barridos, definiendo los mismos aspectos que para el primero, pero, en este caso, se requiere hacer una diferenciación de capas. Concretamente, se debe definir si el barrido que se pretende crear se encuentra o no en el mismo plano que el anterior, es decir, si se produce o no un cambio de capa o coordenada Z.

```
%SIGUIENTES BARRIDOS
datum=4; %Inicialización variable
h_layer=COORD_mm(row(1),3); %Inicialización altura de capa
h_newlayer=h_layer; %Inicialización nueva altura de capa

%Recorre el resto de la matriz, excluyendo el primer barrido
for j=2:1:row_size-2
   %Definición de inicio y fin de la trayectoria, teniendo en cuenta
   si el valor de la cuarta columna es "0" o "-1".
   START=row(j,1);
   if COORD_mm(row(j+1,1),4)==0
     END=row(j+1,1)-1;
   else
     END=row(j+1,1);
   end

   %Si se indica que el principio y el fin es el mismo punto, es que
   ha terminado. Si no, se continúa generando barridos.
```

```
if START==END
  fprintf(FILE,'\n');
else
  h_newlayer=COORD_mm(START,3); %Toma el valor de altura de capa

  %Si la altura de capa es igual a la del primer barrido quiere
  decir que se sitúa en la 1ª capa, en el Plano 0 y, por tanto,
  Datum 4
  if h_newlayer==COORD_mm(row(1),3)
    fprintf(FILE,'mdb.models[%s].ConstrainedSketch(gridSpacing=
    %3.3f,name=%s,sheetSize=%3.3f,transform=mdb.models[%s].
    parts[%s].MakeSketchTransform(sketchPlane=mdb.models[%s].
    parts[%s].datums[%d],sketchPlaneSide=SIDE1,sketchUpEdge=
    mdb.models[%s]. parts[%s].datums[%d],sketchOrientation=LEFT,
    origin=(0.0, 0.0,%3.3f)))\n',Model_name,gridSpacing,
    Sketch_type,sheetSize,Model_name,Part_name,Model_name,
    Part_name,4,Model_name, Part_name,2,0);

  %Si no está en la primera capa, pero se mantiene en la misma capa
  que el barrido anterior
  elseif h_layer==h_newlayer
    cont=0;
    for w=j:-1:1
      if cont<1
        if h_newlayer==COORD_mm(row(w),3)
        else
          fprintf(FILE,'mdb.models[%s].ConstrainedSketch(
          gridSpacing=%3.3f, name=%s,sheetSize=%3.3f,
          transform=mdb.models[%s].parts[%s].
          MakeSketchTransform(sketchPlane=mdb.models[%s].
          parts[%s].datums[%d],sketchPlaneSide=SIDE1,
          sketchUpEdge=mdb.models[%s].parts[%s].datums[%d],
          sketchOrientation= LEFT, origin=(0.0, 0.0,%3.3f)))\n',
          Model_name,gridSpacing,Sketch_type,sheetSize,
          Model_name,Part_name,Model_name,Part_name,datum,
          Model_name,Part_name,2, COORD_mm(row(w),3));

          cont=1;
        end
      end
    end

  %Si existe un cambio de capa
  else
    datum=datum+1;
    fprintf(FILE,'mdb.models[%s].ConstrainedSketch(gridSpacing=%3
    .3f, name=%s,sheetSize=%3.3f,transform=mdb.models[%s].
    parts[%s].MakeSketchTransform(sketchPlane=mdb.models[%s].
    parts[%s].datums[%d],sketchPlaneSide=SIDE1,sketchUpEdge=
    mdb.models[%s]. parts[%s].datums[%d],sketchOrientation=LEFT,
    origin=(0.0, 0.0,%3.3f)))\n',Model_name,gridSpacing,
    Sketch_type,sheetSize,Model_name,Part_name,Model_name,
    Part_name,datum,Model_name, Part_name,2,h_layer);

  end

  %Línea de programación
  fprintf(FILE,'mdb.models[%s].parts[%s].
  projectReferencesOntoSketch(filter=COPLANAR_EDGES,sketch=
  mdb.models[%s].sketches[%s])\n',Model_name,Part_name,Model_name,
  Sketch_type);
```

```
%Definición de croquis
for i=START:1:END-1
   point=COORD_mm(i:i+1,1:2);
   point1=point(1,1:2);
   point2=point(2,1:2);
   fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,
   %3.3f), point2=(%3.3f, %3.3f))\n',Model_name,Sketch_type,
   point1,point2);
end

%Rotación de ejes
var=COORD_mm(row(j)+1,1:2)-COORD_mm(row(j),1:2);
tx=-90;
ty=0;
tz=atand(var(1)/var(2));
rot_x=[1 0 0;0 cosd(tx) -sind(tx);0 sind(tx) cosd(tx)];
rot_y=[cosd(ty) 0 sind(ty);0 1 0;-sind(ty) 0 cosd(ty)];
rot_z=[cosd(tz) -sind(tz) 0;sind(tz) cosd(tz) 0;0 0 1];
I=[1 0 0;0 1 0;0 0 1];
ROT=I*rot_x*rot_y*rot_z;

if h_newlayer==COORD_mm(row(1),3) %Capa 1
   fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
   sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
   %1.6f, %1.6f, %1.6f, %1.6f, %3.3f,%3.3f,%3.3f))\n',
   Model_name,Profile,sheetSize,ROT(1,1:3),ROT(2,1:3),
   ROT(3,1:3),COORD_mm(START,1:2),0);

elseif h_layer==h_newlayer %Misma capa
   cont=0;
   for w=j:-1:1
     if cont<1
       if h_newlayer==COORD_mm(row(w),3)
       else
          fprintf(FILE,'mdb.models[%s].ConstrainedSketch(
          name=%s, sheetSize=%3.3f,transform=(%1.6f, %1.6f,
          %1.6f, %1.6f, %1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
          %3.3f,%3.3f,%3.3f))\n',Model_name,Profile,sheetSize,
          ROT(1,1:3),ROT(2,1:3),ROT(3,1:3),COORD_mm(START,1:2),
          COORD_mm(row(w),3));
          cont=1;
       end
     end
   end

else %Cambio de capa
   fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
   sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
   %1.6f, %1.6f, %1.6f, %1.6f, %3.3f,%3.3f,%3.3f))\n',
   Model_name,Profile,sheetSize,ROT(1,1:3),ROT(2,1:3),
   ROT(3,1:3),COORD_mm(START,1:2),h_layer);
end

fprintf(FILE,'mdb.models[%s].sketches[%s].ConstructionLine(
point1=(%3.3f,0.0),point2=(%3.3f,0.0))\nmdb.models[%s].
sketches[%s]. ConstructionLine(point1=(0.0,%3.3f), point2=(0.0,
%3.3f))\nmdb.models[%s].parts[%s].projectReferencesOntoSketch(
filter=COPLANAR_EDGES,sketch=mdb.models[%s].sketches[%s])\n',
Model_name,Profile,-sheetSize,sheetSize,Model_name,Profile,
-sheetSize,sheetSize,Model_name,Part_name,Model_name,Profile);
```

```
%Perfil
%Determinación altura de capa
if h_newlayer==COORD_mm(1,3)
   height=h_newlayer;
elseif h_newlayer==h_layer
   height=height;
else
   height=h_newlayer-h_layer;
end

%Determinación de ancho de extrusión
dist=0;
for i=START:1:END-1
   p1=COORD_mm(i,1:2);
   p2=COORD_mm(i+1,1:2);
   dist_n=sqrt(sum((p1-p2).^2)); %Cálculo distancia
   dist=dist+dist_n; %Sumatoria distancias
end
%Diferencia E
AE=(COORD_mm(END,5)-COORD_mm(START,5))/1000;
%Ancho de extrusión
width=height+(AE*pi*0.875^2/dist-pi*height^2/4)/height;

%Línea de programación para definir perfil
fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f))\nmdb.models[%s].sketches[%s].
Line(point1=(%3.3f, %3.3f), point2=(%3.3f, %3.3f))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f,%3.3f), vector=(1.0, 0.0))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f), vector=(-1.0,
0.0))\n',Model_name, Profile,-(width-height)/2,height,(width-
height)/2,height, Model_name,Profile,-(width-height)/2,0,(width-
height)/2,0, Model_name,Profile,(width-height)/2,0,(width-
height)/2,height, Model_name,Profile,-(width-height)/2,height,-
(width-height)/2,0);

%Definición barrido
x2=COORD_mm(START+1,1);
x1=COORD_mm(START,1);
y2=COORD_mm(START+1,2);
y1=COORD_mm(START,2);
x=((-y1)/(y2-y1))*(x2-x1)+x1;

if ((x2>x1)&&(y2==y1))||((x2==x1)&&(y2<y1))||
((x2>x1)&&(y2<y1))||((x>0)&&(x2<x1)&&(y2<y1))||((x<0)&&(x2>x1)&&
(y2>y1))

   fprintf(FILE,'mdb.models[%s].parts[%s].SolidSweep(path=
   mdb.models[%s].sketches[%s],pathOrientation=LEFT,pathPlane=
   mdb.models[%s].parts[%s].datums[%d],pathUpEdge=
   mdb.models[%s].parts[%s].datums[%d],profile=mdb.models[%s].
   sketches[%s],sketchOrientation=RIGHT,sketchUpEdge=
   mdb.models[%s].parts[%s].datums[%d])\n',Model_name,Part_name,
   Model_name,Sketch_type,Model_name,Part_name,datum,Model_name,
   Part_name,2,Model_name, Profile,Model_name,Part_name,3);

else
   fprintf(FILE,'mdb.models[%s].parts[%s].SolidSweep(path=
   mdb.models[%s].sketches[%s],pathOrientation=LEFT,pathPlane=
```

```
        mdb.models[%s].parts[%s].datums[%d],pathUpEdge=
        mdb.models[%s]. parts[%s].datums[%d],profile=mdb.models[%s].
        sketches[%s],sketchOrientation=LEFT,sketchUpEdge=
        mdb.models[%s].parts[%s].datums[%d])\n',Model_name,Part_name,
        Model_name,Sketch_type,Model_name,Part_name,datum,Model_name,
        Part_name,2,Model_name, Profile,Model_name,Part_name,3);
    end
  end

  h_layer=h_newlayer; %La altura de capa toma el valor de la última
  capa en la que se ha generado el barrido
  fprintf(FILE,'\n');
end
```

Por último, se cierra el archivo y se da por terminada la creación del Python.

```
fclose(FILE);
```

3.2.4. Coord_Gen_Slic3r

Tras conocer el funcionamiento de la subrutina principal, Abaqus_model, en este apartado se explicará la función utilizada para extraer las coordenadas de un archivo G-code generado en Slic3r 1.3.0. Es preciso recalcar la importancia de que este archivo sea generado con comentarios, ya que el funcionamiento de la función depende de ello.

La función *Coord_Gen_Slic3r* requiere una entrada, el nombre del archivo de código G, y devuelve dos salidas, la matriz de coordenadas y el diámetro del filamento. Su código completo se encuentra en el Anexo B.

Dicha función comienza con la lectura del G-code. Primero, se abre el archivo para determinar la dimensión máxima de la matriz de coordenadas. En la segunda apertura, se crea la matriz, tomando las cinco primeras columnas. Por último, se extraen los comentarios para tomar datos de impresión.

```
function [coordinates,fil_D]=Coord_Gen_Slic3r(filename_gcode)
  %INTRODUCCIÓN ARCHIVO
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  CI=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
  %s %s %s %s',[inf,20]); %Lectura de todas las filas y 20 columnas
  kc=size(CI{2},1); %Determinación de la dimensión máxima de filas de
  la matriz (kc)
  fclose(ARCH); %Cierre de archivo
  ARCH=fopen(filename_gcode,'r'); %Nueva apertura
  C=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
  %s %s %s',[kc,20]); %Lectura de 'kc' filas y 20 columnas
  c=[C{1} C{2} C{3} C{4} C{5}]; %Creación de matriz de tipo string con
  todo el archivo
  fclose(ARCH); %Cierre del archivo
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  COM=textscan(ARCH,'%s %s','Delimiter',';');
  comments=[COM{2}];
  fclose(ARCH); %Cierre de archivo
```

El siguiente paso es la búsqueda del valor del diámetro de filamento a partir de los comentarios del archivo.

```
%Diámetro del filamento
[fil,col]=size(comments);

for i=1:1:fil
   dia=strfind(comments{i},'filament_diameter');
   if isempty(dia)==0
      fil_dia=strrep(comments{i},'filament_diameter = ','');
      fil_D=str2num(fil_dia);
   end
end
```

A continuación, se realiza una limpieza de la matriz de coordenadas, eliminando los comentarios y comandos no necesarios.

```
%ELIMINACIÓN DE COMENTARIOS
k=size(c,1); %Número de columnas de la matriz 'c'

for i=1:1:k
   for j=1:1:5
      f=strfind(c{i,j},';'); %Busca en cada celda un ';'
      ff=strfind(c{i,j},'='); %Busca en cada celda un '='
      if ff>0
         c{i,j}=' '; %Si lo encuentra borra esa celda
      elseif f>0
         for m=j:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
      end
   end
end

%ELIMINACIÓN DE COMANDOS 'M', 'S', 'G28', 'G' y 'F'
for i=1:1:k
   for j=1:1:5
      M=strfind(c{i,j},'M'); %Busca el comando 'M' celda por celda
      H=strfind(c{i,j},'G00'); %Busca el comando 'G00' celda por celda
      V=strfind(c{i,j},'G28'); %Busca el comando 'G28' celda por celda
      G=strfind(c{i,j},'G'); %Busca el comando 'G' celda por celda
      S=strfind(c{i,j},'S'); %Busca el comando 'S' celda por celda
      F=strfind(c{i,j},'F'); %Busca el comando 'F' celda por celda
      if M==1
         c{i,j}=''; %Borra la celda donde se encuentra el comando 'M'
      elseif H==1
         c{i,j}=''; %Borra G0
      elseif V==1 %Borra G28 y todo lo que sigue al comando
         c{i,j}='';
         c{i,j+1}='';
         c{i,j+2}='';
      elseif G==1
         c{i,j}=''; %Borra la celda donde se encuentra el comando 'G'
      elseif S==1
         c{i,j}=''; %Borra la celda donde se encuentra el comando 'S'
      elseif F==1
         c{i,j}=''; %Borra la celda donde se encuentra el comando 'F'
      end
```

```
   end
end
```

Con el resto de información, se genera el vector de posiciones, dividido en cinco columnas: las tres primeras para definir las coordenadas XYZ, la cuarta para definir el comienzo y continuación de las trayectorias (con "0" y "1") y la quinta, para especificar la cantidad de filamento extruido (comando E).

En primer lugar, se rellenan de la primera a la tercera y la quinta, tomando los valores de X, Y, Z y E en cada fila.

```
%VECTOR POSICIÓN + E
pos=zeros(1,5); %Inicialización matriz 'pos'

for i=1:1:k
   for j=1:1:5
      X=strfind(c{i,j},'X'); %Busca 'X' celda por celda
      Y=strfind(c{i,j},'Y'); %Busca 'Y' celda por celda
      Z=strfind(c{i,j},'Z'); %Busca 'Z' celda por celda
      E=strfind(c{i,j},'E'); %Busca el comando 'E' celda por celda
      if X==1
         cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
         únicamente los números
         pos(i,1)=str2num(cp{i,1}); %Pasa los números al vector posición
         (pos)
      elseif Y==1
         cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
         únicamente los números
         pos(i,2)=str2num(cp{i,2}); %Pasa los números al vector posición
         (pos)
      elseif Z==1
         cp{i,3}=strrep(c{i,j},'Z',''); %Borra la 'Z' para dejar
         únicamente los números
         pos(i,3)=str2num(cp{i,3}); %Pasa los números al vector posición
         (pos)
      elseif E==1
         cp{i,4}=strrep(c{i,j},'E',''); %Borra la 'E' para dejar
         únicamente los números
         pos(i,5)=str2num(cp{i,4}); %Pasa los números al vector posición
         (pos)
      end
   end
end
```

A continuación, se completa la cuarta columna, considerando los comentarios. En las trayectorias que no sean necesarias para generar el modelo, como la impresión de la falda (skirt), se asignará a la cuarta columna el valor "2", para luego ser omitido. Los principios de trayectoria, cuyo valor será "0", se definen en aquellos puntos en los que se especifica que se mueve al primer punto del relleno o del perímetro.

```
[k,y]=size(pos);
for i=1:1:k
   inwards=strfind(comments(i),'move inwards before travel');
   skirt=strfind(comments(i),'skirt');
```

```
   if isequal(skirt,{[1]})
     pos(i,4)=2;
   elseif isequal(skirt,{[15]})
     pos(i,4)=2;
   elseif isequal(inwards,{[1]})
     pos(i,4)=2;
     pos(i+1,3)=pos(i,3);
   elseif isequal(comments(i),{'lift nozzle'})
     pos(i,4)=2;
   elseif isequal(comments(i),{'move to first infill point'})
     pos(i,4)=0;
   elseif isequal(comments(i),{'move to first infill (bridge) point'})
     pos(i,4)=0;
   elseif isequal(comments(i),{'move to first perimeter point'})
     pos(i,4)=0;
   else
     pos(i,4)=1;
   end
end
```

Cuando se indica que hay un cambio de capa (move to next layer), se requiere la búsqueda de la Z correspondiente. Por otro lado, cuando se resetea la extrusión (G92 E0), hará falta hacer cambios en la quinta fila, volviendo a tomar el valor de E por defecto al comenzar a extruir. En Slic3r 1.3.0, el valor predeterminado de E al empezar con la nueva extrusión (unretract extruder) es 2 mm.

```
for i=1:1:k
   layer=strfind(comments(i),'move to next layer');
   if isequal(layer,{[1]})
     pos(i,4)=2;
     fir=find(pos(i:k,4)==0);
     pos(i+fir(1)-1,3)=pos(i,3);
   elseif isequal(comments(i),{'reset extrusion distance'})
     pos(i+1,5)=2;
   end
end
```

Una vez se han asignado todas las columnas, se realiza una limpieza, manteniendo solamente aquellas filas en las que existan coordenadas y cuyos valores en la cuarta columna sean “0” ó “1”. De esta forma se eliminan todas las filas del código que no aportan información sobre la trayectoria de extrusión, a las que se le ha dado valor “2” en la cuarta fila.

```
fila=1;
pos_3=pos(:,1:3);
com=sum(pos_3,2);
[k,y]=size(com);
posf=zeros(1,5);
for i=1:1:k
   if com(i,1)>0 && pos(i,4)<2
     posf(fila,1:5)=pos(i,1:5);
     fila=fila+1;
   end
end
```

Tras la limpieza, ya que en el código G las coordenadas que no varían de un punto a otro, no se repiten, es preciso rellenar aquellas coordenadas que no tengan valor asignado con el que poseía anteriormente.

```
%Si en la fila ha encontrado una X, una Y o/y una Z, con lo cual existe
%una posición, pero alguna de las coordenadas no aparece, toma el valor
%anterior (no es cero)
[filf,colf]=size(posf);

for i=2:1:filf
  if posf(i,1)==0 %Si no se especifica el valor de X
    posf(i,1)=posf(i-1,1); %Toma el valor X de la posición anterior
  end
  if posf(i,2)==0 %Si no se especifica el valor de Y
    posf(i,2)=posf(i-1,2); %Toma el valor Y de la posición anterior
  end
  if posf(i,3)==0 %Si no se especifica el valor de Z
    posf(i,3)=posf(i-1,3); %Toma el valor Z de la posición anterior
  end
  if posf(i,5)==0 %Si no se especifica el valor de E
    posf(i,5)=posf(i-1,5); %Toma el valor E de la posición anterior
  end
end
```

El último paso es la preparación de la matriz para ser enviada a la subrutina principal. Para ello, se trasladan las coordenadas de manera que la esquina inferior izquierda de la pieza esté situada en la coordenada (2,2,0) y se pasan las unidades a metros. Este cambio de unidades se realiza con el objetivo de acondicionar la matriz para otras aplicaciones, aunque en Abaqus_model se volverán a pasar a milímetros.

```
  coordenadas=posf(1:filf,1:5);
  [filc,colc]=size(coordenadas);
  minx=coordenadas(1,1);
  miny=coordenadas(1,2);
  for i=1:1:filc
    if coordenadas(i,1)<minx
      minx=coordenadas(i,1);
    end
    if coordenadas(i,2)<miny
      miny=coordenadas(i,2);
    end
  end
  minx=minx-2;
  miny=miny-2;
  for i=1:1:filc
    coordenadas(i,1)=(coordenadas(i,1)-minx)/1000;
    coordenadas(i,2)=(coordenadas(i,2)-miny)/1000;
    coordenadas(i,3)=coordenadas(i,3)/1000;
  end
end %Final de función
```

3.2.5. Coord_Gen_Simplify

En caso de haber generado el archivo G-code con el laminador Simplify 3D, la función que extraerá las coordenadas será *Coord_Gen_Simplify*. El código completo

de la función se encuentra en el Anexo C. La diferencia principal con Slic3r es el tipo de comentarios que presenta.

La función recibe como entrada el nombre del archivo de código G y devuelve la matriz de coordenadas y el valor del diámetro de filamento.

En primer lugar, lee el archivo G-code generado en Simplify 3D y guarda el código en una matriz tipo cadena.

```
function [coordinates,fil_D]=Coord_Gen_Simplify(filename_gcode)

  %INTRODUCCIÓN ARCHIVO
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  CI=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[inf,20]); %Lectura de todas las filas y 20 columnas
  kc=size(CI{2},1); %Determinación de las dimensión máxima de filas de la matriz (kc)
  fclose(ARCH); %Cierre de archivo
  ARCH=fopen(filename_gcode,'r'); %Nueva apertura
  C=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[kc,20]); %Lectura de 'kc' filas y 20 columnas
  c=[C{1} C{2} C{3} C{4} C{5}]; %Creación de matriz de tipo cadena con todo el archivo
  fclose(ARCH); %Cierre del archivo
```

En la matriz tipo cadena de todo el archivo, se busca el valor del diámetro de filamento.

```
%Diámetro de filamento
[fil,col]=size(c);

for i=1:1:fil
  dia=strfind(c{i,2},'filamentDiameters');
  if isempty(dia)==0
    fil_1=strfind(c{i,2},'1.75');
    fil_3=strfind(c{i,2},'3');
    if isempty(fil_1)==0
      fil_D=1.75;
    elseif isempty(fil_3)==0
      fil_D=3;
    end
  end
end
```

A continuación, se sustituyen los comentarios que indican la generación de una trayectoria de relleno (infill/fill), perímetro (perimeter) o capa sólida (solid) por la palabra clave que indicara que es parte de la pieza (part).

```
%MODIFICIÓN DE COMENTARIOS
k=size(c,1); %Número de columnas de la matriz 'c'
cont=0;
for i=1:1:k
  feature=strfind(c(i,2),'feature'); %Busca comentarios 'feature'
  if cont==0&&isequal(feature,{[1]})
    inicio=i;
```

```
      cont=cont+1;
   end
end
for i=inicio:1:k
   for j=1:1:5
      perimeter=strfind(c{i,j},'perimeter');     %Busca     comentarios
      'perimeter'
      infill=strfind(c{i,j},'infill'); %Busca comentarios 'infill'
      solid=strfind(c{i,j},'solid'); %Busca comentarios 'solid layer'
      fill=strfind(c{i,j},'fill'); %Busca comentarios 'fill'
      support=strfind(c{i,j},'support'); %Busca comentarios 'support'
      if perimeter>0
         for m=1:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
         c{i,2}='part';
      elseif infill>0
         for m=1:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
         c{i,2}='part';
      elseif solid>0
         for m=1:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
         c{i,2}='part';
      elseif fill>0
         for m=1:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
         c{i,2}='part';
      elseif support>0
         for m=1:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
         c{i,2}='support';
      end
   end
end
```

Tras la modificación, se eliminan los comentarios. Todos ellos se presentan después de ";".

```
%ELIMINACIÓN DE COMENTARIOS
for i=1:1:k
   for j=1:1:5
      comment=strfind(c{i,j},';'); %Busca en cada celda un ';'
      if comment>0
         for m=j:1:5
            c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
            siguientes de la fila
         end
      end
   end
end
```

Además, se eliminan los comandos que no serán necesarios para la obtención de la matriz coordenadas.

```
%ELIMINACIÓN DE COMANDOS 'M', 'S', 'G28', 'G' y 'F'
for i=1:1:k
  for j=1:1:5
    M=strfind(c{i,j},'M'); %Busca el comando 'M' celda por celda
    H=strfind(c{i,j},'G00'); %Busca el comando 'G00' celda por celda
    V=strfind(c{i,j},'G28'); %Busca el comando 'G28' celda por celda
    G=strfind(c{i,j},'G'); %Busca el comando 'G' celda por celda
    S=strfind(c{i,j},'S'); %Busca el comando 'S' celda por celda
    F=strfind(c{i,j},'F'); %Busca el comando 'S' celda por celda
    if M==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'M'
    elseif H==1
      c{i,j}=''; %Borra G0
    elseif V==1 %Borra G28 y todo lo que sigue al comando
      c{i,j}='';
      c{i,j+1}='';
      c{i,j+2}='';
    elseif G==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'G'
    elseif S==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'S'
    elseif F==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'F'
    end
  end
end
```

Con la información sobrante, se crea la matriz de coordenadas. Concretamente, se rellenan las tres primeras columnas con los valores de X, Y y Z y la quinta con los valores de E. A su vez, en las filas que contienen el comando E, por tanto, representa una trayectoria de extrusión, se asigna el valor "1" a la cuarta fila, y si no existe este comando, un "0".

```
%VECTOR POSICIÓN
pos=NaN(k,5); %Inicialización matriz pos
for i=1:1:k
  %Busca el comando 'E' en la fila
  E3=strfind(c{i,3},'E');
  E4=strfind(c{i,4},'E');
  for j=1:1:5
    X=strfind(c{i,j},'X'); %Busca 'X' celda por celda
    Y=strfind(c{i,j},'Y'); %Busca 'Y' celda por celda
    Z=strfind(c{i,j},'Z'); %Busca 'Z' celda por celda
    E=strfind(c{i,j},'E'); %Busca 'E' celda por celda
    if E3==1
      if X==1
        cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
        únicamente los números
        pos(i,1)=str2num(cp{i,1}); %Pasa los números al vector
        posición (pos)
        pos(i,4)=1;
      elseif Y==1
        cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
        únicamente los números
```

```
            pos(i,2)=str2num(cp{i,2});  %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
          elseif E==1
            cp{i,3}=strrep(c{i,j},'E',''); %Borra la 'E' para dejar
            únicamente los números
            pos(i,5)=str2num(cp{i,3});  %Pasa los números al vector
            posición (pos)
          end
        elseif E4==1
          if X==1
            cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
            únicamente los números
            pos(i,1)=str2num(cp{i,1});  %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
          elseif Y==1
            cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
            únicamente los números
            pos(i,2)=str2num(cp{i,2});  %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
          elseif E==1
            cp{i,4}=strrep(c{i,j},'E',''); %Borra la 'E' para dejar
            únicamente los números
            pos(i,5)=str2num(cp{i,4});  %Pasa los números al vector
            posición (pos)
          end
        else
          if X==1
            cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
            únicamente los números
            pos(i,1)=str2num(cp{i,1});  %Pasa los números al vector
            posición (pos)
            pos(i,4)=0;
            pos(i,5)=0;
          elseif Y==1
            cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
            únicamente los números
            pos(i,2)=str2num(cp{i,2});  %Pasa los números al vector
            posición (pos)
            pos(i,4)=0;
            pos(i,5)=0;
          end
        end
        if Z==1
          cp{i,3}=strrep(c{i,j},'Z',''); %Borra la 'Z' para dejar
          únicamente los números
          pos(i,3)=str2num(cp{i,3}); %Pasa los números al vector posición
          (pos)
        end
      end
    end
```

Para terminar de completar la cuarta fila, se le asignará el valor “2” a las filas que cuya trayectoria no sea de interés para el modelado.

```
cont=0;
fil_support=1;
fil_part=1;
```

```
for i=1:1:k
   part=strfind(c{i,2},'part');
   support=strfind(c{i,2},'support');
   if part==1
      if cont==0
         pos(1:i,4)=2;
         cont=cont+1;
      else
         fil_part=i;
      end
   elseif support==1
      if cont==0
         pos(1:i,4)=2;
         cont=cont+1;
      else
         fil_support=i;
      end
   end
   if fil_support<fil_part&&fil_support>1
      pos(fil_support:fil_part,4)=2;
      fil_support=1;
      fil_part=1;
   end
end
```

En las líneas de programación no aparecen las coordenadas completas (XYZ) ni el valor de E al comenzar la trayectoria, así que se tomarán los valores anteriormente asignados.

```
%Si en la fila ha encontrado una X, una Y o/y una Z, con lo cual existe
%una posición, pero alguna de las coordenadas no aparece, toma el valor
%anterior (no es cero)
for i=2:1:k
   if isnan(pos(i,1))==1 %Si no se especifica el valor de X
      pos(i,1)=pos(i-1,1); %Toma el valor X de la posición anterior
   end
   if isnan(pos(i,2))==1 %Si no se especifica el valor de Y
      pos(i,2)=pos(i-1,2); %Toma el valor Y de la posición anterior
   end
   if isnan(pos(i,3))==1 %Si no se especifica el valor de Z
      pos(i,3)=pos(i-1,3); %Toma el valor Z de la posición anterior
   end
   if isnan(pos(i,5))==1 %Si no se especifica el valor de E
      pos(i,5)=pos(i-1,5); %Toma el valor E de la posición anterior
   end
end
```

Se realiza una limpieza del vector posición, descartando las filas que tienen valor "2" en su quinta columna.

```
fila=1;
posf=zeros(1,5);
for i=1:1:k
   if isnan(pos(i,4))==0 && pos(i,4)<2
      posf(fila,1:5)=pos(i,1:5);
      fila=fila+1;
   end
end
```

Por último, se traslada el punto inferior izquierdo de la pieza a las coordenadas (2,2,0), se pasan las coordenadas a metros y se finaliza la función.

```
   [filf,colf]=size(posf);
   inicio=find(posf(:,5)==0);
   coordenadas=posf(inicio(1):filf,1:5);
   [filc,colc]=size(coordenadas);
   minx=coordenadas(1,1);
   miny=coordenadas(1,2);
   for i=1:1:filc
      if coordenadas(i,1)<minx
         minx=coordenadas(i,1);
      end
      if coordenadas(i,2)<miny
         miny=coordenadas(i,2);
      end
   end
   minx=minx-2;
   miny=miny-2;
   for i=1:1:filc
      coordenadas(i,1)=(coordenadas(i,1)-minx)/1000;
      coordenadas(i,2)=(coordenadas(i,2)-miny)/1000;
      coordenadas(i,3)=coordenadas(i,3)/1000;
   end
end %Fin de la función
```

3.2.6. Coord_Gen_FullControl

La función Coord_Gen_FullControl se ha creado para la interpretación de las trayectorias generadas por el software FullControl G-code Designer a partir del código G de salida. Su código completo se presenta en el Anexo D.

En este caso, no se dispone de comentarios de apoyo para la búsqueda de trayectorias, sino que será necesario apoyarse del comando G0 para saber la línea en la que comienza la trayectoria del filamento extruido. Asimismo, se debe tener en cuenta que el valor "E" en este código es incremental (M83), a diferencia de los anteriores, en los que la extrusión de material se realizaba en valores absolutos (M82).

La variable de entrada de la función es el nombre del archivo y las salidas son la matriz de coordenadas y el diámetro de filamento. Este último será introducido manualmente por el usuario, ya que no se define en el G-code. Tras requerir este dato, se lee el archivo para comenzar a procesar los datos.

```
function [coordinates,fil_D]=Coord_Gen_FullControl(filename_gcode)
   % Entrada de diámetro de filament
   fil_D=str2num(input('Enter the FeedstockFilamentDiameter defined
   in FullControl (only number in mm, e.g. 1.75): ','s'));

   %INTRODUCCIÓN ARCHIVO
   ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
   CI=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
   %s %s %s %s',[inf,20]); %Lectura de todas las filas y 20 columnas
```

```
kc=size(CI{2},1); %Determinación de la dimensión máxima de filas de
la matriz (kc)
fclose(ARCH); %Cierre de archivo
ARCH=fopen(filename_gcode,'r'); %Nueva apertura
C=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
%s %s %s',[kc,20]); %Lectura de 'kc' filas y 20 columnas
c=[C{1} C{2} C{3} C{4} C{5}]; %Creación de matriz de tipo string con
todo el archivo
fclose(ARCH); %Cierre del archivo
ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
COM=textscan(ARCH,'%s %s','Delimiter',';');
comments=[COM{2}];
fclose(ARCH); %Cierre de archivo
```

A continuación, se eliminan los comentarios, los cuales se presentan después de ";", y los comandos no necesarios.

```
%ELIMINACIÓN DE COMENTARIOS
k=size(c,1); %Número de columnas de la matriz 'c'
for i=1:1:k
  for j=1:1:5
    f=strfind(c{i,j},';'); %Busca en cada celda un ';'
    if f>0
      for m=j:1:5
        c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
        siguientes de la fila
      end
    end
  end
end

%ELIMINACIÓN DE COMANDOS 'M', 'S', 'G28', 'G' y 'F'
for i=1:1:k
  for j=1:1:5
    M=strfind(c{i,j},'M'); %Busca el comando 'M' celda por celda
    V=strfind(c{i,j},'G28'); %Busca el comando 'G28' celda por celda
    S=strfind(c{i,j},'S'); %Busca el comando 'S' celda por celda
    F=strfind(c{i,j},'F'); %Busca el comando 'F' celda por celda
    if M==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'M'
    elseif V==1 %Borra G28 y todo lo que sigue al comando
      c{i,j}='';
      c{i,j+1}='';
      c{i,j+2}='';
    elseif S==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'S'
    elseif F==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'F'
    end
  end
end
```

En el siguiente paso, se rellena la matriz de coordenadas. Los valores que acompañan a las coordenadas XYZ, se sitúan en la primera, segunda y tercera fila, respectivamente. Los valores de cantidad de material extruido (E) se añaden a la quinta fila. Por último, en las líneas en las que aparezca un G0, se añadirá un "0" en la cuarta columna, y en aquellas que aparezca G1, un "1". Esto indicará el comienzo (0) y la

continuación (1) de la trayectoria del filamento.

```
%VECTOR POSICIÓN + E
pos=zeros(k,5); %Inicialización matriz 'pos'
for i=1:1:k
   for j=1:1:5
      X=strfind(c{i,j},'X'); %Busca 'X' celda por celda
      Y=strfind(c{i,j},'Y'); %Busca 'Y' celda por celda
      Z=strfind(c{i,j},'Z'); %Busca 'Z' celda por celda
      E=strfind(c{i,j},'E'); %Busca 'E' celda por celda
      H=strfind(c{i,j},'G0'); %Busca 'G0' celda por celda
      G=strfind(c{i,j},'G1'); %Busca 'G1' celda por celda
      if X==1
         cp{i,1}=strrep(c{i,j},'X',''); %Deletes 'X' para solamente
         conservar los números
         pos(i,1)=str2double(cp{i,1}); %Trasfiere los números al vector
         de posición (pos)
      elseif Y==1
         cp{i,2}=strrep(c{i,j},'Y',''); %Elimina 'Y' para solamente
         conservar los números
         pos(i,2)=str2double(cp{i,2}); %Trasfiere los números al vector
         de posición (pos)
      elseif Z==1
         cp{i,3}=strrep(c{i,j},'Z',''); %Elimina 'Z' para solamente
         conservar los números
         pos(i,3)=str2double(cp{i,3}); %Trasfiere los números al vector
         de posición (pos)
      elseif E==1
         cp{i,5}=strrep(c{i,j},'E',''); %Elimina 'E' para solamente
         conservar los números
         pos(i,5)=str2double(cp{i,5}); %Trasfiere los números al vector
         de posición (pos)
      elseif H==1
         cp{i,4}=strrep(c{i,j},'G',''); %Elimina 'G' para solamente
         conservar los números
         pos(i,4)=str2double(cp{i,4}); %Trasfiere el número al vector de
         posición (pos)
         ho=i;
      elseif G==1
         cp{i,4}=strrep(c{i,j},'G',''); %Elimina 'G' para solamente
         conservar los números
         pos(i,4)=str2double(cp{i,4}); %Trasfiere el número al vector de
         posición (pos)
      end
   end
end
```

A continuación, se genera una nueva matriz eliminando aquellas filas en las que se introduce y concluye el G-code o no se presentan coordenadas. Además, los valores de E se pasan de valores incrementales a absolutos.

```
%Inicio y fin de G-code (sin comentarios)
[k,y]=size(pos);
for i=1:1:k
   if isequal(comments(i),{'END OF THE START GCODE'})
      for j=i:-1:1
         pos(j,4)=2; %Asigna un 2 a todas las filas anteriores
      end
   elseif isequal(comments(i),{'START OF THE END GCODE'})
```

```
      for j=i:1:k
        pos(j,4)=2; %Asigna un 2 a todas las filas posteriores
      end
    end
end

fila=1;
pos_3=pos(:,1:3); %Toma parte de matriz de coordenadas XYZ
com=sum(pos_3,2); %Suma coordenadas XYZ de cada fila
[k,y]=size(com);
posf=zeros(1,5); %Nuevo vector posición
%Si la sumatoria de coordenadas es superior a 0 y los valores de la
cuarta fila son 0 ó 1, graba la línea en el nuevo vector
for i=1:1:k
   if com(i,1)>0 && pos(i,4)<2
     posf(fila,1:5)=pos(i,1:5);
     fila=fila+1;
   end
end

%Sumatoria de E (valores absolutos)
[filf,colf]=size(posf);
for i=2:1:filf
   posf(i,5)=posf(i,5)+posf(i-1,5);
end
```

Para terminar de acondicionar la matriz, se completan los valores no especificados, debido a que en el G-code se evita la redundancia en coordenadas que se mantienen constantes, tomando el valor que tenía anteriormente. Adicionalmente, se traslada la esquina inferior izquierda de la pieza a la coordenada (2,2,0) y se pasan las unidades a metros. Con esto finaliza la función.

```
   %Si en la fila ha encontrado una X, una Y o/y una Z, con lo cual
   %existe una posición, pero alguna de las coordenadas no aparece,
   %toma el valor anterior (no es cero)
   for i=2:1:filf
     if posf(i,1)==0 %Si no se especifica el valor de X
       posf(i,1)=posf(i-1,1); %Toma el valor X de la posición anterior
     end
     if posf(i,2)==0 %Si no se especifica el valor de Y
       posf(i,2)=posf(i-1,2); %Toma el valor Y de la posición anterior
     end
     if posf(i,3)==0 %Si no se especifica el valor de Z
       posf(i,3)=posf(i-1,3); %Toma el valor Z de la posición anterior
     end
   end

   %Trasladar la pieza al punto (2,2,0) y pasar a metros
   coordinates=posf(1:filf,1:5);
   [filc,colc]=size(coordinates);
   minx=coordinates(1,1);
   miny=coordinates(1,2);
   for i=1:1:filc
     if coordinates(i,1)<minx
       minx=coordinates(i,1);
     end
     if coordinates(i,2)<miny
       miny=coordinates(i,2);
```

```
      end
   end
   minx=minx-2;
   miny=miny-2;
   for i=1:1:filc
       coordinates(i,1)=(coordinates(i,1)-minx)/1000;
       coordinates(i,2)=(coordinates(i,2)-miny)/1000;
       coordinates(i,3)=coordinates(i,3)/1000;
   end
end %Fin de función
```

3.3. CONCLUSIONES

En este estudio se ha desarrollado una nueva técnica de modelado CAD basada en operaciones de barrido para poder obtener cualquier pieza a partir de un fichero G-code, sin limitaciones de geometría, de forma automatizada y con capacidad para obtener el modelo 3D en pocos segundos.

Para comprobar su utilidad y ventajas o desventajas como herramienta de predicción de piezas impresas antes de su fabricación, será necesario compararla con otras alternativas de modelado, tanto basadas en geometría como utilizando otro tipo de enfoque.

Capítulo 4. Revisión de alternativas de modelado

Con el objetivo de comparar la metodología desarrollada en este trabajo con otras alternativas, se ha realizado una revisión de otras metodologías basadas en geometría y vóxeles. Tras analizar las limitaciones de cada uno de ellas, se elegirá la mejor opción disponible.

4.1. ALTERNATIVAS DE MODELADO BASADAS EN GEOMETRÍA

Con la finalidad de dar solución a los problemas de mallado o auto-intersección que se producen en ciertas piezas utilizando Abaqus/CAE 6.14-1, se barajaron otras alternativas de software para el modelado, analizando su potencial para automatizar el proceso. Entre el software probado se encuentran Blender 3.2.2, Gmsh 4.8.0 [73] y Autodesk Inventor Professional 2023.

4.1.1. Blender 3.2.2

En Blender 3.2.2 se probaron dos maneras de modelado. En la primera de ellas, se establece la trayectoria de la curva y se le atribuye un perfil que también debe ser dibujado. El segundo método se basa en la repetición de un cuerpo a lo largo de una curva, donde el cuerpo sería un prisma cuya base tengo la forma del perfil deseado y la curva, la trayectoria de barrido a seguir.

En este caso, no se ha procedido a automatizar el proceso utilizando Python ya que existen limitaciones relacionadas con el proceso de modelado del software.

Una de esas limitaciones es la auto-intersección de los filamentos, que genera problemas en el mallado del modelo para su posterior análisis mecánico. En la Figura 24 se muestra la diferencia entre definir una trayectoria con los vértices biselados o no. A priori, puede parecer que el sólido de la Figura 24a.1, con vértices biselados, es una mejor representación, ya que su sección es constante. Sin embargo, al observar la estructura, se detectan intersecciones en el interior. A parte de esto, en ambos casos existe auto-intersección entre el principio y el final de la trayectoria.

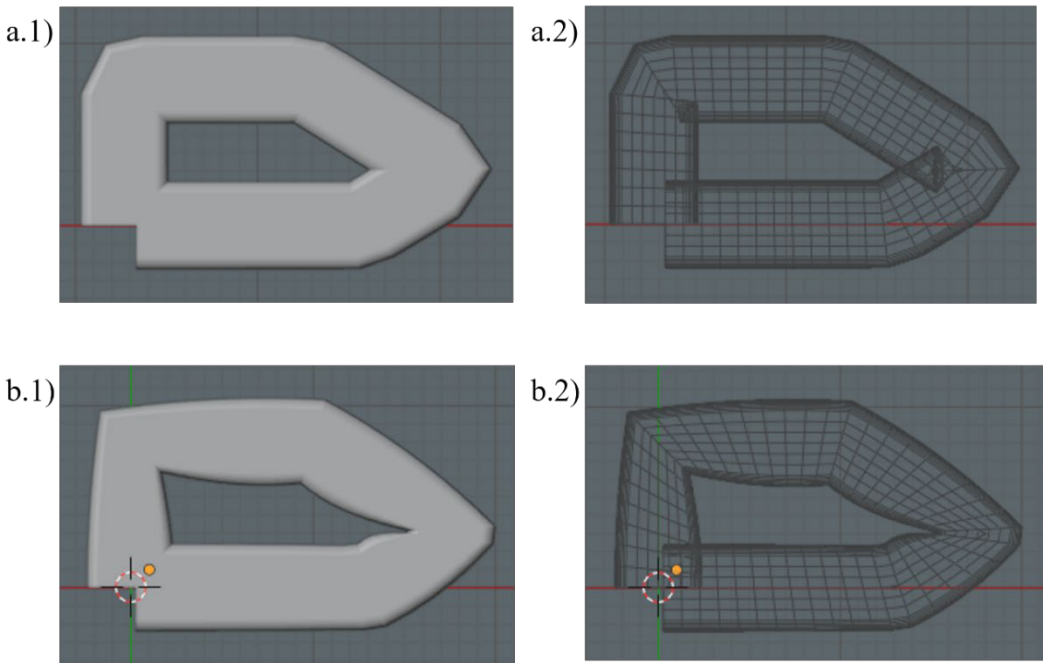


Figura 24. Resultados de modelado en Blender 3.2.2: con biselado de vértices de la trayectoria (a) y sin biselar (b), en vista de sólido (1) y de estructura (2)

Además, el modelado de los cuerpos y barridos en Blender 3.2.2 es tipo contorno, es decir, solamente se representa la superficie límite de estos objetos. Para obtener sólidos se debe añadir el modificador específico para solidificar, pero necesita dos cuerpos para ser ejecutado y su resolución no es óptima, como se puede observar en la Figura 25.

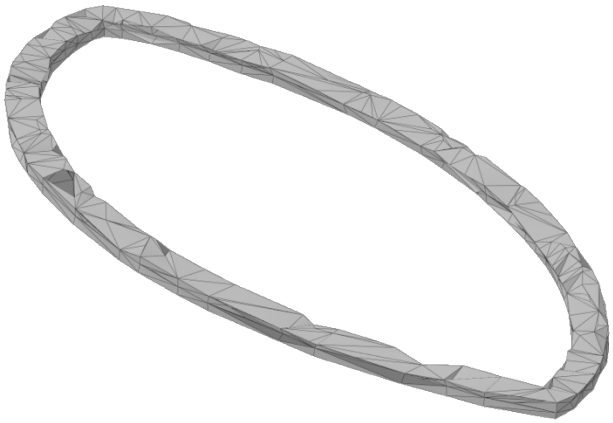


Figura 25. Resolución solidificar en Blender 3.2.2

Por todas estas limitaciones, se ha descartado esta alternativa para el modelado de piezas 3D.

4.1.2. Gmsh 4.8.0

Debido a que la mayoría de los problemas detectados en el modelado CAD tienen

que ver con el mallado, se decidió generar directamente la malla en Gmsh 4.8.0 [73].

Para el proceso de modelado se ha diseñado una subrutina en Matlab (Anexo E) que tomará como referencia una tabla de coordenadas en formato .xlsx, la cual se puede obtener mediante las funciones de generación de coordenadas diseñadas en este trabajo, y genera como salida un archivo .txt, apto para ser interpretado por Gmsh 4.8.0.

En este archivo, primero, se establecen los ocho puntos que formarán el perfil y, luego, las líneas y circunferencias que unen estos puntos. Se define el sentido del bucle y se especifica que forma una superficie. Esta superficie será extruida una distancia, que deberá tener un formato u otro dependiendo del tipo de trayectoria (lineal o curva). A continuación, la superficie que se extruirá será el perfil resultante al final de la trayectoria anteriormente generada. En la Figura 26 puede observarse cómo quedaría un tramo de la trayectoria de impresión de una pieza con Gmsh 4.8.0, así como el perfil definido por ocho puntos.

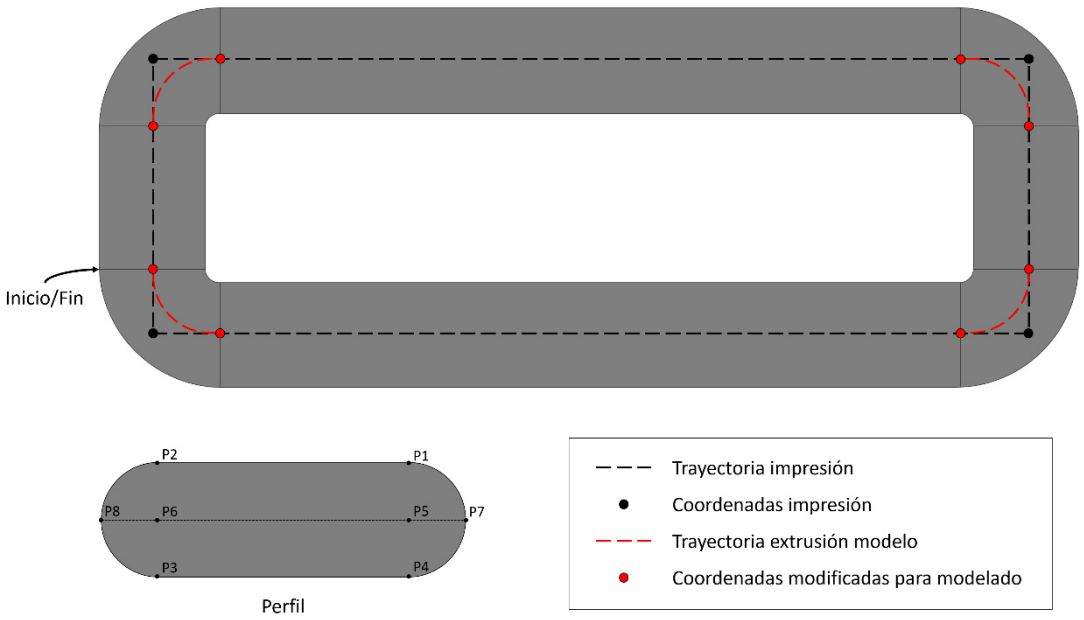


Figura 26. Generación de modelo mediante extrusión en Gmsh 4.8.0

Cabe destacar, que se optó por modificar la trayectoria, redondeando las esquinas, para evitar las esquinas incompletas, que a su vez producían auto-intersecciones en el modelo y requerían la representación de un nuevo perfil para cada tramo. Esta limitación se muestra en la Figura 27a (1).

Además, en la representación de las trayectorias curvas mediante Gmsh 4.8.0, se debe tener en cuenta el ángulo de rotación. Mientras que en giros de 90°, como los

presentados en la Figura 27b, lo único que se necesita para calcular las nuevas coordenadas que definirán el giro, es restar a las trayectorias anteriores y posteriores el valor de la mitad del ancho de extrusión; si la esquina presenta un ángulo interior muy agudo, habrá que realizar un cálculo más complejo, teniendo en cuenta el punto de corte de las dos rectas que lo forman. Tras hallar este punto, se deberá buscar los puntos más cercanos al corte, entre los que exista una distancia suficiente para generar una revolución del perfil. Esto acortará el modelo, por tanto, es otra limitación que presenta el modelado en Gmsh 4.8.0, y puede observarse en la Figura 27c.

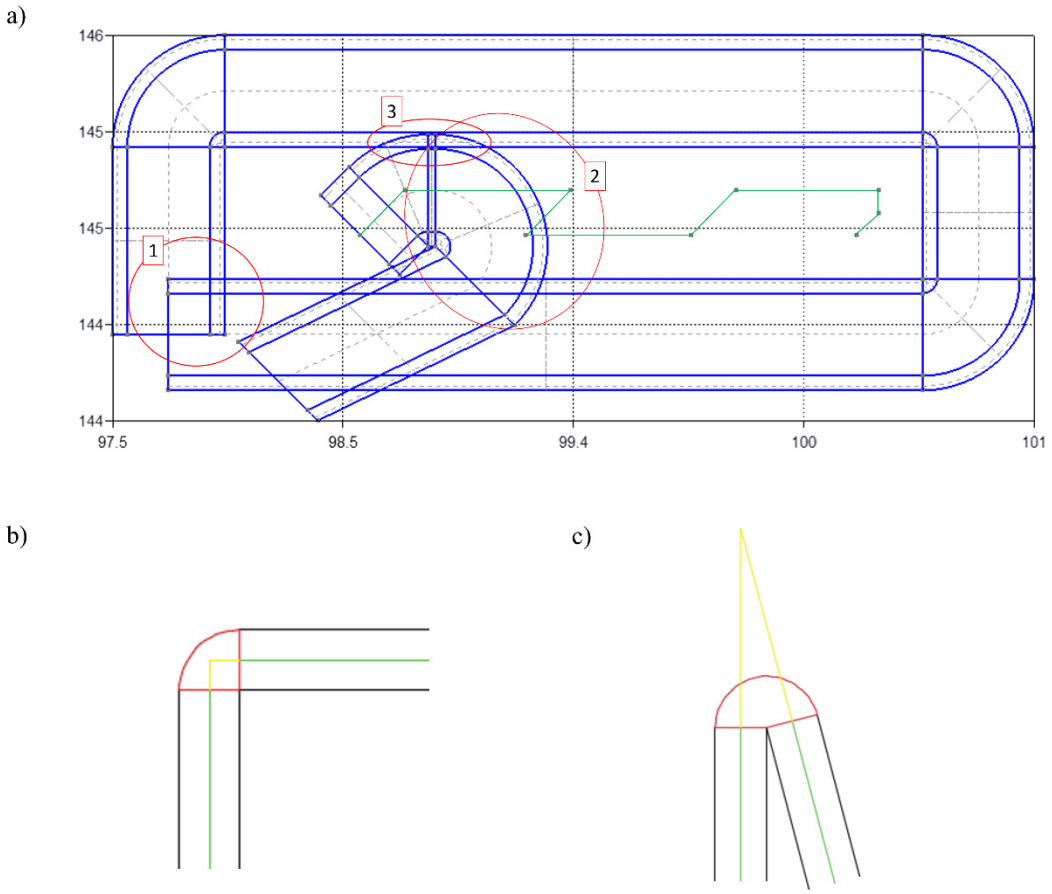


Figura 27: Limitaciones Gmsh: a) auto-intersecciones y resultado de giros en trayectorias pequeñas, b) giro a 90°, y c) giro con ángulos agudos (acortamiento modelo)

Relacionado con lo anterior, debido a estos giros en las esquinas, cuando las trayectorias son muy pequeñas, estas pueden deformarse y no poder tomar la forma deseada, como se muestra en la Figura 27a (2). La trayectoria a seguir sería la trazada en verde, pero el giro del filamento no permite su correcta representación y, como se enumeró anteriormente, un corte por tramos no es viable, ya que auto-interseccionaría. Adicionalmente, se han detectado también auto-intersecciones entre distintos tramos, por ejemplo, relleno-perímetro, como se muestra en Figura 27a (3).

Por todos los motivos anteriormente descritos, el modelado mediante Gmsh

4.8.0 se ha descartado como alternativa.

4.1.3. Autodesk Inventor 2023

Por último, se creó una subrutina para generar el modelado automático en Autodesk Inventor Professional 2023 (Anexo F).

En este caso, el archivo de entrada es una hoja Excel (.xlsx) con las coordenadas de la pieza y, el de salida, es un archivo .cls, con la definición de las trayectorias como croquis 3D y los perfiles como croquis 2D, en un plano de trabajo situado al final de cada trayectoria. Cabe destacar que el programa automático genera varios archivos de salida, ya que el software no acepta más de 1000 filas de entrada, al ejecutar el archivo .cls. En la Figura 28 se muestra la representación de los filamentos utilizando la operación de barrido, modelados mediante Autodesk Inventor Professional 2023.

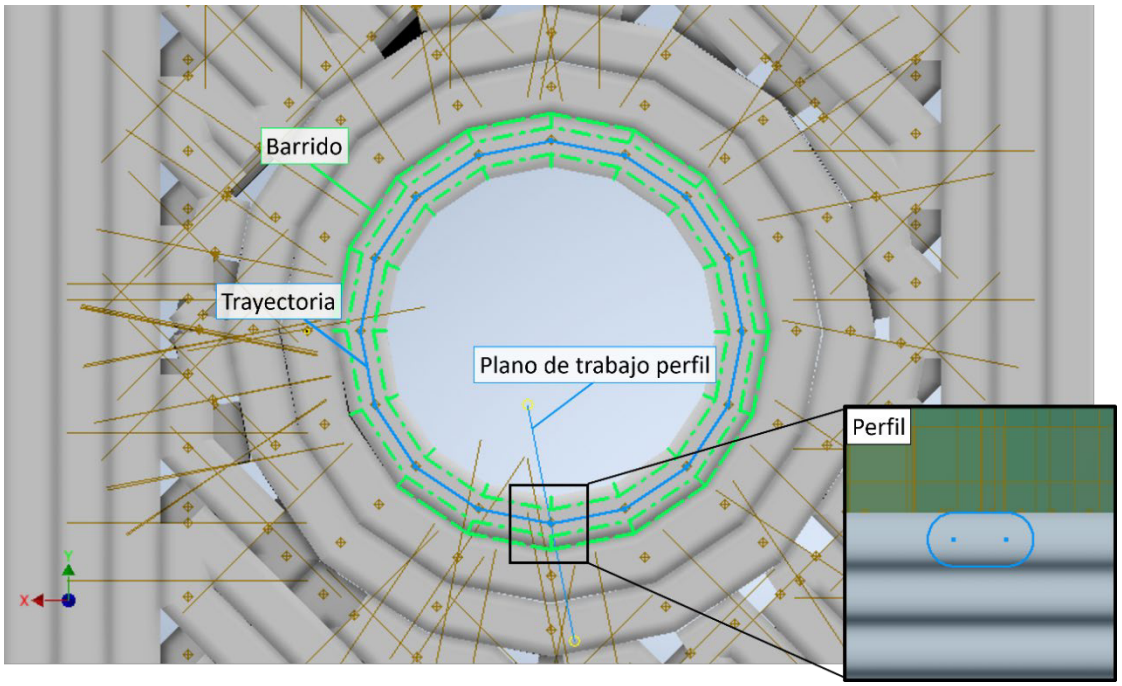


Figura 28. Modelado con Autodesk Inventor

Además, en dicha figura puede observarse la cantidad de puntos y planos de trabajo que se generan durante el modelado. En este caso, este número ha sido superior al deseado, ya que el modelo ha requerido una gran cantidad de cortes y omisiones de trayectoria para poder ser representada. Estas operaciones se han efectuado de manera manual y no es posible su predicción, por lo que se toma como una limitación que ha resultado en el descarte de este software para el modelado.

4.2. MODELADO BASADO EN VÓXELES

Tras analizar diferentes softwares CAD como alternativas de modelado y observar que no se conseguían mejorar las prestaciones de la metodología desarrollada, el siguiente paso fue evaluar otras técnicas de modelado como la basada en vóxeles. Entre las opciones disponibles en la literatura, cabe destacar la metodología llamada VOLume COnserving model for 3D printing (VOLCO) [31]. La metodología mencionada fue la única que se identificó con un código disponible para su implementación automática. A continuación, se analizan los aspectos básicos de dicha metodología.

4.2.1. Representación del filamento

El método basado en vóxeles, representa cada filamento como una serie de esferas voxelizadas, colocadas unas después de otras, teniendo en cuenta las interacciones entre ellas y conservando el volumen constante. La peculiaridad de este modelado es que tiene en cuenta la interacción entre filamentos, es decir, cuando un nuevo filamento va a depositarse donde ya hay otro anterior, el material de esta unión se expande, como se muestra en la Figura 29. Este fenómeno es una representación realista de lo que ocurre al imprimir piezas en 3D y parte de la premisa de que los filamentos depositados no son cilindros perfectos [31].

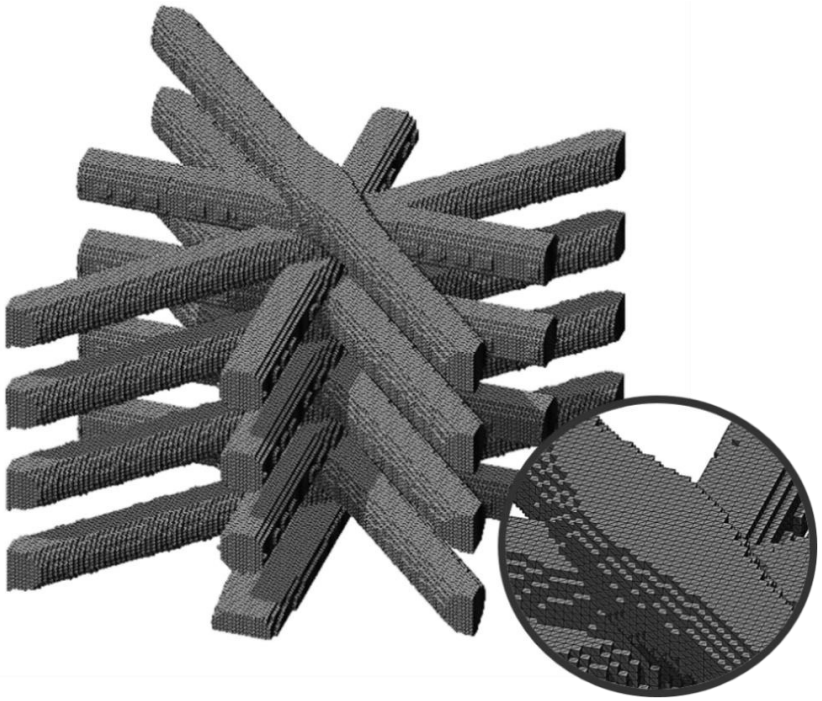


Figura 29. Representación de filamentos en VOLCO y expansión de las uniones

4.2.2. Descripción del proceso

La generación de un modelo voxelizado con VOLCO se realiza mediante macros en Microsoft Excel, en el archivo denominado "*Setup.xls"*. Para ello se deben definir las coordenadas (las cuatro primeras columnas de la matriz) en la pestaña "Toolpath", el tamaño de vóxel, el radio de la esfera, altura de capa y área de trabajo. Una vez definido lo anterior, se generan una serie de archivos que son procesados con varias subrutinas de Matlab, generando el archivo STL que contiene el modelo.

En el último paso de la generación del modelo, también se obtienen los datos de volumen, altura y porosidad. Para que el modelo obtenido se acerque lo máximo posible al modelo teórico, se compara el nuevo volumen con el que, en teoría, se depositaría según el G-code. Con la función "spline" en Matlab, se podrá realizar una interpolación de volúmenes y radios de esfera (*OriginalSphereRadius*), iterativamente, hasta obtener un modelo con un volumen igual o muy similar al teórico.

Como se muestra en la Figura 30, una vez obtenido el STL del modelo 3D ajustado al volumen teórico, se importa el archivo en Abaqus/CAE 6.14-1, se aplica el AEF y se obtienen las fuerzas de reacción y, con ello, el módulo de Young.

4.2.3. Limitaciones

La limitación que presenta el VOLCO en la generación del modelo voxelizado, se debe a la memoria RAM disponible en el equipo, ya que, dentro de las subrutinas de Matlab, se trabaja con matrices de vóxeles en tres dimensiones, en las cuales se definen los espacios llenos (1) o vacíos (0) y cada uno de estos espacios corresponde con un vóxel. Con lo cual, cuanto más grande es la pieza y cuanto más pequeño sea el tamaño de vóxel, se generará un matriz más grande que podría no ser soportada por la memoria.

4.3. CONCLUSIONES

Tras haber analizado las diferentes alternativas de modelado propuestas, se ha decido comparar el modelado DECODE con la metodología basada en vóxeles, VOLCO. Esta comparación resulta más interesante por el cambio de enfoque en la base del modelado. Además, la limitación más importante de esta última metodología es la eficiencia computacional, por lo que será otro aspecto a comparar con DECODE.

Capítulo 5. Comparación de metodologías de modelado

En este capítulo se comparan varias metodologías de modelado y se determina cuál es la idónea, según la pieza que se desee modelar.

En los siguientes apartados, se describen dos metodologías, utilizadas para obtener los modelos 3D de piezas impresas, a los que luego se les ha aplicado el análisis de elementos finitos (AEF) para estudiar su comportamiento mecánico. Estas dos metodologías son la basada en geometría, denominada AutomateD SwEep CAD modeller Of Extrusion-based GcoDE (DECODE) [71], y la basada en vóxeles, llamada VOLume COnserving model for 3D printing (VOLCO) [31]. La primera de ellas se ha desarrollado en esta tesis y, la segunda, es parte de la investigación de Andrew Gleadall, Universidad de Nottingham. Esta comparación se ha publicado en el artículo de acceso abierto "Comparison of CAD and Voxel-Based Modelling Methodologies for the Mechanical Simulation of Extrusion-Based 3D Printed Scaffolds" [72]. Los pasos para la obtención de los modelos con cada metodología se muestran en la Figura 30.

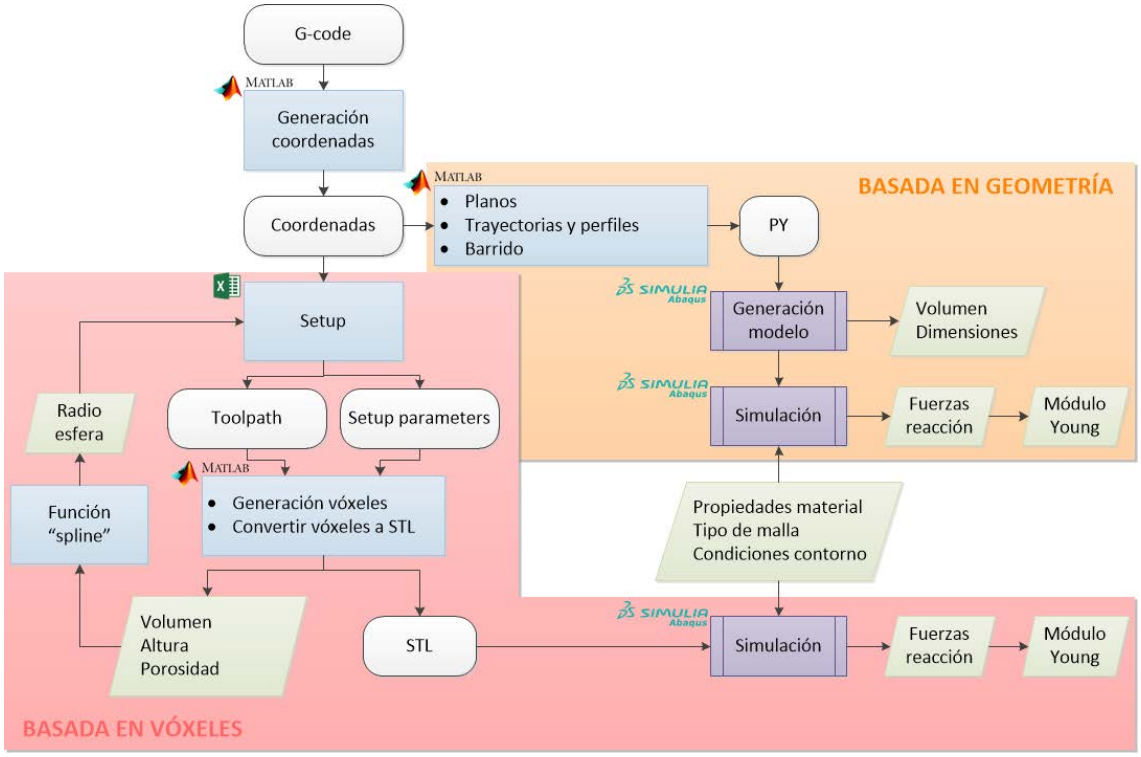


Figura 30. Proceso de modelado de metodologías

Ambas metodologías parten de un archivo G-code, utilizado para la impresión de la pieza en 3D. Mediante una subrutina en Matlab, se obtienen las coordenadas para modelar la pieza.

5.1. OBTENCIÓN DE COORDENADAS

5.1.1. Descripción del proceso

Como se ha puntualizado anteriormente, ambas metodologías parten de las coordenadas obtenidas a partir del G-code. Estas se obtienen con una subrutina de Matlab que necesita como entrada el archivo G-code comentado, con el "Verbose G-code" activado.

Actualmente, esta subrutina está adaptada para obtener coordenadas de los G-code generados en Slic3r 1.3.0, Simplify3D 4.1.1 o FullControl G-code Designer, eliminando comandos y comentarios que no son necesarios, localizando las coordenadas XYZ y teniendo en cuenta los comentarios que definen una falda, un relleno, un perímetro (interior o exterior), un soporte, movimientos al primer punto de una trayectoria, movimientos a la siguiente capa, movimientos de retracción o una elevación del cabezal. El código explicado paso a paso se ha descrito en los apartados 3.2.4, 3.2.5 y 3.2.6.

Una vez obtenidas las coordenadas XYZ y sabiendo si el movimiento entre una y otra es con extrusión o al vacío, o si se trata del primer punto de una trayectoria, se obtiene una matriz de coordenadas con cinco columnas y un número de filas igual al de coordenadas. Esta matriz puede trasladarse a un archivo Excel (.xls/.xlsx) para ser utilizada tanto en VOLCO como en DECODE, aunque para este último no es necesario ya que puede trabajar con la matriz albergada en Matlab (.mat).

En caso de desear trabajar con VOLCO, se utilizarían las funciones descritas anteriormente como subrutinas independientes. Es decir, se eliminaría el encabezado en el que se comienza la función y el "end" que la cerraría. En su lugar, será necesario introducir unas primeras líneas que lean el nombre del fichero de entrada y de salida y un último bloque para guardar las coordenadas en un archivo Excel.

```
%LECTURA NOMBRES ARCHIVOS
filename_gcode=input('Enter the name of the G-code file (ending in
.gcode): ','s');
filename_xls=input('Enter the name of the Excel file (ending in .xlsx):
','s');


%GRABACIÓN COORDENADAS EN .XLSX
[fil,col]=size(coordenadas);
i_end=floor(fil/1000000);

%Si el número de coordenadas excede la dimensión máxima de la hoja de
%datos, se dividen en distintas pestañas
if i_end>0
```

```
for i=0:1:i_end-1
A=coordenadas(1+1000000*i:1000000+1000000*i,1:4);
sheet=1+1*i;
xlswrite(filename_xls,A,sheet);
end
A=coordenadas(i_end*1000000+1:fil,1:4);
sheet=sheet+1;
xlswrite(filename_xls,A,sheet);
else
    xlswrite(filename_xls,coordenadas);
end
```

5.1.2. Limitaciones

La principal limitación del proceso de obtención de coordenadas se da al guardar estas coordenadas en una hoja Excel, ya que, el número de coordenadas puede exceder el número máximo de filas de una hoja de datos (1048576 filas). Con lo que, para piezas con una gran cantidad de coordenadas, no podría ejecutarse su modelado completo en VOLCO, ya que sólo dispone de una hoja de datos para albergar el Toolpath.

En el caso de DECODE, esta limitación se omite gracias a la posibilidad de leer las coordenadas desde la matriz (.mat) o de leer varias hojas de datos de un mismo documento Excel, consecutivamente. La limitación que existiría en este caso, sería por la memoria RAM disponible en el equipo utilizado. Según el valor de esta, variará el número máximo de coordenadas que podrán almacenarse.

5.2. ANÁLISIS DE ELEMENTOS FINITOS

5.2.1. Descripción del proceso

Una vez se obtiene el modelo de la pieza impresa en Abaqus/CAE 6.14-1, sea mediante la ejecución del archivo Python en la metodología DECODE o importando el archivo STL generado por VOLCO, se puede proceder a la simulación de este. En la Figura 31, se muestra el diagrama con todos los pasos para el AEF.

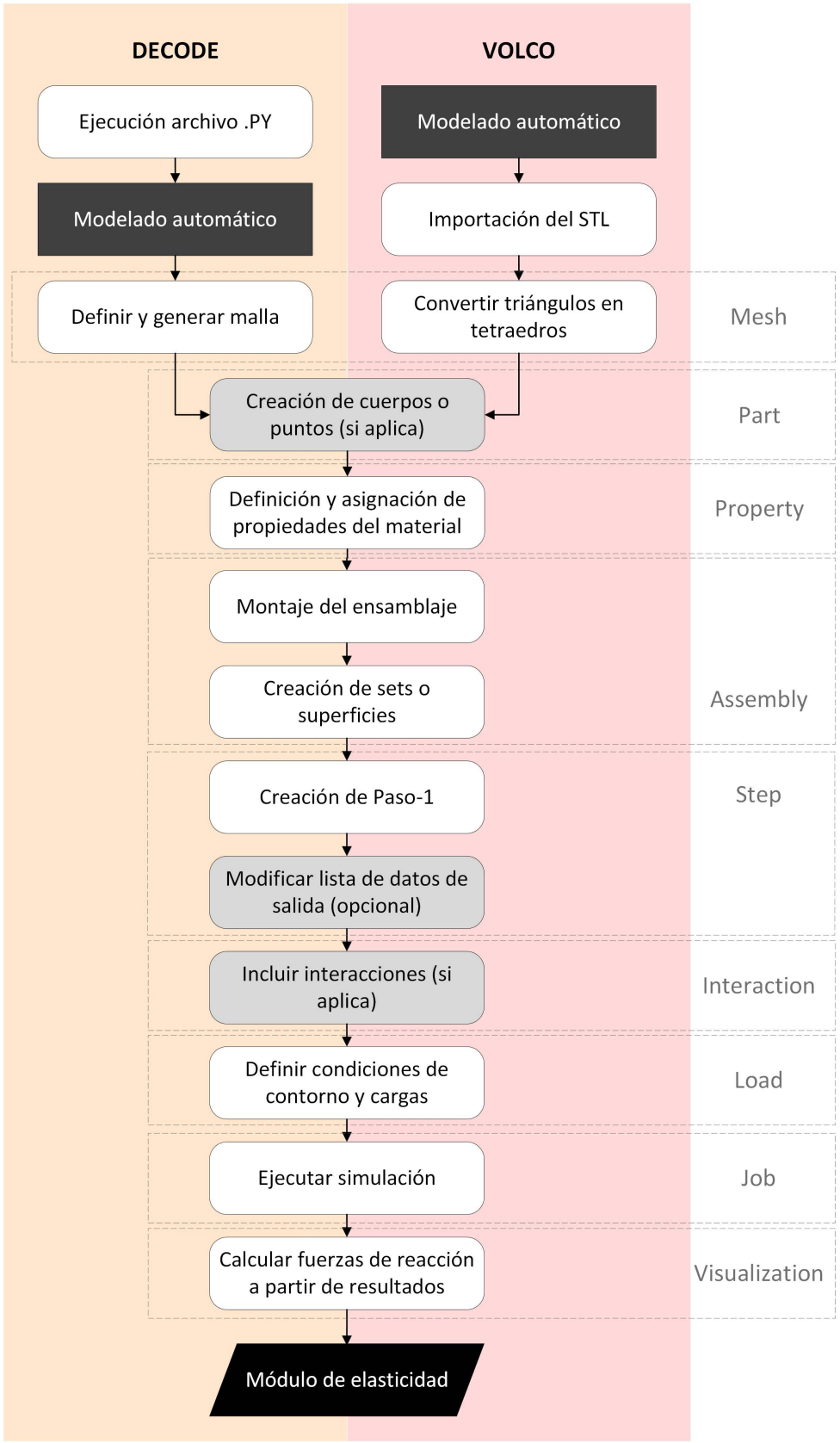


Figura 31. Pasos para la aplicación de AEF a los modelos generados con DECODE y VOLCO, con postprocesado centrado en la determinación del módulo de elasticidad a partir de las fuerzas de reacción obtenidas en el análisis y desplazamiento aplicado

En el caso de importar el modelo voxelizado, el primer paso será acudir al módulo "Mesh" para editar la malla y convertir los triángulos en tetraedros. Si se trata del modelo generado en Abaqus, no voxelizado, se deberá crear la malla de elementos finitos de la geometría en el módulo "Mesh", definiendo el tipo de malla y el tamaño de semilla. A continuación, en ambos casos, será necesario definir las propiedades del material en el módulo "Property", como la densidad y módulo de elasticidad, y atribuirle dichas características a la pieza. El paso primordial en cualquier simulación es la definición de los pasos de carga en el módulo "Step", para luego establecer las condiciones de contorno o carga aplicadas, en "Load". Según el caso práctico y el tipo de ensayo, podría requerirse más pasos en la preparación de la simulación, como la creación de conjuntos (sets) o superficies en "Assembly", a partir de la geometría, o la definición de interacciones entre dichas superficies en "Interaction". Además, un paso opcional para el ahorro de tiempo de cálculo es modificar los datos de salida (Field Output) en el módulo "Step". Finalmente, se ejecutará la simulación en el módulo "Job" y, una vez se complete, podrán analizarse los resultados desde el módulo "Visualization".

En los Anexos se presentan detalladamente dos ejemplos, un ensayo de una miniprobeta a flexión (Anexo G) y un ensayo de un scaffold a compresión (Anexo H), con los pasos requeridos para su simulación en Abaqus/CAE 3.14-1.

Por último, una vez simulado el modelo, se podrán calcular las fuerzas de reacción y así calcular el módulo de elasticidad de la pieza.

5.2.2. Limitaciones

El software de simulación puede presentar limitaciones, sobre todo al mallar una pieza. Se deberá probar con distintas mallas y tamaños de semilla o, incluso, aplicar "Virtual Topology", una herramienta que permite ignorar detalles, como caras y aristas muy pequeñas o redundantes, para simplificar la malla y que sea posible el análisis numérico. Sin embargo, todavía existe la posibilidad de obtener elementos distorsionados al mallar, en particular, en piezas complejas, y, por tanto, no poder finalizar el proceso de simulación.

Además, la memoria del equipo también limitará el proceso de análisis. Esto podría ocurrir en piezas demasiado grandes, para las cuales los cálculos a realizar no podrían ser soportados.

5.3. CASO PRÁCTICO: COMPARACIÓN DE METODOLOGÍAS

Con el objetivo de comparar las dos metodologías de modelado presentadas anteriormente, DECODE y VOLCO, y comparar sus resultados con datos teóricos y experimentales, se toma como pieza de ensayo un scaffold cilíndrico.

Además, se aplicarán diferentes patrones y porcentajes de relleno para comparar la variación de los resultados en función de la geometría de la pieza.

Por otro lado, en los Anexo G y Anexo H, se han incluido dos ejemplos, una miniprobeta y un scaffold, respectivamente, modelados con DECODE y VOLCO y simuladas en Abaqus/CAE 6.14-1, en los que se describen los pasos seguidos para aplicar la metodología de modelado.

5.3.1. Material

El material de impresión empleado para la fabricación de los scaffolds utilizados en este experimento fue la policaprolactona (PCL), un poliéster biodegradable con propiedades adecuadas para aplicaciones en la ingeniería tisular. Las propiedades del material se muestran en la Tabla 3.

Tabla 3. Propiedades del material PCL

Propiedades del material PCL – Regemat 3D	
Peso molecular	50000 g/mol
Densidad	1.1 g/cm³
Fuerza de tracción	45 MPa
Elongación en el punto de fluencia	15 %
Módulo de elasticidad	350 MPa
Resistencia al impacto IZOD	8 kJ/m2
Dureza Shore D	46
Temperatura de deflexión térmica (0.45 MPa)	57 °C

5.3.2. Geometría y parámetros de fabricación

Para este estudio se ha utilizado un scaffold cilíndrico (Ø10 x 7 mm). En primer lugar, se introdujo el cilindro sólido en el software laminador (Slic3r 1.3.0) y se seleccionaron tres configuraciones de impresión diferentes, para obtener tres piezas con diferentes patrones (rectilíneo y giroide) y densidades de relleno (40 y 50% de densidad, lo que equivale al 60 y 50% de porosidad, respectivamente). La Tabla 4 muestra los parámetros de impresión de las tres configuraciones estudiadas: un scaffold definido por un patrón rectilíneo y una porosidad del 50% (50_rect), uno

similar al primero pero con una porosidad del 60% (60_rect) y otro con un patrón de relleno giroide y una porosidad del 50% (50_gyr).

Tabla 4. Parámetros de impresión de scaffolds cilíndricos

Parámetros de impresión		
Parámetros comunes para todas las piezas		
Ancho de extrusión de perímetros externos		0.44 mm (0.39 mm^3/s)
Ancho de extrusión de perímetros		0.48 mm (0.88 mm^3/s)
Ancho de extrusión de relleno		0.48 mm (2.51 mm^3/s)
Diámetro de filamento		1.75 mm
Velocidad de relleno		20 mm/s
Diámetro de boquilla		0.4 mm
Temperatura		190 °C
Altura de primera capa		0.35 mm
Altura de capa		0.3 mm
Ángulo de relleno		90°
Parámetros particulares		
50_rect	Densidad de relleno	50%
	Patrón de relleno	Rectilíneo
60_rect	Densidad de relleno	40%
	Patrón de relleno	Rectilíneo
50_gyr	Densidad de relleno	50%
	Patrón de relleno	Giroide

Tras establecer las características de los scaffods, se obtuvieron los archivos de código G con Slic3r 1.3.0. para la impresión y modelado de las piezas. En este caso, se modelaron tanto con DECODE como con VOLCO para aplicar la simulación mediante AEF y poder comparar los resultados.

5.3.3. Análisis de Elementos Finitos

Para obtener la rigidez y el módulo elástico de los modelos de scaffolds, se simularon en Abaqus/CAE 6.14-1 mediante AEF. Concretamente, se aplicó un ensayo de compresión con las siguientes condiciones de contorno: se restringió el movimiento vertical en la base del scaffold, se encastraron dos puntos de esta base para evitar el desplazamiento en el plano horizontal y se aplicó un desplazamiento vertical de 0.2 mm en la capa superior para simular el movimiento de la placa de compresión.

Otras partes importantes del proceso de simulación son la selección de las propiedades del material y la definición del tipo de malla. La densidad y el módulo de elasticidad son los especificados en la ficha técnica del PCL (Tabla 3), y el coeficiente de Poisson se definió como 0.46, de acuerdo a algunos estudios [74,75]. Se asumió un

comportamiento lineal del material elástico y se incluyeron los efectos no lineales de las grandes deformaciones y desplazamientos.

Por otro lado, los modelos obtenidos mediante DECODE se mallaron como elementos de tipo C3D10 (tetraedro cuadrático de 10 nodos) y C3D4 (tetraedro lineal de 4 nodos), y los modelos basados en vóxeles sólo como C3D4, el tipo de elemento por defecto para transformar vóxeles en tetraedros. Esta diferencia se debe a que el modelo voxelizado no permite el mallado con elementos de segundo orden. El tamaño de semilla de la malla se ha variado en función de la pieza, así como la aplicación de topología virtual, para evitar elementos distorsionados. Las condiciones de mallado para cada modelo se muestran en la Tabla 5. Además, se asumieron uniones rígidas entre capas (un único sólido).

Tabla 5. Asignación de malla

Pieza	Metodología	Tipo de malla	Tamaño de semilla (mm)	Topología virtual
50_rect	DECODE	C3D10	0.3	Sí
		C3D4	0.4	No
	VOLCO	C3D4	0.05	No
60_rect	DECODE	C3D10	0.1	No
		C3D4	0.1	No
	VOLCO	C3D4	0.05	No
50_gyr	DECODE	C3D10	-	-
		C3D4	0.05	No
	VOLCO	C3D4	0.05	No

Durante el proceso de preparación del modelo para la simulación en Abaqus/CAE 6.14-1, no fue posible mallar el scaffold 50_gyr con elementos cuadráticos.

Tras la simulación, se obtuvieron las fuerzas de reacción en la capa base y, por lo tanto, se evaluó el módulo de Young de acuerdo con la Ecuación (3), extraída de la norma ISO 604 (Plásticos. Determinación de las propiedades en compresión). Nótese que el área equivalente corresponde al área de la base de la pieza sólida.

$$E = \frac{\sigma}{\varepsilon} = \frac{F/A}{\Delta L/L_0} = \frac{F \cdot L_0}{A \cdot \Delta L} \tag{3}$$

En la ecuación, E representa el módulo de Young o módulo de elasticidad (MPa), σ es el esfuerzo de compresión (MPa), ε es la deformación, F es la fuerza de reacción (N), A es el área equivalente (mm^2), ΔL es el desplazamiento (mm) y L_0 es la altura original (mm).

5.3.4. Fabricación de scaffolds y pruebas experimentales

Los scaffolds reales fueron fabricados por extrusión de material (MEX) en una impresora 3D BQ Hephestos 2. Además, la temperatura de extrusión se fijó en 190°.

Las réplicas sometidas a caracterización mecánica bajo carga de compresión fueron cuatro por grupo. Los ensayos mecánicos se realizaron en una máquina de ensayo LIYI (LI-1065, Dongguan Liyi Environmental Technology Co., Ltd., China) en el modo de control de desplazamiento. La velocidad de la cruceta se fijó en 1 mm/min y el módulo de compresión se calculó como la pendiente del segmento inicial en el gráfico tensión-deformación resultante.

Por otro lado, los scaffolds impresos fueron comparados geométricamente con los modelos, mediante imágenes obtenidas del microscopio, para determinar qué representación se acerca más a la realidad. Concretamente, se utilizó el microscopio Olympus BX51 con un factor de aumento x2 para recoger imágenes de la parte superior de los tres tipos de scaffolds (50_rect, 60_rect y 50_gyr).

5.3.5. Resultados y discusión

Con el objetivo de comparar las metodologías de modelado, se han analizado sus resultados a efectos de eficiencia, precisión dimensional y módulo de elasticidad.

5.3.5.1. Eficiencia del modelado

Con respecto a la eficiencia del modelado, esta se estima a partir del tiempo y la CPU requeridos para los procesos de modelado y simulación. Los tiempos necesarios para generar las coordenadas y seguir los pasos de modelado y simulación en cada configuración de los scaffolds se recogen en la Tabla 6, así como la memoria mínima necesaria para su simulación.

El ordenador utilizado para el modelado y la simulación dispone de un procesador Intel® Core™ i9-9820X CPU @ 3.30GHz 3.31 GHz, una memoria RAM instalada de 64.0 GB y un sistema operativo de 64 bits.

El uso de mallas C3D4 ahorra memoria en la simulación de los modelos, pero no siempre tiempo. En general, el modelado y la simulación con DECODE son más eficientes, aunque es muy similar a VOLCO para geometrías complejas como el giroide, que también presenta la dificultad de no poder ser mallado con elementos de segundo orden con el software y hardware disponibles.

Tabla 6. Tiempo y memoria CPU requerida para modelar y simular los scaffolds

Pieza	Metodología	Tipo malla	Generación coordenadas (s)	Proceso modelado (s)	Proceso simulación (s)	Tiempo total (h:mm:ss)	Memoria mínima requerida (MB)
50_rect	DECODE	C3D10	10	55	255	0:05:10	1925
		C3D4			477	0:08:52	326
	VOLCO	C3D4		2339	4967	2:01:46	9549
60_rect	DECODE	C3D10	7	33	1559	0:26:32	7491
		C3D4			319	0:05:52	1238
	VOLCO	C3D4		959	2955	1:05:14	7985
50_gyr	DECODE	C3D10	7	338	-	-	-
		C3D4			2970	0:55:08	6905
	VOLCO	C3D4		920	2280	0:53:20	6901

Los modelos generados mediante las distintas técnicas de modelado se presentan en la Figura 32.

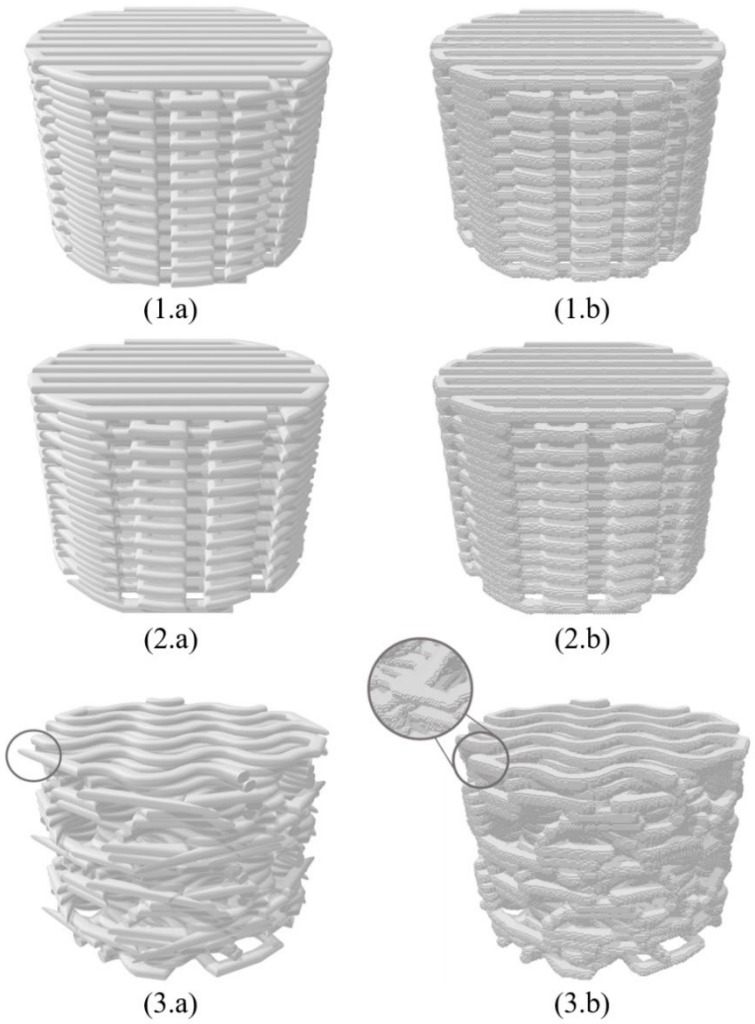


Figura 32. Modelos DECODE (a) y VOLCO (b): 50_rect (1), 60_rect (2) y 50_gyr (3)

Se han estudiado las metodologías de modelado y simulación en términos de aplicabilidad a toda la gama de estructuras MEX anteriores. En el caso particular de la adecuación dimensional y de forma, se han encontrado limitaciones en función del software y hardware utilizado, ya que, para piezas grandes o complejas, Abaqus/CAE 6.14-1 no puede soportar su simulación, mallado o modelado. Por ejemplo, la pieza de gran tamaño presentada en la Figura 33.a no fue posible modelarla a partir de un fichero Python. Por otro lado, el scaffold 50_gyr o la pieza de esquina mostrada en la Figura 33.b fueron modelados pero no mallados por Abaqus/CAE 6.14-1 debido a la geometría compleja y a la gran cantidad de memoria necesaria para mallar la pieza, respectivamente. Finalmente, en otros casos, las piezas pueden ser modeladas y malladas pero no simuladas porque la memoria mínima requerida para el AEF es superior a la disponible.

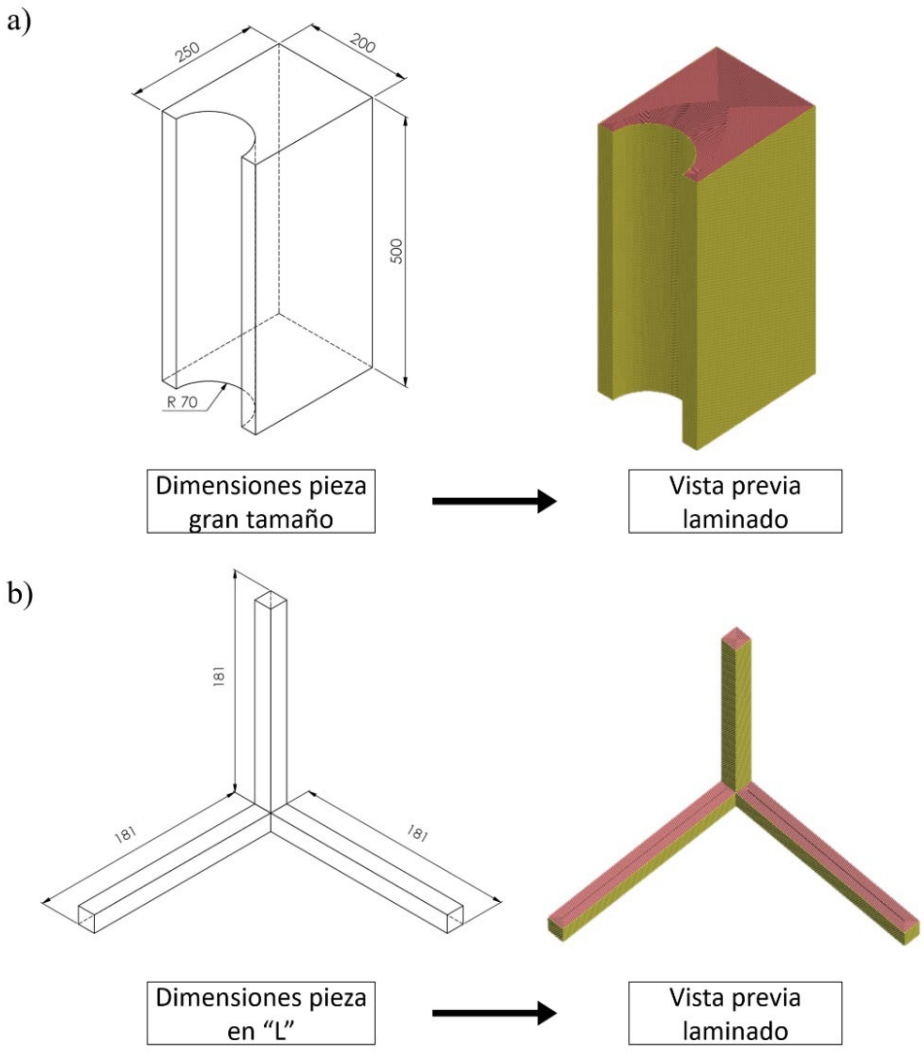


Figura 33. Piezas suplementarias (dimensiones en mm) para la comprobación de las limitaciones de modelado

Por otro lado, en ambas técnicas de modelado se utiliza Microsoft Excel para registrar las coordenadas obtenidas a partir del código G. En el caso más favorable, utilizando Microsoft Excel 64 bits, la hoja de trabajo se limita a 1048576 filas por 16384 columnas, pero no al uso máximo de 2GB de RAM como en la versión de 32 bits. En resumen, como cada coordenada se guarda en una nueva fila, habrá un límite de 1048576 coordenadas en VOLCO. En DECODE, esto no sería una limitación ya que permite la lectura de varias hojas de trabajo de forma continua, así como la matriz de coordenadas generada en Matlab R2021a, que sólo presentará limitaciones de memoria debido al hardware disponible.

El ordenador utilizado para la modelización tiene una memoria RAM instalada de 64.0 GB, pero la memoria disponible para todas las matrices es de 43.7 GB. Además, cada elemento de la matriz requiere aproximadamente 8 bytes de memoria. Así pues, el número de elementos estará limitado a $5.86 \cdot 10^9$.

Por un lado, las limitaciones de la matriz y las cuatro columnas necesarias para definir una coordinada fijan en $1.465 \cdot 10^9$ el número máximo de coordenadas de la matriz.

Por otro lado, afecta a VOLCO en la construcción de la matriz de vóxeles, que tendrá tantos elementos como vóxeles (definidos de forma binaria) se necesiten para definir la pieza. Este número dependerá de las dimensiones y del tamaño del vóxel, tal y como se define en la Ecuación (4).

$$\text{No. voxels} = \frac{(X_{end} - X_{start}) \cdot (Y_{end} - Y_{start}) \cdot (Z_{end} - Z_{start})}{\text{TamañoVoxel}^3} \qquad (4)$$

donde

Xend, Yend, Zend → Coordenadas máximas (mm)

Xstart, Ystart, Zstart → Coordenadas mínimas (mm)

TamañoVoxel → Indica el tamaño de vóxel (mm)

En la Tabla 7 se muestran varios ejemplos para determinar el tamaño de las estructuras que puede modelar VOLCO a nivel práctico. Las que requieren más memoria de la disponible no podrán ser modeladas. Cabe señalar que se ha fijado un tamaño máximo de vóxel de 100 µm para permitir una resolución de filamento apropiada, ya que el diámetro de la boquilla utilizada es de 0.4 mm, lo que permite alturas de capa máxima de 0.3 mm.

Tabla 7. Número de vóxeles de acuerdo a dimensiones y tamaño de vóxel asignado

Pieza	Dimensiones (mm)	Tamaño de vóxel (µm)	No. mínimo de vóxeles (memoria requerida) vs. No. Máximo de elementos (memoria disponible)
Probeta	80 x 10 x 4	100	$3.2 \cdot 10^6$ (24.41 MB) < $5.86 \cdot 10^9$ (44713 MB)
Probeta	80 x 10 x 4	5	$2.56 \cdot 10^{10}$ (190.73 GB) > $5.86 \cdot 10^9$ (43.66 GB)
Pieza de gran tamaño	200 x 250 x 500	100	$2.5 \cdot 10^{10}$ (186.26 GB) > $5.86 \cdot 10^9$ (43.66 GB)
Pieza en "L"	181 x 181 x 181	100	$5.93 \cdot 10^{10}$ (44.18 GB) > $5.86 \cdot 10^9$ (43.66 GB)

Como resumen, en la Tabla 8 se presentan los pasos que son posibles o no seguir para cada pieza, usando cada metodología de modelado, con el hardware y software disponible.

Tabla 8. Viabilidad de simulación de diferentes piezas

Pieza	VOLCO				DECODE			
	Generación STL	Abaqus			Generación PY	Abaqus		
		Importar	Mallar	AEF		Importar	Mallar	AEF
Probeta	✓	✓	✓	✓	✓	✓	✓	✓
Pieza de gran tamaño					✓			
Pieza en "L"					✓	✓		
Scaffold rectilíneo	✓	✓	✓	✓	✓	✓	✓	✓
Scaffold giroide	✓	✓	✓	✓	✓	✓		
Miniprobeta	✓	✓	✓	✓	✓	✓	✓	✓

En todos los casos, la limitación para el modelado, mallado o AEF es la memoria disponible, excepto para el mallado del modelo DECODE del scaffold giroide y la generación del archivo STL de la pieza de gran tamaño en VOLCO. En el caso del scaffold, se debe a las limitaciones del mallado de piezas complejas en Abaqus/CAE 6.14-1. En el caso de VOLCO, la pieza está definida por 7196287 coordenadas que no pueden registrarse en una sola hoja de trabajo de Microsoft Excel.

5.3.5.2. Precisión dimensional

Uno de los parámetros más importantes para comparar la precisión de la geometría es el volumen. Este puede ser medido en los modelos mediante las herramientas de consulta de Abaqus/CAE 6.14-1, aunque VOLCO también genera datos de salida con esta información. Por otro lado, el volumen teórico se obtiene del archivo G-code que especifica la cantidad de filamento utilizado.

Todos los datos recogidos se comparan en la Tabla 9. La desviación entre los volúmenes teóricos y el modelo VOLCO es casi del 0% debido a la aplicación de la función *spline* mencionada anteriormente.

Tabla 9. Comparación de volúmenes

Pieza	Volumen teórico (mm^3)	Volumen experimental (mm^3)	Metodología	Volumen del modelo (mm^3)	Desviación teórica	Desviación experimental
	Vt	Ve		Vm	|Vm-Vt|/Vt*100	|Vm-Ve|/Ve*100
50_rect	278.8	274.91	DECODE	278.46	0.12%	1.29%
			VOLCO	278.8	0%	1.42%
60_rect	231.16	224.18	DECODE	230.95	0.09%	3.02%
			VOLCO	231.15	0%	3.11%
50_gyr	188.83	183.64	DECODE	188.81	0%	2.82%
			VOLCO	188.83	0%	2.83%

En comparación con los datos teóricos, las diferencias de volumen son insignificantes. Por otro lado, teniendo en cuenta las desviaciones respecto a los resultados experimentales, entre el 1.29% y el 3.11%, se podría asumir que la precisión dimensional de los modelos es aceptable. Esta última afirmación se ve reforzada por la premisa de que los parámetros de impresión seleccionados y la temperatura tienen una influencia considerable en las dimensiones finales de la pieza impresa, que no siempre se acercan a las medidas teóricas [76].

En relación de las dimensiones, también se ha realizado una comparación entre las de los modelos con el diseño sólido inicial (teórico) y los scaffolds impresos (reales). Esta comparación se muestra en la Tabla 10.

Las diferencias entre el scaffold teórico y los modelos radican en las limitaciones de las técnicas de modelado. En el caso de la metodología DECODE, la operación de barrido genera puntas afiladas en ángulos muy agudos. Y, por otro lado, VOLCO expande el filamento en las uniones. Sin embargo, otro factor determinante en la inexactitud dimensional de las piezas impresas mediante MEX son los gradientes de temperatura no uniformes [54].

Tabla 10. Comparación de dimensiones

Pieza	Dimensión	Teórica (mm)	Real (mm)	Metodología	Modelo (mm)	Desviación teórica	Desviación real
		T	R		M	|M-T|/T*100	|M-R|/R*100
50_rect	Diámetro	10	7.06	DECODE	10.104	1.04%	43.12%
				VOLCO	10.025	0.25%	42.00%
	Altura	7	6.04	DECODE	6.95	0.71%	15.07%
				VOLCO	6.95	0.71%	15.07%
60_rect	Diámetro	10	6.83	DECODE	10.098	0.98%	47.85%
				VOLCO	10.05	0.50%	47.14%
	Altura	7	5.91	DECODE	6.95	0.71%	17.60%
				VOLCO	6.95	0.71%	17.60%
50_gyr	Diámetro	10	7.42	DECODE	10.832	8.32%	45.98%
				VOLCO	10.075	0.75%	35.78%
	Altura	7	5.63	DECODE	6.954	0.66%	23.52%
				VOLCO	6.95	0.71%	23.45%

La comparación de la forma y anchura de los filamentos entre los modelos y los scaffolds impresos se realizó para los tipos 50_rect, 60_rect y 50_gyr y se muestra en la Figura 34. Las imágenes y medidas de las piezas impresos se obtuvieron mediante imágenes de microscopio. Como se puede observar, el ancho de los filamentos voxelizados y los reales no es constante, en contraste con el modelo CAD. Sin embargo, en el caso de los modelos voxelizados, el ancho de los filamentos es menor que el correspondiente a los scaffolds impresos.

En resumen, el modelo DECODE puede ser una aproximación de la pieza impresa que, aunque no represente los filamentos de forma variable, el valor medio del ancho es similar al que tendrían a lo largo del recorrido. Por otro lado, el modelo VOLCO presenta una forma final más parecida al scaffold real, aunque el valor del ancho de los filamentos no coincide.

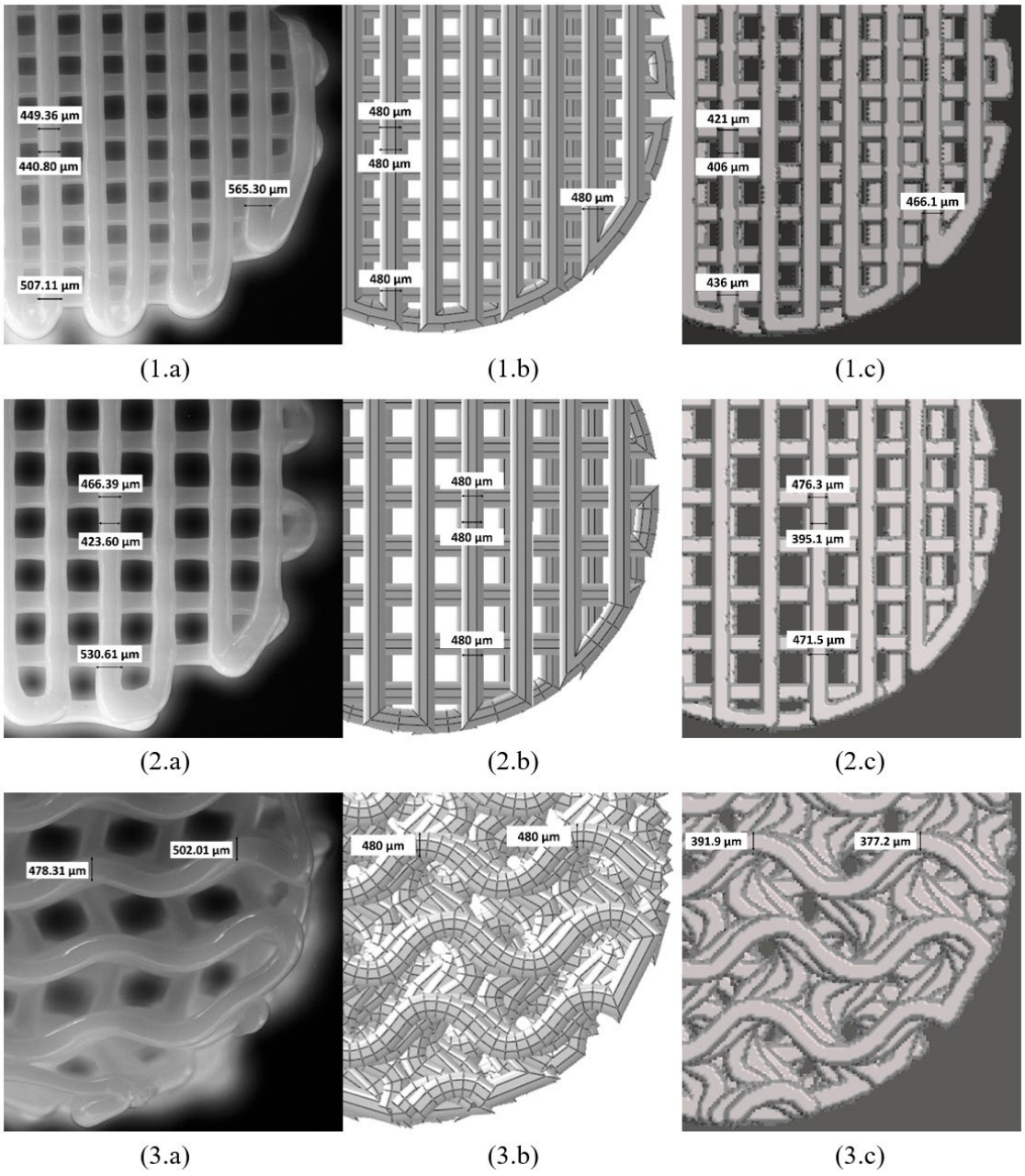


Figura 34. Comparación de ancho de filamentos de los scaffolds 50_rect (1), 60_rect (2) y 50_gyr (3): imágenes de microscopio de scaffolds impresos (a), modelo CAD (b) y modelo voxelizado (c)

5.3.5.3. Módulo de elasticidad equivalente

Los modelos correspondientes a las distintas configuraciones de scaffolds se simularon con Simulia Abaqus/CAE 6.14-1. A cada modelo se le aplicó un ensayo de compresión y se obtuvieron las fuerzas de reacción que aparecen en la capa base y, con los cálculos pertinentes, los módulos de Young. Los resultados de las simulaciones y de los ensayos experimentales se recogen en la Tabla 11.

Tabla 11. Comparación de módulos elásticos

Pieza	Módulo de Young experimental (MPa)	Metodología	Tipo de malla	Módulo de Young del modelo (MPa)	Desviación
	Er			Em	|Em-Er|/Er*100
50_rect	58.88	DECODE	C3D10	49.16	16.52%
			C3D4	85.19	44.68%
		VOLCO	C3D4	103.4	75.61%
60_rect	55.58	DECODE	C3D10	28.93	47.95%
			C3D4	40.56	27.03%
		VOLCO	C3D4	71.09	27.91%
50_gyr	39.21	DECODE	C3D10	-	-
			C3D4	11.88	69.70%
		VOLCO	C3D4	41.71	6.37%

Durante la etapa de simulación, no fue posible mallar el modelo CAD del scaffold 50_gyr, pero sí se pudo obtener un modelo 50_gyr mallado con elementos lineales, probando múltiples tamaños de semilla hasta encontrar el que no generara elementos distorsionados. El problema del mallado es uno de los más recurrentes en la metodología de modelado basado en geometría, especialmente en geometrías complejas. En estos casos, el modelado basado en vóxeles resulta más adecuado.

La comparación de los resultados de ambas técnicas de modelado muestra que el modelo basado en vóxeles es más rígido que el basado en la geometría. Esto se debe a que el modelado VOLCO tiene en cuenta la intersección de los filamentos, expandiendo el material en las uniones. Esto da lugar a mayores secciones de material en el contacto entre capas, lo que aumenta la rigidez de la pieza sometida a compresión en comparación con el modelado DECODE.

El tipo de malla utilizado también influye en la rigidez de la pieza, como puede observarse en los distintos modelos basados en la geometría. Los elementos lineales (C3D4) son menos precisos y, por tanto, tienden a aumentar la rigidez del modelo.

Otra conclusión que se puede extraer de los resultados del módulo elástico es la relacionada con la predicción de la rigidez de la pieza. Como se ha demostrado anteriormente y se presentó en la Tabla 10, existe una gran desviación dimensional entre las piezas impresas y los valores teóricos. Esto también provoca diferencias entre los resultados mecánicos simulados y los experimentales. Con el procedimiento actual, no es posible predecir el valor exacto del módulo de Young, tanto con la metodología de modelado VOLCO como DECODE, pero sí es posible comparar con precisión diferentes geometrías. En otras palabras, las variaciones del módulo no son

proporcionales entre los resultados experimentales y los de simulación, pero el modelo más rígido según las simulaciones se corresponde con la pieza impresa más rígida.

5.4. CONCLUSIONES

Este es el primer estudio que considera el modelado de las piezas impresas antes de su fabricación para aplicar el AEF y comparar qué técnica de modelado es más adecuada, práctica y eficiente para un caso concreto.

Una de las conclusiones que se pueden extraer de los resultados es que los modelos obtenidos no pueden considerarse representaciones exactas de las piezas impresas. Por otro lado, los modelos tampoco son precisos entre sí, ya que proporcionan resultados muy diferentes en función de la metodología o el tipo de malla utilizada. El modelado VOLCO proporciona modelos más rígidos debido a la expansión del material en las uniones. Sin embargo, el módulo elástico de los modelos sigue el mismo patrón que el experimental. En otras palabras, aunque no sean proporcionales, el modelo más rígido se corresponde con la pieza impresa más rígida en ambas metodologías. Por tanto, es posible determinar cuál será la geometría más rígida.

Aunque el modelado basado en la geometría es más rápido y más eficiente desde el punto de vista computacional en geometrías sencillas, también presenta más problemas de mallado que el basado en vóxeles. Estos problemas de mallado pueden resolverse reduciendo el tamaño de la semilla o aplicando la topología virtual, pero en determinados casos, esto tampoco funciona.

En relación con la precisión dimensional, las diferencias de volumen entre los modelos y la referencia teórica son insignificantes. En el modelo CAD, el volumen se aproxima mucho al previsto en el archivo de código G. En cambio, en el modelo voxelizado es necesario ajustar manualmente el volumen mediante la función spline.

Por todas estas razones, si el objetivo fuera encontrar la técnica de modelado más rápida y sencilla para diseñar y optimizar una pieza a imprimir con tecnología MEX, el nuevo modelado automático basado en geometría sería la opción recomendada. Sin embargo, para modelos pequeños y complejos, el modelado basado en vóxeles sería la opción más adecuada.

Por último, el modelado y el análisis de elementos finitos podrían ser los pasos previos a la optimización de piezas que pretenden imprimirse en 3D, reduciendo el trabajo experimental y mejorando la rentabilidad del proceso.

Capítulo 6. Metodología de optimización

En los capítulos anteriores se presentó la metodología de modelado 3D desarrollada (DECODE, basada en la creación de operaciones de barrido) y una comparación entre dicha metodología y la alternativa más prometedora encontrada en la literatura (VOLCO). Estas dos herramientas permiten modelar, a partir de un archivo G-code (cada una de ellas con sus ventajas e inconvenientes), una geometría bastante aproximada a la que realmente se imprimiría, por lo que es posible usar dichos modelos 3D para simular el comportamiento mecánico. Por tanto, la combinación de cualquiera de estas dos metodologías de modelado y el AEF podrían servir como herramientas de cálculo para optimizar el diseño de piezas impresas.

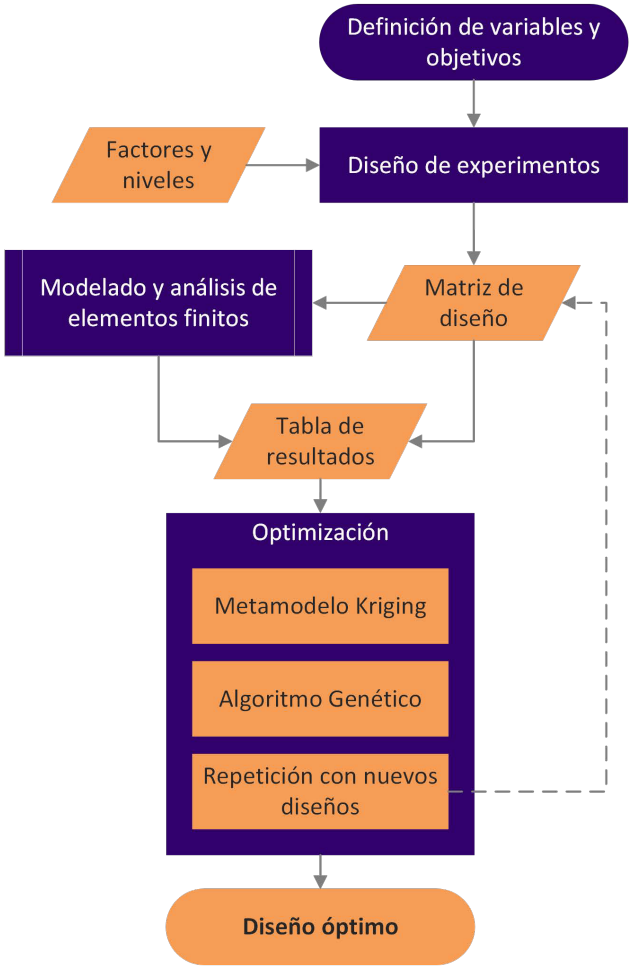


Figura 35. Proceso de optimización de piezas impresas en 3D mediante AG

Cabe destacar que, hasta el momento, se ha cubierto el primer objetivo específico de la tesis (metodología de modelado 3D de piezas impresas), por lo que

faltaría abordar el segundo objetivo, que se corresponde con establecer una metodología para la optimización de parámetros geométricos asociados a la fabricación MEX. Este capítulo se centra precisamente en este aspecto, es decir, en el proceso de optimización de parámetros geométricos de piezas obtenidas mediante MEX, antes de ser impresas, haciendo uso de algoritmos genéticos (AG). El proceso seguido es el que se muestra en la Figura 35.

En los siguientes subapartados se definen los criterios para el diseño de experimentos, se justifica la elección del metamodelo Kriging para la estimación de resultados y se describen los pasos para obtener el diseño óptimo mediante AG. Asimismo, se concretan las condiciones que debe cumplir un diseño para obtener tal designación.

6.1. DISEÑO DE EXPERIMENTOS

Con el objetivo de realizar un análisis riguroso de la influencia de los parámetros geométricos en el resultado de las piezas impresas en 3D, la metodología propuesta comienza con la aplicación de un diseño de experimentos. Estos experimentos se basan en los modelos obtenidos mediante las metodologías descritas anteriormente y datos teóricos obtenidos del archivo G-code que interactúan con los resultados, a partir de la retroalimentación con datos experimentales o de las simulaciones.

En el caso particular de esta investigación, se examinarán las propiedades geométricas de las piezas y se analizará el resultado de la combinación de varios de estos factores. Las variables de entrada y de respuesta se definirán específicamente para cada aplicación, pero en general, para la optimización de piezas impresas en 3D, las de entrada estarán relacionadas con el diseño geométrico (por ejemplo, altura de capa, porcentaje de relleno, porcentaje de relleno, número de perímetros, etc.) y la de salida, con el comportamiento mecánico, masa o, en el caso de aplicaciones en la ingeniería tisular, propiedades de carácter biológico (incluyendo aspectos geométricos como interconectividad, porosidad, etc.).

Para la selección de las variables de entrada, se escogerán aquellas con una influencia mayor en el resultado, atendiendo, siempre que sea posible, a la bibliografía disponible o el estudio estadístico de los datos. Teniendo en cuenta la naturaleza de los factores, se aplicará un diseño factorial, ya que este permitirá evaluar el efecto de múltiples variables independientes y sus interacciones en relación con los requisitos mecánicos y de masa de la pieza o cualquier otro resultado relacionado con la calidad

de la pieza impresa y el ahorro de recursos. Además, entre los distintos diseños de experimentos existentes, el diseño factorial tiene ciertas ventajas en la metodología de optimización propuesta. Esto se debe a que en el proceso de optimización que se describirá en los siguientes apartados, se emplea un metamodelo (Kriging) para estimar los resultados de diferentes diseños propuestos durante la optimización, todo ello en base a la información disponible del diseño de experimentos previo. Por tanto, si el diseño de experimentos es factorial, los datos disponibles se corresponderán con los valores extremos de cada variable, lo que implicará que el metamodelo realizará siempre interpolaciones (no extrapolaciones) para estimar los resultados, lo que implica una mayor precisión.

La fase más importante para cada aplicación es la de planificación, por lo que los primeros pasos serán la definición de objetivos, la selección de parámetros a optimizar y sus niveles, y determinación de los procesos a llevar a cabo para la medición.

Para cada variable de entrada (factor) se definirán al menos dos niveles (-1, +1), que representarán los valores extremos de dicha variable. En caso de asignar un número mayor de niveles, se considerarán las limitaciones de recursos y tiempo. En la matriz de diseño, se recogerán todas las combinaciones posibles de los niveles de los factores. Sin embargo, cuando el número de combinaciones sea demasiado grande, se puede utilizar un diseño factorial fraccionado para reducir la cantidad de experimentos requeridos. En la Figura 36 se presenta un ejemplo de diseño factorial cuyos factores no tienen la misma cantidad de niveles de prueba. El diseño factorial sería 2^2 x 3, es decir, los dos primeros factores con dos niveles y el último con tres.

	Experimento	Factor A	Factor B	Factor C
2^2	1	-1	-1	-1
	2	+1	-1	-1
	3	-1	+1	-1
	4	+1	+1	-1
2^2 x 3	5	-1	-1	0
	6	+1	-1	0
	7	-1	+1	0
	8	+1	+1	0
	9	-1	-1	+1
	10	+1	-1	+1
	11	-1	+1	+1
	12	+1	+1	+1

Figura 36. Ejemplo diseño factorial 2^2 x 3

Por otro lado, en el caso del análisis del comportamiento de las piezas mediante AEF, solamente se contempla el estudio de una réplica, ya que en cada repetición se

obtiene el mismo resultado. Mientras que, en los ensayos experimentales se requerirá la utilización de al menos tres réplicas, para evitar los errores por la variabilidad de resultados.

Una vez realizado el diseño de experimentos, se tendrá una tabla con toda la información relevante para cada uno de los diseños. es decir, los valores de las diferentes variables de diseño (asociadas a parámetros geométricos del laminador para fabricación MEX, tales como altura de capa, porcentaje de relleno, etc.) y las variables de respuesta relevantes para la optimización, que se obtienen del G-code y el modelado y AEF (por ejemplo, rigidez del modelo 3D, peso del mismo, etc.). Con esta información, es posible pasar a la siguiente etapa de la metodología.

6.2. METAMODELOS

La optimización del diseño, como se comentará en el siguiente apartado, se basa en el uso de algoritmos genéticos. Estos algoritmos emulan la teoría de evolución de las especies, de modo que los individuos más adaptados al medio son los que sobreviven y los que tienen descendencia, consiguiendo así que las sucesivas generaciones sean cada vez mejores. Para la correcta evolución y convergencia del algoritmo, es necesario generar muchos individuos por cada generación, y muchas generaciones, por lo que todos esos individuos (o diseños) propuestos deben ser evaluados para determinar su aptitud al medio. En el caso de la presente tesis, la evaluación de las variables de respuesta que afectan a la optimización puede ser costosa computacionalmente, por lo que es inviable calcular dichas respuestas para cada uno de los diseños propuestos internamente en el algoritmo genético. En este contexto, se propone el uso de metamodelos que permitan estimar los resultados de esos nuevos diseños a partir de la información disponible del diseño de experimentos. Por tanto, estos metamodelos representan el sistema (variables de respuesta) a partir de los datos recogidos durante el diseño de experimentos, permitiendo analizar y tomar decisiones sobre dicho sistema.

En estudio previos [77] se compararon diferentes métodos de estimación de resultados, como el método Kriging, la interpolación lineal basada en la triangulación de Delaunay y el ajuste por mínimos cuadrados con ecuaciones polinómicas de orden 2 y 3. Tras esta evaluación, se encontró que el método Kriging presentaba un error similar al método que había sido anteriormente considerado el mejor entre otros metamodelos, concretamente el de regresión con polinomios de orden 2. Sin embargo, el Kriging tiene la ventaja de ser un modelo interpolador, lo que significa que a medida

que se aumenta la cantidad de datos de muestra, las estimaciones se vuelven cada vez más precisas. Esto contrasta con el ajuste por mínimos cuadrados, que está limitado por la ecuación de ajuste. Debido a las ventajas presentadas por este método, se decidió utilizar el metamodelo Kriging [78].

El Kriging es un metamodelo ampliamente utilizado en la interpolación y extrapolación de resultados. Es conocido por ser estadísticamente preciso y proporcionar estimaciones lineales insesgadas de los valores de los puntos. Utiliza una combinación ponderada de los valores de las muestras conocidas para estimar la respuesta en un punto dado, minimizando la varianza del error de predicción. Esto significa que el Kriging se comporta como un método interpolador y coincide con los puntos de datos, siempre y cuando no haya múltiples mediciones en la misma ubicación [77].

El método Kriging consiste en ajustar coeficientes basados en la distribución espacial de los datos, minimizando la varianza y manteniendo un sesgo nulo. Esto se logra calculando el semivariograma, que representa la mitad de la varianza de los datos en función de la distancia entre ellos, y ajustando un modelo de correlación a estos datos [77].

Por otro lado, a pesar de que el método permite emplear diferentes modelos de correlación y de egresión polinómica, en este caso se usó un modelo de correlación exponencial y un orden de regresión polinómica variable en función de los datos disponibles, desde el orden 2 hasta el 0, como se explicará en el siguiente apartado.

En conclusión, se ha decidido utilizar el metamodelo Kriging debido a su capacidad para estimar los valores desconocidos utilizando los datos disponibles del diseño de experimentos, siendo un metamodelo con bastante potencial y capacidad de refinamiento a medida que se añaden nuevos datos. Esto resulta especialmente útil en el proceso de optimización, ya que permite obtener estimaciones precisas y confiables que ayudan en la toma de decisiones y mejora del diseño en general.

6.3. OPTIMIZACIÓN BASADA EN ALGORITMOS GENÉTICOS

La optimización se llevó a cabo mediante un algoritmo evolutivo, en concreto, un algoritmo genético. Para comenzar con el proceso de optimización mediante algoritmos genéticos, los valores de todas las variables, objetivos y restricciones deben obtenerse o calcularse con antelación.

Este proceso de optimización comienza con la generación de una población inicial aleatoria de 100 individuos y la evaluación de la función de aptitud para cada uno de ellos[78,79]. A continuación, se someten a la selección por torneo de dos individuos, el cruce entre los ganadores para generar nuevos descendientes, la mutación para añadir variabilidad a la nueva población y explorar nuevos resultados y la creación de la nueva generación, aplicando la reparación y el elitismo, para eliminar las soluciones que no cumplen con las restricciones y conservar las mejores soluciones de la generación anterior. Este proceso se repite hasta 100 generaciones, con 100 individuos cada una (Figura 37).

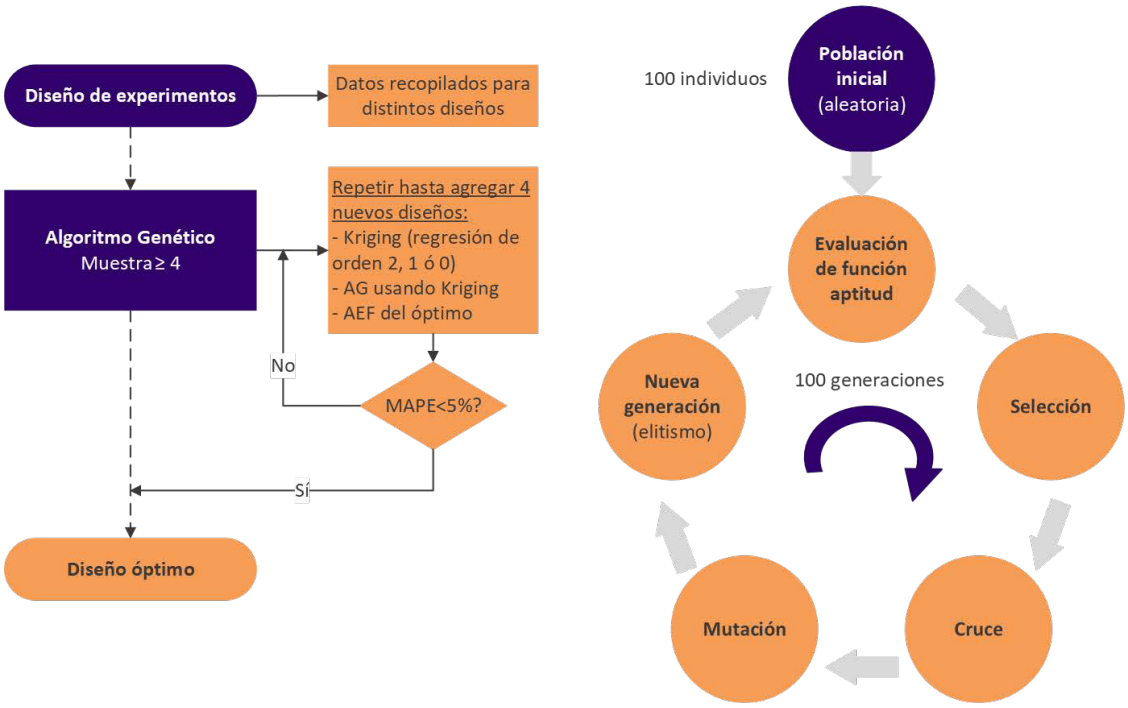


Figura 37. Proceso de optimización

La información obtenida a lo largo del diseño de experimentos se utiliza como base para los metamodelos. El desarrollo del metamodelo Kriging se puede simplificar utilizando un orden bajo, pero la precisión disminuye. Por lo tanto, se obtienen mejores estimaciones utilizando el modelo de regresión de orden 2, pero se necesitan más muestras o muestras distribuidas más uniformemente. Por ello, el algoritmo de optimización tiene un bucle que emplea primero un modelo de regresión de orden 2 para predecir cada respuesta e inmediatamente cambia a un modelo de regresión de orden 1 si el desarrollo del metamodelo no tiene éxito (y así sucesivamente hasta el orden 0). Como resultado, siempre se emplea el modelo de regresión óptimo basado en los datos disponibles [79,80].

Cabe destacar que, en este caso, en el proceso de cruce, el algoritmo genera un número aleatorio entre 0 y 1 para cada variable de diseño. Si el valor es menor o igual a 0.5 (probabilidad de cruce del 50%), el primer descendiente heredará el valor de la variable del primer progenitor, y el segundo, heredará el del segundo progenitor. Si el número generado es mayor a 0.5, el primer descendiente heredará el valor de la variable del segundo progenitor, y el segundo, heredará el del primer progenitor. De esta manera, se realiza un intercambio de información genética entre los padres y se obtienen dos descendientes con características mixtas.

Tras el cruce, se vuelve a generar un valor aleatorio para cada individuo de la población resultante (entre 0 y 1). El individuo asociado muta si este valor es inferior o igual a 0.6 (probabilidad de mutación del 60%). La mutación consiste en elegir una variable de diseño aleatoria y añadir el resultado de un valor aleatorio (entre -0.5 y 0.5), multiplicado por el intervalo máximo de esa variable de diseño, para alterarla ligeramente. Para las variables discretas, se aplica una operación de redondeo para obtener un valor entero. En caso de que la mutación provoque que los nuevos diseños se salgan del espacio de búsqueda definido, se realiza una reparación en dichas variables, estableciendo el valor límite. De esta manera, se crea la nueva generación, a la cual se le aplica elitismo para preservar al mejor individuo de la generación anterior.

La función de aptitud se define para cada caso concreto, decidiendo si se desea minimizar o maximizar los objetivos. Además, durante el proceso de optimización, si no se cumplen las restricciones, se aplica un factor de penalización que reduce el valor de aptitud.

En el proceso de optimización, el algoritmo genético se aplica a las muestras para obtener una configuración óptima teórica. Esta nueva configuración debe ser modelada y simulada, para introducirla de nuevo a la optimización y obtener una nueva configuración óptima. Si la configuración óptima resultante no se encuentra entre las muestras obtenidas con anterioridad, este proceso se repite, al menos, hasta obtener cuatro nuevos diseños (Figura 37).

Finalmente, para establecer un nuevo diseño como el óptimo, debe cumplir la condición de error porcentual absoluto medio (MAPE). El MAPE se calcula como se muestra en la Ecuación (5), donde A_t es el valor real (actual) de los resultados de las restricciones y los objetivos, F_t representa los valores previstos (forecast) y n es el número de resultados comparados. El MAPE debe ser inferior al 5% para validar el nuevo diseño.

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| < 5\% \quad (5)$$

6.4. CONCLUSIONES

En conclusión, en las aplicaciones prácticas de este trabajo se aplicará la metodología de optimización empleando un diseño de experimentos factorial inicial, seguido de la creación del metamodelo de aproximación Kriging y una optimización multiobjetivo mediante algoritmos genéticos.

Se comenzará con una matriz de diseño con todas las combinaciones de los niveles elegidos de los parámetros de diseño (factores), excepto cuando el número de experimentos sea demasiado alto. En este caso, se realizará un análisis previo de los experimentos que se descartarán y se evaluará su influencia en el estudio. Esto permitirá reducir la cantidad de experimentos necesarios, sin comprometer significativamente los resultados obtenidos.

Para llevar a cabo el proceso de optimización utilizando AG, se realizarán al menos cuatro repeticiones, siempre que sea factible. En cada repetición, se introducirá el diseño óptimo obtenido como un nuevo punto de datos para regenerar el metamodelo. Después de las cuatro iteraciones, se realizará una comprobación del valor del MAPE y el cumplimiento de las restricciones establecidas. Si el MAPE es satisfactorio y se cumplen las restricciones, se considerará que se ha alcanzado un diseño óptimo.

Capítulo 7. Aplicaciones prácticas

En este capítulo se describe la aplicación práctica de la metodología a dos casos: una probeta de tamaño reducido sometida a flexión y una estructura porosa para regeneración de tejido óseo. En ambos casos, se han analizado los parámetros de diseño a optimizar, se han modelado y simulado mediante AEF y se han fabricado para comprobar la validez del proceso de optimización.

En el caso de la probeta, se han estudiado los parámetros geométricos de impresión que influyen en las propiedades mecánicas, masa y tiempos de impresión de la pieza, para optimizarlos. Además, se han fabricado las piezas mediante MEX con el objetivo de comparar sus resultados experimentales con los simulados y deducir la validez de los resultados obtenidos en el AEF y la optimización.

Por otro lado, en el caso práctico del scaffold, el objetivo principal es obtener nuevos patrones de relleno con curvatura y compararlos con los disponibles en los laminadores comerciales, optimizando sus parámetros de diseño para obtener la configuración óptima para la proliferación celular y regeneración de tejido óseo, cumpliendo con los requerimientos mecánico. Además, se fabricarán y aplicará un ensayo biológico para verificar la predicción del comportamiento de las células.

7.1. MINIPROBETAS A FLEXIÓN

La primera aplicación práctica para la validación de las metodologías desarrolladas en esta tesis, incluyendo tanto la de modelado como la de optimización, es la de una miniprobeta (un octavo de la probeta estandarizada de la norma ISO 178 “Plásticos. Determinación de las propiedades de flexión”) de ácido poliláctico (PLA) impresa en 3D, que será sometida a flexión. Se ha utilizado una muestra reducida pero proporcional a la probeta estándar (todas sus dimensiones se han reducido en un factor de 2) para ahorrar en recursos computacionales y tiempo.

El objetivo de esta aplicación es modelar la pieza mediante DECODE y definir los parámetros de impresión óptimos, concretamente, la altura de capa, el porcentaje de relleno y el número de perímetros, para obtener una pieza impresa con un módulo de elasticidad en el rango deseado, lo más ligera posible y cuyo tiempo de impresión sea mínimo. Mientras que otros parámetros, como el patrón de relleno y orientación del filamento se mantienen constantes.

Además, para comparar los resultados de las simulaciones con la realidad, se

procedió a la impresión y ensayo de la miniprobetas, incluyendo los resultados experimentales a la optimización, para obtener un factor de corrección de los módulos de Young del modelo y la pieza real.

7.1.1. Selección de parámetros a optimizar

En este trabajo, se ha decido optimizar los parámetros de impresión relacionados con la geometría, es decir, aquellos que afectan a la trayectoria de los filamentos. Por tanto, se han excluido parámetros de fabricación, como la temperatura o velocidad de impresión, que en ciertas condiciones podrían modificar dimensionalmente la sección del filamento extruido, pero que no son el objeto de esta tesis doctoral.

Según A. Alafaghani et al. [40], los parámetros que más afectan a las propiedades mecánicas de una pieza impresa en 3D son la dirección de construcción y la altura de capa. Aunque, concluye que la dirección de construcción adecuada será aquella que permita que las capas y las direcciones de las cargas se sitúen en el mismo plano. Asimismo, considera que el patrón de relleno tiene una influencia menos significativa cuanto mayor es el porcentaje de relleno o menor las dimensiones de la pieza. Por lo tanto, este estudio sirve como base para determinar que la dirección de la impresión será a lo largo del eje Z, quedando la parte de la probeta sometida a flexión en el plano XY (Figura 38).

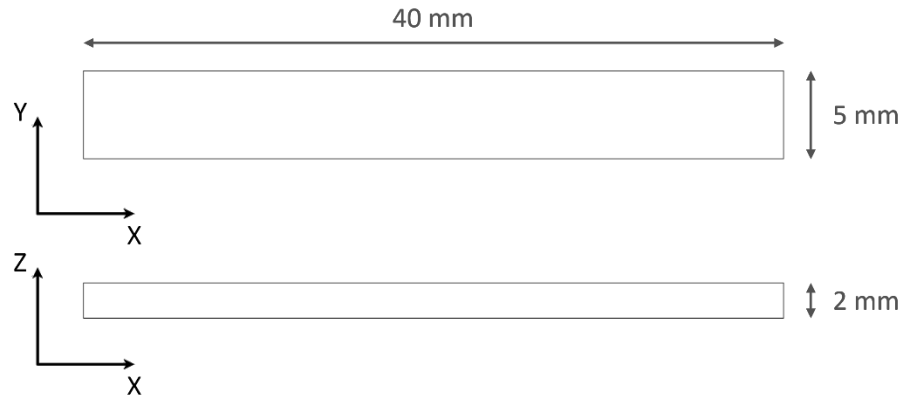


Figura 38. Orientación de miniprobeta en laminador y dimensiones (1/8 de la probeta estándar de la ISO 178)

En algunos estudios a tracción, se afirma que cuando la orientación de las capas coinciden con la dirección de la carga de tracción, el aumento del ángulo de trama mejora la resistencia a la tracción, aunque de manera no muy significativa [48]. Otros autores han estudiado la influencia de este ángulo en la resistencia a flexión. O.S. Es-Said et al. [81] afirman que el ángulo de trama de 0° mostró una mayor resistencia a flexión, en comparación con la de 45/-45° y 90° en probetas orientadas a 0° (probeta

orientada longitudinalmente al eje x de impresión). Además, O.A. Mohamed et al. [49] han determinado que el ángulo de la trama muestra una baja influencia en el módulo de almacenamiento, la pérdida y la amortiguación mecánica, es decir, en el rendimiento mecánico-dinámico de piezas impresas. No obstante, en la optimización realizada, han determinado que la configuración óptima presenta un ángulo de trama de 0°. Por tanto, tomando los resultados obtenidos por otros autores para probetas sometidas a flexión, el ángulo de trama de las piezas será 0°.

Según A.H. Nickel et al. [82], el patrón utilizado para depositar un material en una capa tiene un importante efecto sobre las tensiones y deformaciones resultantes. Sin embargo, pocos autores consideran el patrón de relleno como un parámetro relevante en los resultados de las propiedades mecánicas. A. Alafaghani et al. [40] afirman que el patrón de relleno afecta en menor medida que otros parámetros, debido a que su relevancia depende del porcentaje de relleno utilizado, es decir, a mayor porcentaje de relleno menor importancia tiene el patrón. Por tanto, se ha decidido tomar el mismo patrón de relleno para todas las muestras, teniendo en cuenta los resultados obtenidos por B. Akhoundi et al. [83]. Concretamente, en su estudio examinaron el efecto de diferentes patrones y porcentajes de relleno sobre las resistencias a la tracción y a la flexión. Los resultados mostraron que el patrón concéntrico dio lugar a las propiedades mecánicas más elevadas. Con lo cual, todas las miniprobetas tendrán un patrón de relleno concéntrico. En este patrón las capas de material se disponen de manera concéntrica alrededor de un eje central, con lo cual el ángulo de trama solamente afectaría a las capas sólidas aplicadas a la base y a la cara superior, cuyo relleno sigue un patrón rectilíneo.

Habiendo definido los parámetros que se mantendrán fijos (orientación de impresión, ángulo de trama y patrón de relleno), se estudiará la influencia del porcentaje de relleno, número de perímetros y altura de capa en las miniprobetas y se obtendrá la configuración óptima para la mejora de sus propiedades mecánicas, con el menor peso y tiempo de impresión. Concretamente, se estudiarán porcentajes de relleno del 20, 50, 70 y 100%, 1 y 3 perímetros, y 0.15 y 0.3 mm de altura de capa (se aportarán más detalles en el apartado 6.1 de diseño de experimentos). Cabe destacar que, en este caso, el número de perímetros no sólo se refiere a las capas de paredes laterales exteriores de la pieza, sino que también se añade ese número de capas sólidas en la base y la cara superior, es decir, en el laminador se modificarán “perimeters” y “solid layers” (“top” y “bottom”).

7.1.2. Aplicación de la metodología

Con el fin de obtener los parámetros geométricos óptimos para la generación del G-code de una miniprobeta de ensayo, consiguiendo que la pieza impresa tenga un módulo de Young superior a 2200 MPa (módulo de elasticidad equivalente, puesto que se considera en el cálculo que la probeta está maciza), el mínimo peso y que se imprima en el mínimo tiempo, se han aplicado las metodologías de modelado y optimización desarrolladas en este trabajo. Concretamente, la metodología aplicada es la DECODE, basada en geometría, ya que, aunque podría modelarse con VOLCO, se trataría de un proceso menos eficiente.

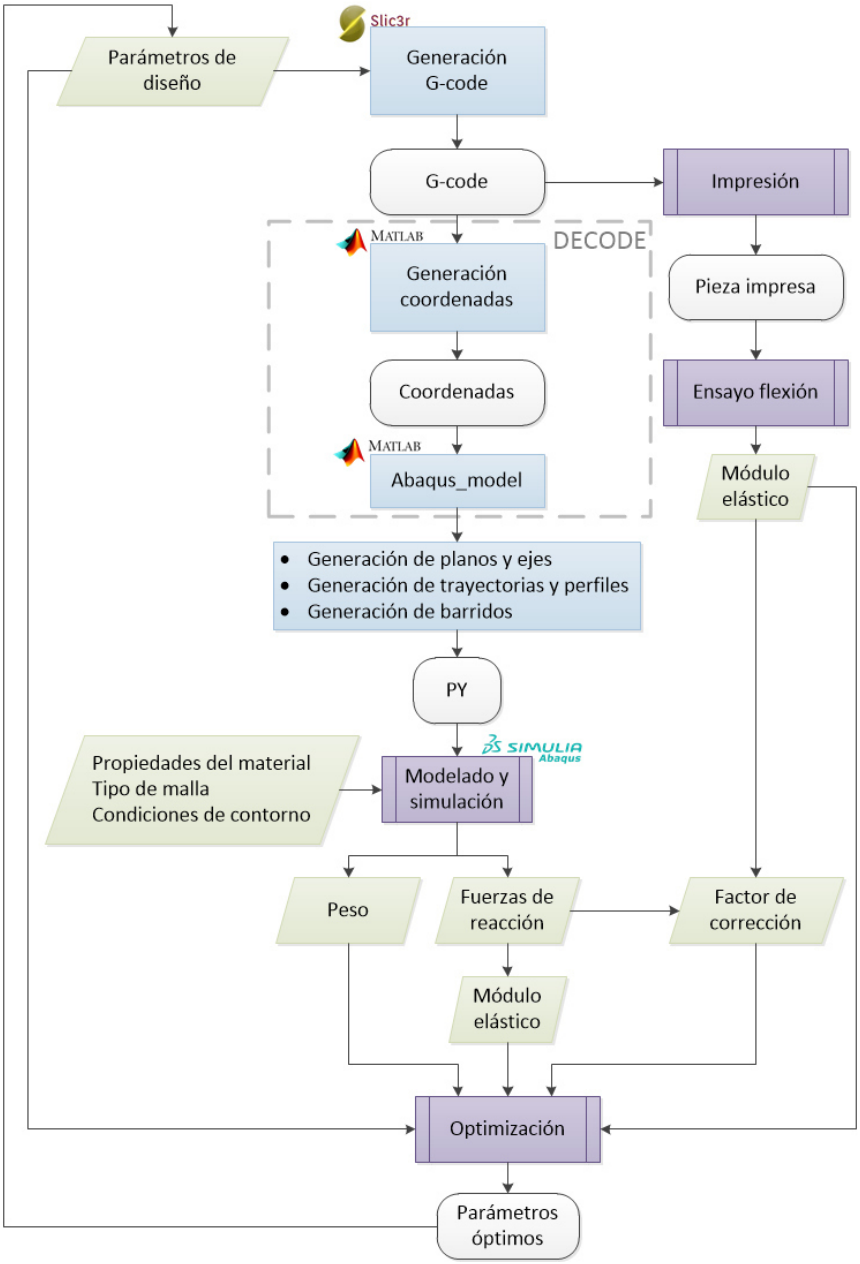


Figura 39. Metodología para el modelado y optimización de miniprobetas

Para el desarrollo de todo el proceso se han seguido los pasos presentados en el diagrama de la Figura 39. Una vez seleccionados los parámetros de diseño, se genera el G-code correspondiente a cada diseño. Con ese archivo, se procede a generar el modelo 3D con la metodología DECODE, para posteriormente realizar la simulación del ensayo de flexión y determinar el módulo de elasticidad equivalente a partir de las fuerzas de reacción obtenidas en la simulación y desplazamiento aplicado. Paralelamente, también se fabricaron y ensayaron a flexión las diferentes probetas del diseño de experimentos. Los valores obtenidos, tanto de las simulaciones como de los ensayos reales, se emplearon como base de datos inicial para realizar la optimización. En los siguientes apartados se detallan las diferentes etapas de la metodología.

7.1.2.1. Material

Las miniprobetas serán impresas en PLA Smartfil (Smart Materials 3D Printing S.L., Alcalá la Real, España), cuyas propiedades se muestran en la Tabla 12.

Tabla 12. Propiedades de PLA Smartfil

Material Properties of PLA – Smartfil	
Diámetro	1.75 mm
Densidad	1.24 g/cm³
Resistencia a tracción	114 MPa
Alargamiento a la rotura	100 %
Módulo de elasticidad	3861 MPa
Temperatura de impresión	220 ± 20 °C
Temperatura de la cama	0 - 60 °C
Temperatura de deflexión térmica	65 °C

Para simplificar el proceso de aplicación de la metodología de modelado y AEF, se tomarán los datos de catálogo para la definición de las propiedades del material, aunque otros autores han optado por ensayar el material para definir sus propiedades [6,25].

7.1.2.2. Diseño de experimentos

Los parámetros geométricos de impresión seleccionados para su estudio, serán las variables a introducir en el proceso de optimización mediante algoritmos genéticos. Entre ellas, dos son continuas (altura de capa y porcentaje de relleno) y una, discreta (n° de perímetros). En este caso, se han tomado los puntos mínimo y máximo del intervalo de alturas de capa (0.15 y 0.3 mm), los puntos mínimo y máximo y otros dos puntos dentro del intervalo del porcentaje de relleno (20, 50, 70 y 100 %), y el número

mínimo y máximo de perímetros (1 y 3). Aplicando un diseño de experimentos factorial (todas las combinaciones posibles con los niveles anteriores) se obtienen 16 combinaciones (2x4x2), es decir, 16 muestras para la optimización. La configuración de cada una de las muestras se presenta en la Tabla 13.

Tabla 13. Valor de variables para cada configuración de miniprobeta

Miniprobeta	Parámetro geométrico		
	Altura de capa (mm)	Porcentaje de relleno (%)	N° de perímetros
Muestra	VAR1	VAR2	VAR3
1	0.3	20	1
2	0.3	20	3
3	0.3	50	1
4	0.3	50	3
5	0.3	70	1
6	0.3	70	3
7	0.3	100	1
8	0.3	100	3
9	0.15	20	1
10	0.15	20	3
11	0.15	50	1
12	0.15	50	3
13	0.15	70	1
14	0.15	70	3
15	0.15	100	1
16	0.15	100	3

Las variables dependientes a estudiar serán los resultados del módulo de elasticidad equivalente calculado en las simulaciones y el obtenido en los ensayos experimentales (MPa), el peso de los modelos (g) y los tiempos de impresión teóricos (s), dados por el G-code.

7.1.2.3. Modelado y Análisis de Elementos Finitos

Para el modelado de las miniprobetas se han seguido los pasos a de la metodología DECODE [71]. Es decir, se comienza con la generación del código G con comentarios en un laminador gratuito y de código abierto, Slic3r 1.3.0.

Para generar este archivo es imprescindible contar con el modelo sólido de la pieza que se desea imprimir. En este caso, el STL de una miniprobeta con las medidas y orientación presentadas anteriormente, en la Figura 38. A continuación, será necesario especificar la configuración de la impresión y el filamento.

En este caso práctico se ha decido mantener la temperatura de impresión y de

la cama, el ancho de extrusión, el patrón de relleno y el ángulo de trama. Por otro lado, se tomarán como variable para la optimización el porcentaje de relleno, la altura de capa y el número de perímetros. Los valores asignados se muestran en la Tabla 14.

Tabla 14. Configuración de impresión miniprobetas

Configuración de impresión miniprobeta	
Temperatura de impresión	200 °C
Temperatura de la cama	0 °C
Ancho de extrusión	0.48 mm
Patrón de relleno	Concéntrico
Ángulo de trama	0°
Porcentaje de relleno	20, 50, 70 100 %
Altura de capa	0.15, 0.3 mm
N° perímetros	1, 3

A continuación, se introduce en la subrutina Abaqus_model, que detecta el software de origen y genera un archivo Python con la información para la generación de planos, ejes de referencia, trayectorias, perfiles y barridos, a partir de las coordenadas que extrae del G-code. El modelo se obtiene al ejecutar el archivo Python en Abaqus/CAE 6.14-1.

Al disponer del modelo dentro del software de simulación, solamente habrá que seguir los pasos del AEF especificados en el apartado 5.2 y mostrados en el Anexo G. Concretamente, es necesario especificar y atribuirle las propiedades del PLA, mallar con tetraedro cuadrático de 10 nodos (C3D10) y tamaño de semilla máximo de 1 mm, crear el ensamblaje para el ensayo a flexión de tres puntos, añadir un paso a la simulación, detallar las interacciones entre superficies, especificar las condiciones de contorno, ejecutar la simulación e interpretar los resultados.

Para el ensayo de flexión de tres puntos es necesario modelar tres piezas cilíndricas rígidas, de altura y radio 5 mm, que representaran los dos apoyos y la cruceta estándar (según la norma ISO 178:2020) de las que se disponen para los ensayos experimentales, como se mostraba en la Figura 17. Por ello, las interacciones entre estos cilindros y la miniprobeta serán del tipo superficie-superficie, con un coeficiente de fricción tangencial de 0.15 y un comportamiento normal de tipo contacto rígido. Los soportes se encuentran encastrados con una distancia entre ellos de 32 mm (según la normal, debe ser igual a 16 multiplicado por el espesor de la pieza) y centrados con respecto a la miniprobeta, mientras que, a la cruceta, que se encuentra en el centro de la miniprobeta, se le aplica desplazamiento vertical

descendente de 1 mm. En la Figura 40 se muestra el resultado del desplazamiento de la miniprobeta 7 en el Paso-1.

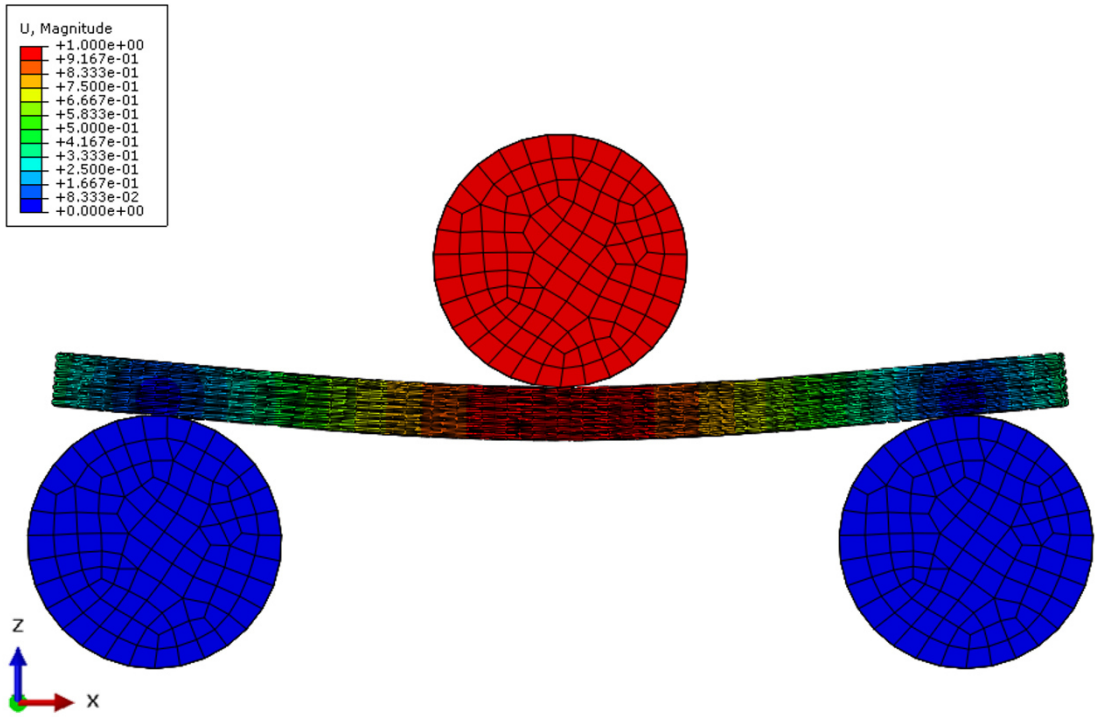


Figura 40. Resultado de desplazamiento de la miniprobeta 7 en Paso-1

Para la determinación del módulo de Young equivalente, es necesario obtener la fuerza de reacción generada por ese desplazamiento de 1 mm hacia abajo que realiza la cruceta y aplicar la Ecuación (6). Esta fuerza de reacción vertical se mide en un punto de referencia situado en el centro de la circunferencia de la cruceta.

$$E = \frac{\sigma}{\varepsilon} = \frac{3FL/2bh^2}{6sh/L^2} = \frac{3FL^3}{12sbh^3} \qquad (6)$$

donde

E → Módulo de Young o módulo de elasticidad (MPa)

σ → Esfuerzo de flexión (MPa)

ε → Deformación

F → Fuerza de reacción (N)

L → Distancia entre apoyos (mm)

b → Ancho de la probeta (mm)

h → Espesor de la probeta (mm)

s → Flecha (mm)

Los resultados de las fuerzas y módulos para cada miniprobeta se presentan en la Tabla 15.

Tabla 15. Resultados del AEF de las miniprobetas

Miniprobeta	Resultado AEF	
	Fuerza de reacción (N)	Módulo de Young (MPa)
1	13.49	2386.53
2	18.64	3297.68
3	15.33	2712.50
4	18.64	3297.59
5	16.76	2965.79
6	18.81	3328.12
7	17.96	3176.99
8	18.80	3326.51
9	9.95	2197.47
10	16.30	3602.55
11	12.15	2685.10
12	16.31	3602.97
13	14.31	3162.46
14	17.22	3804.89
15	16.10	3558.09
16	17.01	3759.21

7.1.2.4. Factor de corrección

Como se concluyó en el caso práctico para la comparación de metodologías de modelado (apartado 5.3.5.3), no es posible predecir el valor exacto del módulo de elasticidad de la pieza impresa. Es por ello por lo que se ha añadido a la optimización el factor de corrección entre el módulo de Young equivalente del modelo y la pieza impresa (E_{real}/E_{modelo}). De modo que, si se multiplica el módulo obtenido en las simulaciones por este factor de corrección, se podría predecir el resultado de las piezas impresas.

Con el objetivo de obtener una predicción precisa del factor de corrección para las distintas configuraciones, por un lado, se simularon los modelos y, por otro, se imprimieron y ensayaron las miniprobetas; con los resultados de los módulos de elasticidad, se obtuvieron sus factores de corrección y se introdujeron en la optimización. De esta manera, se genera un valor estimado de este factor en los nuevos diseños resultantes del proceso de optimización.

7.1.2.5. Impresión 3D y ensayo a flexión

Con el objetivo de comparar las dimensiones, pesos, tiempos de impresión y módulos de elasticidad teóricos (del modelo) con los reales, se decidió imprimir 5 réplicas de cada configuración, para su medición y ensayo experimental.

Para ello, las piezas se fabricaron con PLA Smartfil de 1.75 mm de diámetro, en la impresora 3D Ender-3 (Shenzhen Creality 3D Technology Co., Ltd., China), sin cama caliente, temperatura de boquilla de 200 °C y ancho de extrusión de 0.48 mm. El resultado de la impresión se presenta en la Figura 41.

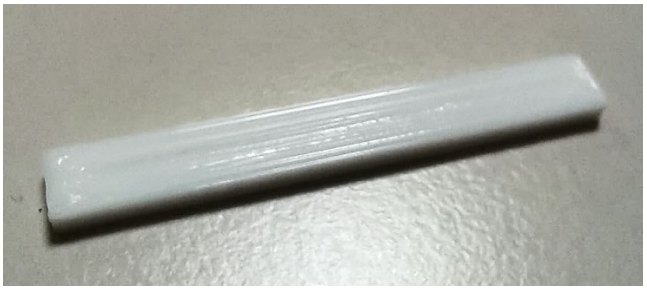


Figura 41. Miniprobeta impresa

Todas las piezas se midieron, se pesaron y se ensayaron a flexión en una máquina de ensayo LIYI, en el modo de control de desplazamiento. La distancia entre soporte se fijó en 32 mm, situándose la cruceta en el centro de estos (a 16 mm de cada lado) y la miniprobeta centrada (Figura 42). Los radios, tanto de los soportes como de la cruceta, son de 5 mm. La velocidad de la cruceta se fijó en 1 mm/min y el módulo de elasticidad se calculó como la pendiente del segmento inicial (deformación entre 0.05 y 0.25%) en el gráfico tensión-deformación resultante.

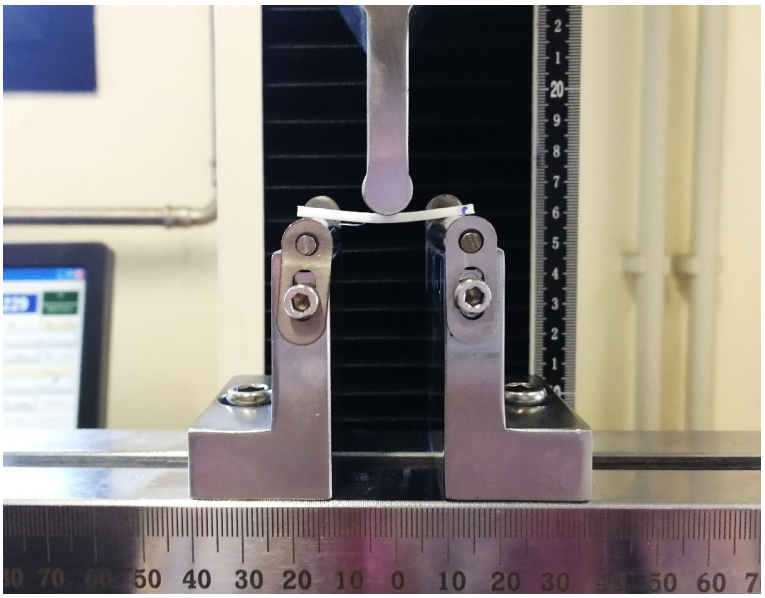


Figura 42. Ensayo a flexión miniprobeta

Los resultados del módulo de Young equivalente medio y desviación típica muestral de todas las réplicas, de cada miniprobeta, se muestran en la Tabla 16.

Tabla 16. Resultados experimentales del módulo de Young de las miniprobetas

Miniprobeta	Módulo de Young medio y desviación típica (MPa)
1	1447.44 ± 35.30
2	2401.44 ± 69.41
3	1758.88 ± 110.21
4	2537.84 ± 58.59
5	2231.07 ± 108.02
6	2633.34 ± 57.26
7	2423.22 ± 79.64
8	2576.28 ± 71.00
9	1384.23 ± 63.99
10	2497.24 ± 70.75
11	1936.52 ± 109.94
12	2562.86 ± 53.66
13	2238.52 ± 97.91
14	2703.72 ± 137.27
15	2574.67 ± 56.58
16	2702.41 ± 64.93

Las mediciones de las dimensiones de las miniprobetas impresas se compararon con los modelos obtenidos para comprobar la desviación dimensional de la longitud, ancho y espesor. La longitud y ancho de los modelos obtenidos presentan el mismo valor que el diseño inicial (40 x 5 mm), en cambio, el espesor varía según la altura de capa especificada. Para alturas de capa de 0.3 mm, el espesor del modelo es 2.1 mm (7 capas), mientras que para las de 0.15 mm, es 1.95 mm (13 capas). En la Figura 43, se muestra la comparación entre estos valores y la longitud, ancho y espesor medios de cada una de las configuraciones de las miniprobetas. A partir de esta comparación, se ha deducido que no existen grandes discrepancias y, por lo tanto, estas desviaciones no afectarán en gran medida a las simulaciones del comportamiento mecánico de la pieza.

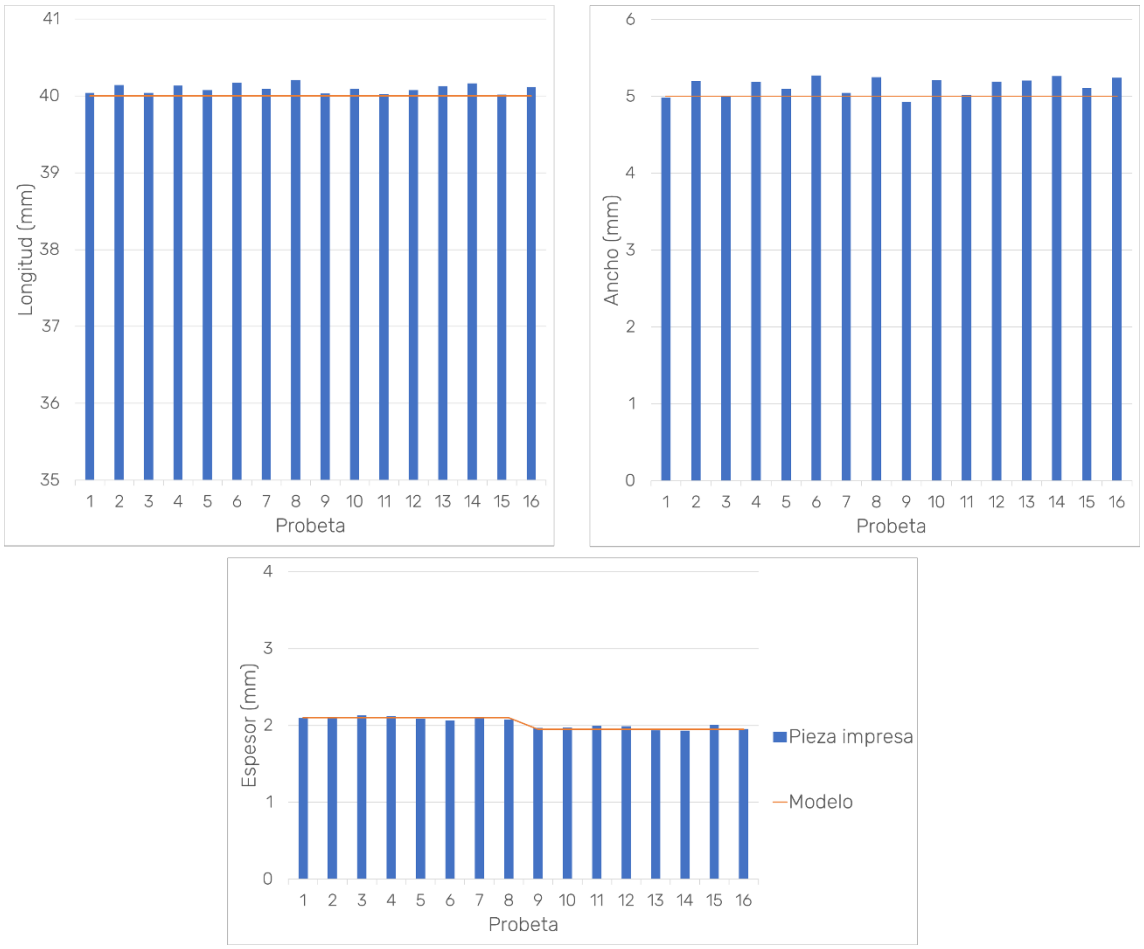


Figura 43. Comparación dimensiones miniprobetas impresas y modelos

7.1.2.6. Estudio estadístico

En paralelo al proceso de optimización, se llevó a cabo un estudio estadístico para así obtener conclusiones adicionales sobre la influencia de los parámetros geométricos de impresión en el comportamiento mecánico. Para ello se han estudiado los resultados experimentales y de las simulaciones por separado.

- *Estudio de datos experimentales*

Para el estudio estadístico de los datos experimentales, se han analizado los módulos elásticos equivalentes de todas las réplicas de cada miniprobeta (todos los datos se encuentran en el Anexo I).

En primer lugar, se aplicó el test de normalidad Kolmogorov-Smirnov a los datos para comprobar si es aplicable un test paramétrico. En la primera parte del test, se ha comprobado si los datos siguen una distribución Gaussiana estándar N(0,1), mientras que, en la segunda, se ha verificado si los datos siguen una distribución normal N(m,sigma). El resultado de dicho test ha rechazado ambas hipótesis nulas (h=1) con evidencia significativa (p=0.0029), por lo tanto, los datos no siguen una distribución

normal. Su distribución se muestra en la Figura 44.

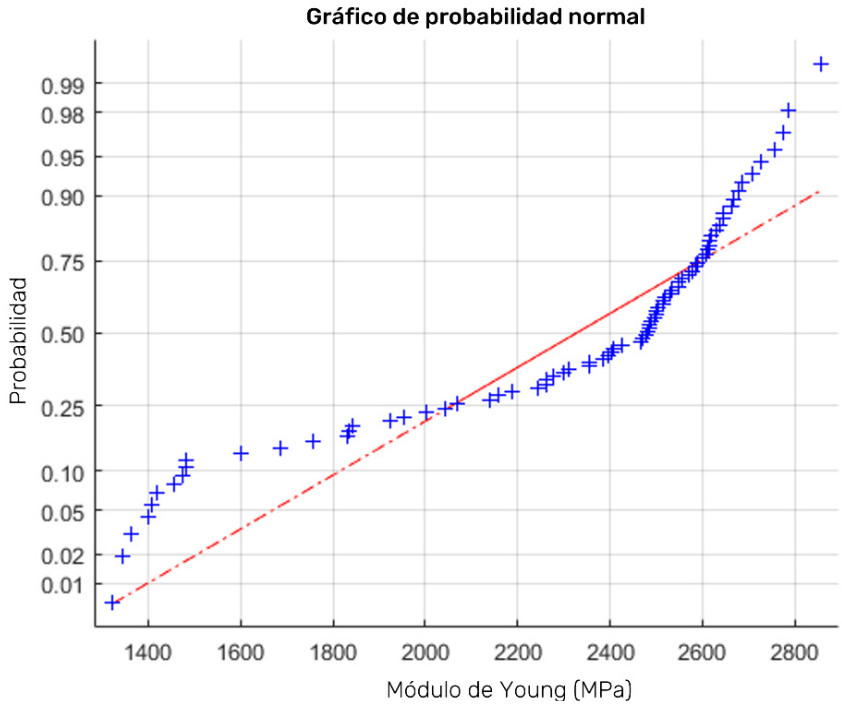


Figura 44. Análisis distribución de datos experimentales de las miniprobetas

Por lo tanto, después de obtener los resultados del test de distribución, se procede a realizar el test no paramétrico de Kruskal-Wallis para determinar si hay diferencias significativas entre los grupos de estudio (cada configuración representa un grupo). En base a los resultados del test de Kruskal Wallis, donde el valor de p ha resultado ser $1.5296 \cdot 10^{-9} < 0.01$ (99%), se puede concluir que existe evidencia estadística para rechazar la hipótesis nula, es decir, los factores considerados en el análisis tienen un impacto significativo en el resultado. En la Figura 45 se muestra la gráfica de caja y bigotes con la distribución de los resultados de los grupos de miniprobetas.

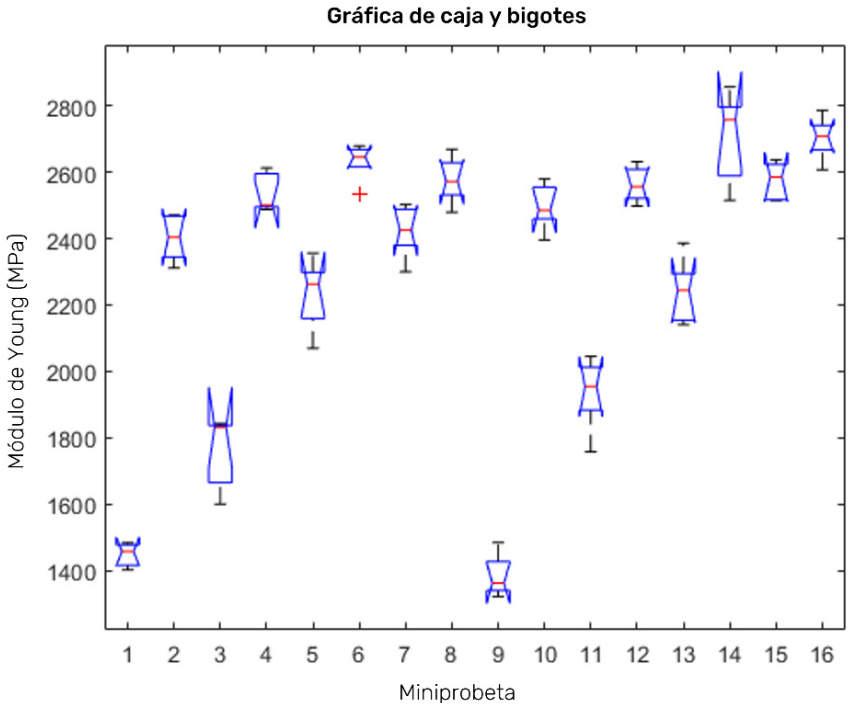


Figura 45. Gráfica de caja y bigotes de datos experimentales de las miniprobetas

A continuación, se ha utilizado un test multicomparación para determinar qué pares de miniprobetas presentan diferencias significativas. En la Tabla 17 se muestra un resumen de las miniprobetas que presentan diferencias significativas.

Tabla 17. Resumen resultados test multicomparación

Miniprobeta	Configuración			Miniprobetas con diferencias significativas
	Altura de capa (mm)	Porcentaje de relleno (%)	N° perímetros	
1	0.3	20	1	6, 8, 14, 15, 16
2	0.3	20	3	-
3	0.3	50	1	6, 14, 16
4	0.3	50	3	-
5	0.3	70	1	-
6	0.3	70	3	1, 3, 9
7	0.3	100	1	-
8	0.3	100	3	1, 9
9	0.15	20	1	6, 8, 12, 14, 15, 16
10	0.15	20	3	-
11	0.15	50	1	14, 16
12	0.15	50	3	9
13	0.15	70	1	-
14	0.15	70	3	1, 3, 9, 11
15	0.15	100	1	1, 9
16	0.15	100	3	1, 3, 9, 11

Si los grupos se agrupan por pares con igual porcentaje de relleno y número de perímetros, puede analizarse si la altura de capa afecta a la diferencia significativa en los resultados. En este caso, los pares de datos a comparar son 1-9, 2-10, 3-11, 4-12, 5-13, 6-14, 7-15 y 8-16 (Figura 46). Ninguno de ellos presenta una diferencia significativa con su pareja, así que, en principio, la altura de capa no tendría una gran influencia en la diferencia de resultados.

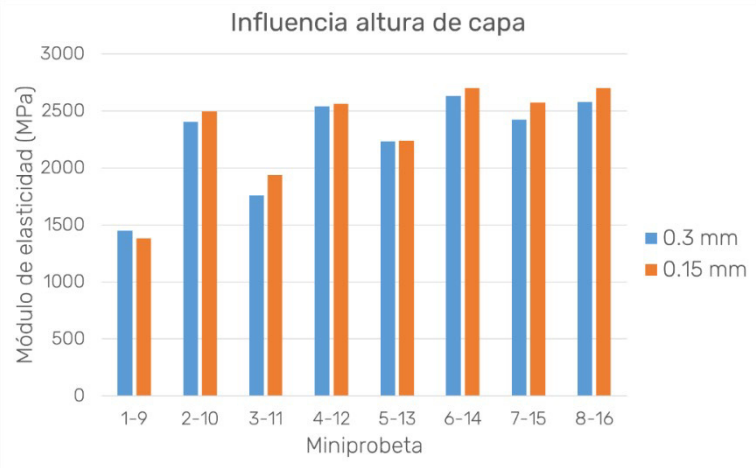


Figura 46. Influencia de la altura de capa en la rigidez de la miniprobeta

Para el análisis de la influencia del porcentaje de relleno, se han comparado 1-3-5-7, 2-4-6-8, 9-11-13-15 y 10-12-14-16 (Figura 47). En estos grupos, se ha detectado que entre la miniprobeta 9 y la 15 existe una diferencia significativa y su configuración solamente difiere en el porcentaje de relleno, 20% y 100%, respectivamente. Sin embargo, no podría afirmarse que este parámetro sea el único que influya.

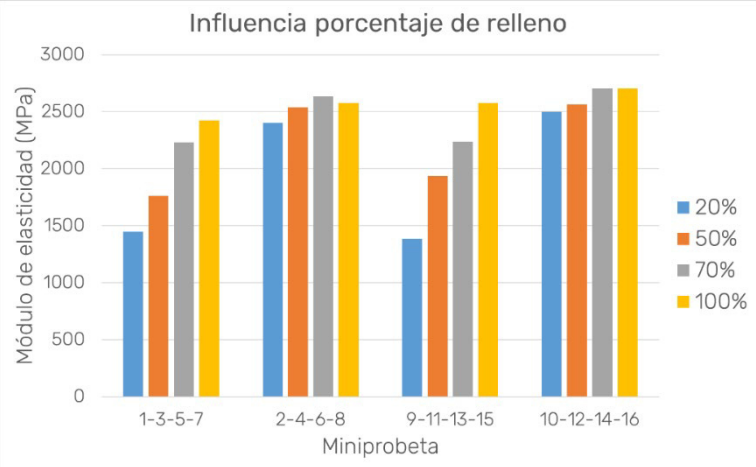


Figura 47. Influencia del porcentaje de relleno en la rigidez de la miniprobeta

Por último, para analizar la relevancia del número de perímetros en el resultado, se han estudiado los pares 1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16 (Figura 48). Este parámetro no parece tener influencia por sí solo en la diferenciación de resultados.

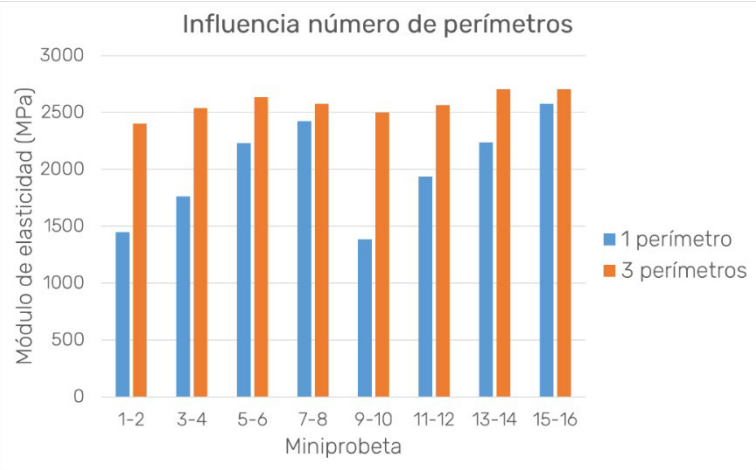


Figura 48. Influencia del número de perímetros en la rigidez de la miniprobeta

De este análisis se deduce que las diferencias significativas no se deben a la variación de un solo parámetro, sino a una combinación de los tres. Para analizar esto en más profundidad y tener una visión general de la influencia de las variables, se ha recurrido a aplicar un análisis de varianza (ANOVA), aunque los resultados no sigan una distribución normal. El test ANOVA multifactorial con un nivel de significancia del 95% ha generado los resultados mostrados en la Tabla 18.

Tabla 18. Resultados test ANOVA multifactorial para datos experimentales

Grupo	Valor p	Diferencia significativa (p<0.05)
Altura de capa (g1)	0.0003	Sí
Porcentaje de relleno (g2)	0.0000	Sí
N° de perímetros (g3)	0.0000	Sí
g1*g2	0.1041	No
g1*g3	0.7762	No
g2*g3	0.0000	Sí

Por lo tanto, ANOVA concluye que cada uno de los parámetros individuales (altura de capa, porcentaje de relleno y número de perímetros) tienen un efecto significativo en la variable dependiente. Al igual que la interacción entre el porcentaje de relleno y número de perímetros. Sin embargo, las combinaciones altura de capa y porcentaje de relleno, y altura de capa y número de perímetros, no tienen un efecto conjunto significativo en la variable dependiente.

Además, teniendo en cuenta los datos presentados y la variación de los resultados en función de la selección de parámetros se obtienen otras conclusiones como:

- El número de perímetros y el porcentaje de relleno afectan más a la rigidez de la pieza que la altura de capa.

- A alturas de capa más pequeñas, la rigidez será mayor. Ya que, cuanto menor es la altura, mayor es el ancho de filamento que interacciona con la capa adyacente y, por tanto, la adherencia entre ellas (Figura 49.a). Sin embargo, no supondrá una gran diferencia en los resultados (Figura 46).
- Cuanto mayor es el número de perímetros, la variación del porcentaje de relleno tiene menos influencia en los resultados (Figura 49.b). Mientras que, para densidades de relleno más altas, menos diferencia habrá al variar el número de perímetros (Figura 49.c).

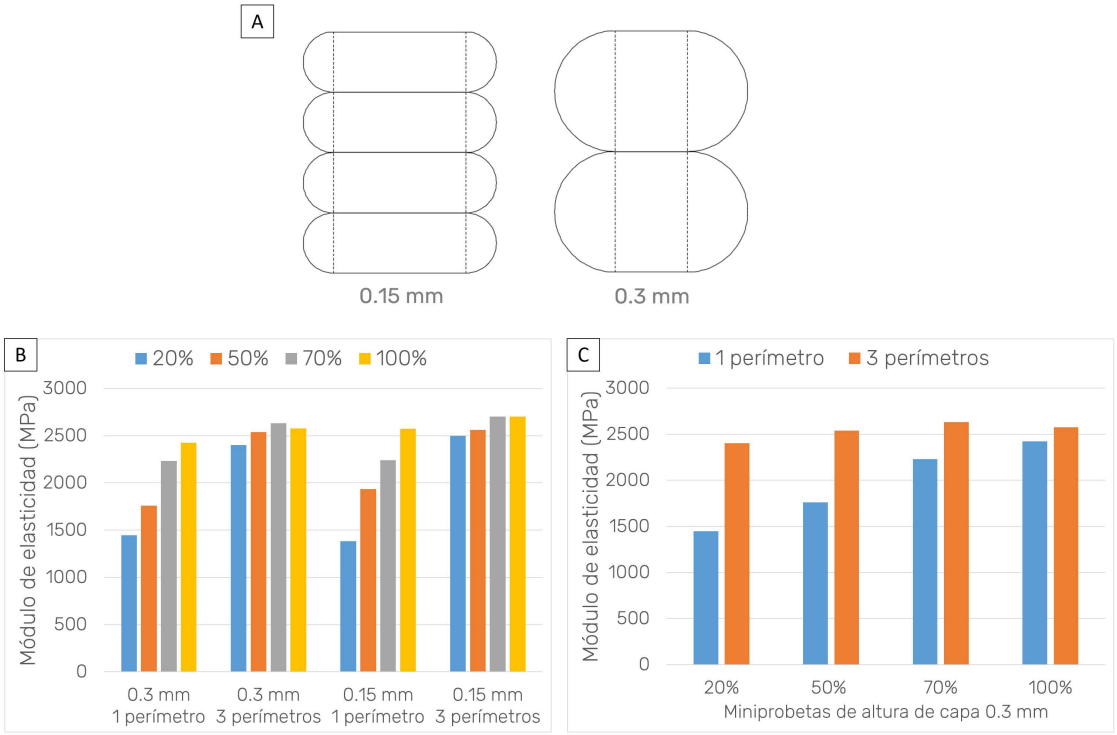


Figura 49. Influencia de parámetros geométricos: (A) Diferencia de deposición según la altura de capa. (B) Influencia del porcentaje de relleno según el número de perímetros. (C) Influencia del número de perímetros según el porcentaje de relleno.

- *Estudio de datos de simulación*

A diferencia de los estudiados para las miniprobetas impresas, los módulos elásticos resultantes de las simulaciones de las probetas sí siguen una distribución normal N(m,sigma), según el test de Kolmogorov-Smirnov (h=0). La distribución de los datos se muestra en la Figura 50.

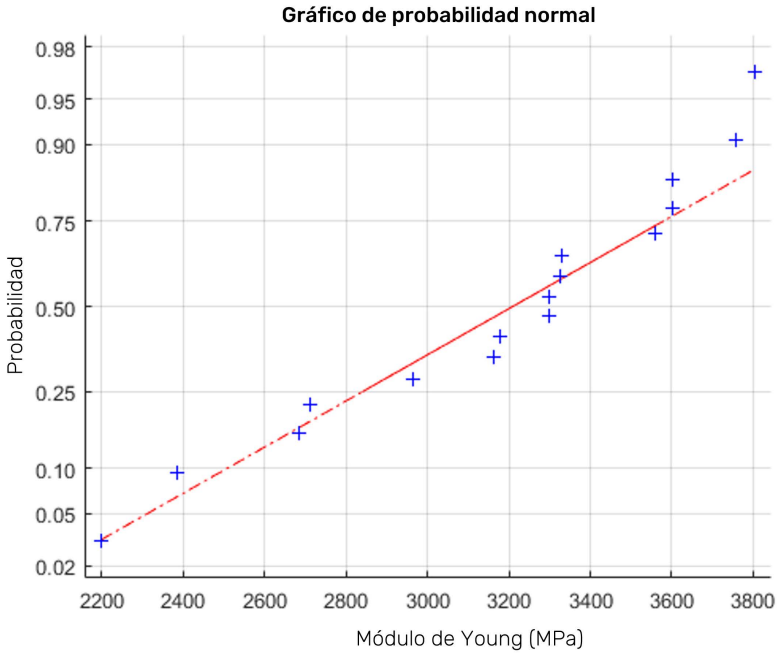


Figura 50. Análisis de distribución de datos simulaciones miniprobetas

Por lo tanto, teniendo en cuenta la normalidad de los datos y que, al ser datos de simulaciones, que no cambian entre mediciones repetidas, también presentan homogeneidad de varianza, se puede aplicar ANOVA. El test ANOVA multifactorial con un nivel de significancia del 95% ha generado los resultados de la Tabla 19.

Tabla 19. Resultados test ANOVA multifactorial para simulaciones

Grupo	Valor p	Diferencia significativa (p<0.05)
Altura de capa (g1)	0.0143	Sí
Porcentaje de relleno (g2)	0.0089	Sí
N° de perímetros (g3)	0.0008	Sí
g1*g2	0.1820	No
g1*g3	0.0506	No
g2*g3	0.0167	Sí

En conclusión, tanto los efectos individuales de los parámetros como la interacción entre porcentaje de relleno y número de perímetros tienen un impacto en el módulo de elasticidad resultante. Lo cual concuerda con las conclusiones obtenidas en el estudio estadístico de los datos experimentales. Los datos de elasticidad de las simulaciones, agrupados según el parámetro estudiado, se muestran en la Figura 51.

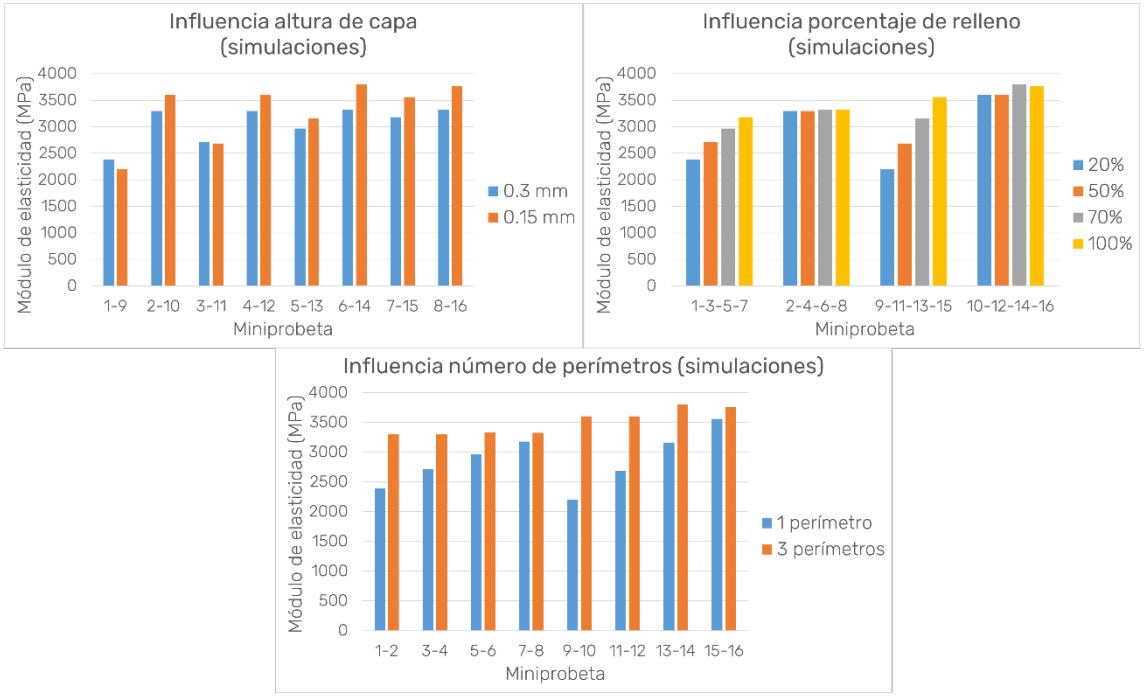


Figura 51. Influencia de parámetros en resultados de simulaciones miniprobeta

7.1.2.7. Optimización

Para comenzar con el proceso de optimización de los parámetros geométricos de las miniprobetas, ha sido necesario definir las variables (con sus valores extremos), restricciones, objetivos y la función aptitud. Estos se especifican en la Figura 52.

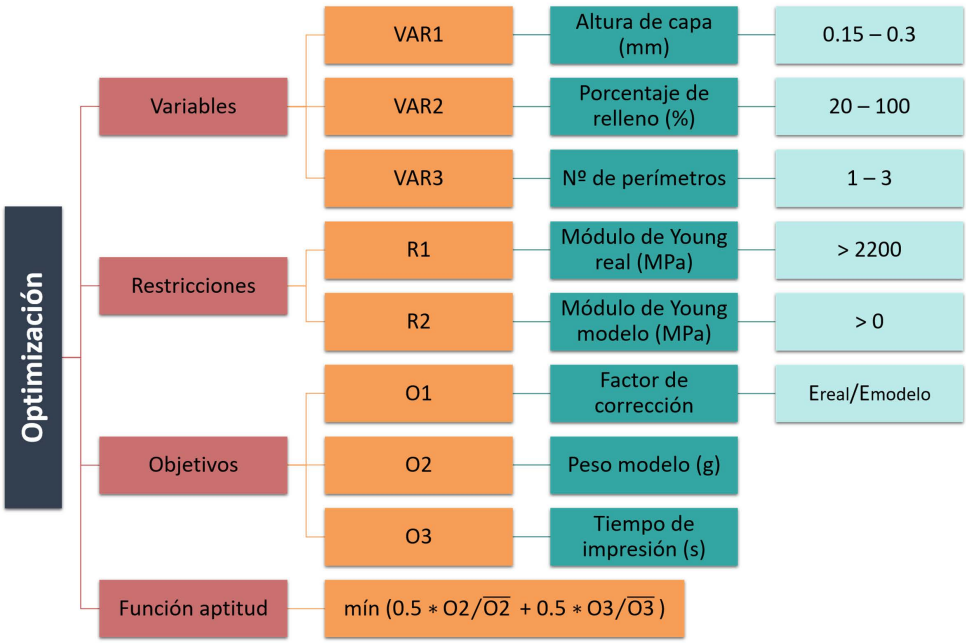


Figura 52. Variables, restricciones, objetivos y función aptitud para optimización de miniprobetas

- *Restricciones*

Una de las restricciones impuestas es que el módulo de Young equivalente de la pieza impresa (R1) sea superior a 2200 MPa. La otra, simplemente es que el módulo de Young obtenido de la simulación (R2) sea superior a 0 MPa. Siempre que pueda simularse, esta segunda restricción no supondrá una penalización, pero su inclusión en la optimización permitirá conocer el factor de corrección y su predicción al proponer la configuración óptima.

- *Objetivos y función aptitud*

Entre los objetivos se encuentran el factor de corrección (O1) que se obtiene dividiendo el módulo de Young de la pieza impresa entre el del modelo simulado. Este no se incluye en la función aptitud, pero se toma como objetivo para obtener la predicción de su valor en cada nueva propuesta de configuración óptima (es decir, se considera "objetivo" puesto que interesa como una variable de respuesta, pero no afecta a la optimización).

En cambio, los otros dos objetivos, peso (O2) y tiempo de impresión (O3) teóricos, sí se tienen en cuenta en la función aptitud, ya que se pretende minimizarlos, ponderándolos al 50% y normalizando sus valores al rango 0-1 (esta normalización permite que tanto el objetivo asociado al peso como al tiempo de impresión estén en un orden de magnitud equivalente, de modo que ambos tendrán la misma importancia o peso para la optimización). Con lo que la función aptitud quedaría como se muestra en la Ecuación (7).

$$\text{Función de aptitud} = \text{mín}\left(0.5 \cdot \frac{O2}{\overline{O2}} + 0.5 \cdot \frac{O3}{\overline{O3}}\right) \qquad (7)$$

Para los objetivos de optimización, se utilizan valores teóricos o del modelo en lugar de los valores de la pieza impresa, puesto que, en un caso normal de aplicación de la metodología, no se fabricarían las piezas. Por otro lado, hay que destacar que estos objetivos se enfocan en la minimización de sus valores, es decir, que no tienen que cumplir con una restricción numérica específica, como sí ocurre con las restricciones. Además, para verificar que los valores reales y teóricos son proporcionales y, por tanto, la predicción del menor peso y tiempo teóricos, corresponderán con la realidad, se han comparado los pesos y tiempo de impresión de los modelos y piezas impresas. En la Tabla 20, se muestra esta comparación.

Tabla 20. Comparación de tiempos de impresión teóricos y pesos del modelo con los reales

Muestra	Tiempo impresión teórico (s)	Tiempo impresión real medio (s)	Peso modelo (g) M	Peso medio pieza impresa (g) R	Error relativo peso (%) |M-R|/R*100
1	95	210	0.248	0.245	1.1
2	154	253.2	0.437	0.454	3.7
3	109	210.8	0.308	0.304	1.5
4	154	245.6	0.437	0.453	3.5
5	140	217	0.368	0.360	2.1
6	164	259.6	0.450	0.465	3.3
7	136	236.2	0.417	0.408	2.2
8	164	270.8	0.449	0.468	4.0
9	153	261.2	0.221	0.221	0.1
10	250	345.6	0.397	0.401	1.1
11	200	292.6	0.292	0.288	1.1
12	250	353	0.397	0.403	1.5
13	211	329.2	0.361	0.357	1.1
14	267	365.2	0.439	0.440	0.2
15	250	344.8	0.427	0.424	0.6
16	268	355.4	0.439	0.442	0.8

En la comparación de los tiempos de impresión, se ha observado diferencia entre los datos teóricos y reales. Esta diferencia es similar en todas las muestras, lo que hace que los tiempos sean casi proporcionales. Esto se debe a que los tiempos de impresión pueden variar según la configuración de la impresora 3D o las condiciones ambientales (que afectan, por ejemplo, al tiempo que tarda en calentar la boquilla de impresión). En cualquier caso, se ha decidido tomar el tiempo de impresión dado por el G-code como válido para buscar la configuración en el que sea mínimo.

Por otro lado, la diferencia entre los pesos del modelo y las piezas impresas se considera despreciable, ya que como máximo alcanza el 4% de error relativo y el error medio es de 1.7%. Por tanto, se tomará el valor del peso del modelo como dato válido para la optimización.

- *Aplicación del algoritmo genético*

Una vez se han definido las variables, restricciones, objetivos y función aptitud que se van a introducir en el algoritmo genético, solamente falta definir sus valores para obtener la estimación de configuración óptima. Los valores introducidos en primer lugar, para iniciar el proceso de optimización, son los presentados en la Tabla 21, correspondientes con el diseño de experimentos aplicado. Las configuraciones que no cumplen con las restricciones se presentan sombreadas.

Tabla 21. Resultados iniciales variables, restricciones y objetivos miniprobetas

Nº	VAR1 (mm)	VAR2 (%)	VAR3	R1 (MPa)	R2 (MPa)	O1 (R1/R2)	O2 (g)	O3 (s)
1	0.3	20	1	1447.44	2386.53	0.607	0.248	95
2	0.3	20	3	2401.44	3297.68	0.728	0.437	154
3	0.3	50	1	1758.88	2712.50	0.648	0.308	109
4	0.3	50	3	2537.84	3297.59	0.770	0.437	154
5	0.3	70	1	2231.07	2965.79	0.752	0.368	140
6	0.3	70	3	2633.34	3328.12	0.791	0.450	164
7	0.3	100	1	2423.22	3176.99	0.763	0.417	136
8	0.3	100	3	2576.28	3326.51	0.774	0.449	164
9	0.15	20	1	1384.23	2197.47	0.630	0.221	153
10	0.15	20	3	2497.24	3602.55	0.693	0.397	250
11	0.15	50	1	1936.52	2685.10	0.721	0.292	200
12	0.15	50	3	2562.86	3602.97	0.711	0.397	250
13	0.15	70	1	2238.52	3162.46	0.708	0.361	211
14	0.15	70	3	2703.72	3804.89	0.711	0.439	267
15	0.15	100	1	2574.67	3558.09	0.724	0.427	250
16	0.15	100	3	2702.41	3759.21	0.719	0.439	268

Tras aplicar la optimización mediante algoritmos genéticos, se determina que el diseño óptimo estimado es con una altura de capa de 0.3 mm, un porcentaje de relleno de aproximadamente el 68.73% y un perímetro. Además, estima que el valor del módulo de elasticidad de la pieza impresa será 2200 MPa, el del modelo, 2948.18 MPa, el factor de corrección de 0.746, pesará 0.364 g y el tiempo de impresión será aproximadamente 138 s.

Como se trata de una configuración que no se ha diseñado anteriormente, se ha obtenido tanto el modelo como la pieza real para su simulación y ensayo experimental, y así comparar los resultados estimados con los reales. La comparación de los resultados de esta nueva probeta, denominada miniprobeta 17, se muestra en la Tabla 22.

Tabla 22. Resultados estimados y reales de la miniprobeta 17

	Módulo de Young real (MPa)	Módulo de Young modelo (MPa)	Factor de corrección (R1/R2)	Peso (g)	Tiempo de impresión (s)
Estimado	2200	2948.18	0.746	0.364	138
Real	2166.62	2711.35	0.799	0.366	122

Para que este nuevo diseño sea considerado óptimo debe cumplir dos condiciones: que el MAPE sea inferior al 5% y que cumpla con las restricciones, concretamente, que el módulo de elasticidad de la pieza impresa sea superior a 2200 MPa. En la Ecuación (8), se realizan las comprobaciones por las que se determina que

no es la configuración óptima.

$$\text{MAPE} = \frac{100\%}{3} \cdot \left(\left| \frac{2166.62-2200}{2166.62} \right| + \left| \frac{0.366-0.364}{0.366} \right| + \left| \frac{122-138}{122} \right| \right) = 5.06\%$$

$$\text{MAPE} > 5\% \rightarrow \text{No cumple} \qquad (8)$$

$$2166.62\ \text{MPa} < 2200\ \text{MPa} \rightarrow \text{No cumple}$$

Sin embargo, este nuevo diseño se introducirá como un dato más que se añade a la base de datos del diseño de experimentos, alimentando así el algoritmo genético con más puntos de muestreo y, por tanto, aumentando la precisión del metamodelo y las posibilidades de encontrar soluciones óptimas o cercanas a la óptima.

Tras aplicar la optimización mediante algoritmos genéticos, añadiendo la miniprobeta 17, ha dado como diseño óptimo una configuración no estudiada anteriormente: altura de capa de 0.3 mm, porcentaje de relleno de 69.39% y un perímetro. Como en el caso anterior, se obtendrán los valores del modelo y la pieza impresa para compararlos con los estimados por la optimización. Estos resultados se presentan en la Tabla 23.

Tabla 23. Resultados estimados y reales de la miniprobeta 18

	Módulo de Young real (MPa)	Módulo de Young modelo (MPa)	Factor de corrección (R1/R2)	Peso (g)	Tiempo de impresión (s)
Estimado	2200.08	2843.61	0.775	0.367	131
Real	2298.71	2707.65	0.849	0.367	122

Tras llevar a cabo la comprobación presentada en el Ecuación (9), se determina que la configuración de la miniprobeta 18 puede ser la óptima. A pesar de ello, se estableció como criterio de validación, repetir el proceso, al menos, cuatro veces.

$$\text{MAPE} = \frac{100\%}{3} \cdot \left(\left| \frac{2298.71 - 2200.08}{2298.71} \right| + \left| \frac{0.367-0.367}{0.367} \right| + \left| \frac{122-131}{122} \right| \right) = 3.99\%$$

$$\text{MAPE} < 5\% \rightarrow \text{Cumple} \qquad (9)$$

$$2298.71\ \text{MPa} > 2200\ \text{MPa} \rightarrow \text{Cumple}$$

Tras alimentar al algoritmo con los resultados de esta nueva configuración, la ejecución de la optimización ha resultado en la propuesta de un nuevo diseño: 0.3 mm de altura de capa, 68.9% de porcentaje de relleno y un perímetro. Sus resultados estimados y reales se muestran en la Tabla 24.

Tabla 24. Resultados estimados y reales de la miniprobeta 19

	Módulo de Young real (MPa)	Módulo de Young modelo (MPa)	Factor de corrección (R1/R2)	Peso (g)	Tiempo de impresión (s)
Estimado	2200.73	2710.4	0.812	0.366	122
Real	2055.07	2707.7	0.759	0.367	122

Como se muestra en la Ecuación (10), esta configuración no puede ser considerada óptima tras conocer los resultados reales.

$$\text{MAPE} = \frac{100\%}{3} \cdot \left(\left| \frac{2055.07 - 2200.73}{2055.07} \right| + \left| \frac{0.367-0.366}{0.367} \right| + \left| \frac{122-122}{122} \right| \right) = 2.41\%$$

$$\text{MAPE} < 5\% \rightarrow \text{Cumple} \tag{10}$$

$$2055.07 \text{ MPa} < 2200 \text{ MPa} \rightarrow \text{No cumple}$$

En la cuarta iteración, introduciendo los resultados de la miniprobeta 19, se ha estimado que el diseño óptimo es: 0.3 mm de altura de capa, 69.19% de porcentaje de relleno y un perímetro. Los resultados de la miniprobeta 20 se presentan en la Tabla 25.

Tabla 25. Resultados estimados y reales de la miniprobeta 20

	Módulo de Young real (MPa)	Módulo de Young modelo (MPa)	Factor de corrección (R1/R2)	Peso (g)	Tiempo de impresión (s)
Estimado	2200.09	2707.67	0.813	0.367	122
Real	2140.58	2708.36	0.79	0.367	122

Al realizar la comprobación de la Ecuación (11), se obtiene que la miniprobeta obtenida con este nuevo diseño no es óptimo.

$$\text{MAPE} = \frac{100\%}{3} \cdot \left(\left| \frac{2140.58 - 2200.09}{2140.58} \right| + \left| \frac{0.367-0.367}{0.367} \right| + \left| \frac{122-122}{122} \right| \right) = 0.93\%$$

$$\text{MAPE} < 5\% \rightarrow \text{Cumple} \tag{11}$$

$$2140.58 \text{ MPa} < 2200 \text{ MPa} \rightarrow \text{No cumple}$$

Tras realizar las 4 iteraciones, se deduce que la miniprobeta 18 es la óptima, ya que es la que cumple con la restricción de tener un módulo de elasticidad superior a 2200 MPa, con el mínimo peso y menor tiempo de impresión posibles. En la Tabla 26 se muestra la comparación de los resultados obtenido en la optimización.

Tabla 26. Resumen de propuestas de miniprobeta óptima

N°	VAR1	VAR2	VAR3	R1	R2	01	02	03	Cumplimiento	
	(mm)	(%)		(MPa)	(MPa)	(R1/R2)	(g)	(s)	MAPE	R1
17	0.3	68.73	1	2166.62	2711.35	0.799	0.366	122	No	No
18	0.3	69.39	1	2298.71	2707.65	0.849	0.367	122	Sí	Sí
19	0.3	68.9	1	2055.07	2707.7	0.759	0.367	122	Sí	No
20	0.3	69.19	1	2140.58	2708.36	0.79	0.367	122	Sí	No

En la Tabla 27, se puede comprobar que existe un error muy bajo (menor del 1%) en la predicción del peso de la pieza a partir del modelo. Además, tanto los tiempos de impresión reales como por pesos medios de las piezas impresas son de los más bajos, comparados con el resto de diseños.

Tabla 27. Comparación de tiempos y pesos de nuevos diseños

Muestra	Tiempo impresión teórico (s)	Tiempo impresión real medio (s)	Peso modelo (g)	Peso medio pieza impresa (g)	Error relativo peso (%)
			M	R	|M-R|/R*100
1	95	210	0.248	0.245	1.1
2	154	253.2	0.437	0.454	3.7
3	109	210.8	0.308	0.304	1.5
4	154	245.6	0.437	0.453	3.5
5	140	217	0.368	0.360	2.1
6	164	259.6	0.450	0.465	3.3
7	136	236.2	0.417	0.408	2.2
8	164	270.8	0.449	0.468	4.0
9	153	261.2	0.221	0.221	0.1
10	250	345.6	0.397	0.401	1.1
11	200	292.6	0.292	0.288	1.1
12	250	353	0.397	0.403	1.5
13	211	329.2	0.361	0.357	1.1
14	267	365.2	0.439	0.440	0.2
15	250	344.8	0.427	0.424	0.6
16	268	355.4	0.439	0.442	0.8
17	122	217	0.366	0.367	0.28
18	122	215.8	0.367	0.369	0.43
19	122	210	0.367	0.366	0.64
20	122	211.4	0.367	0.364	0.71

En esta aplicación, se ha añadido el parámetro de factor de corrección con el objetivo de poder obtener un cociente que permita la predicción del módulo de elasticidad real, a partir de los resultados obtenidos en el modelo. En este caso, tras la comparación de las 20 probetas, la media aritmética de este factor de corrección ha sido 0.732, con una desviación típica de 0.058. La dispersión de los datos se muestra en la Figura 53.

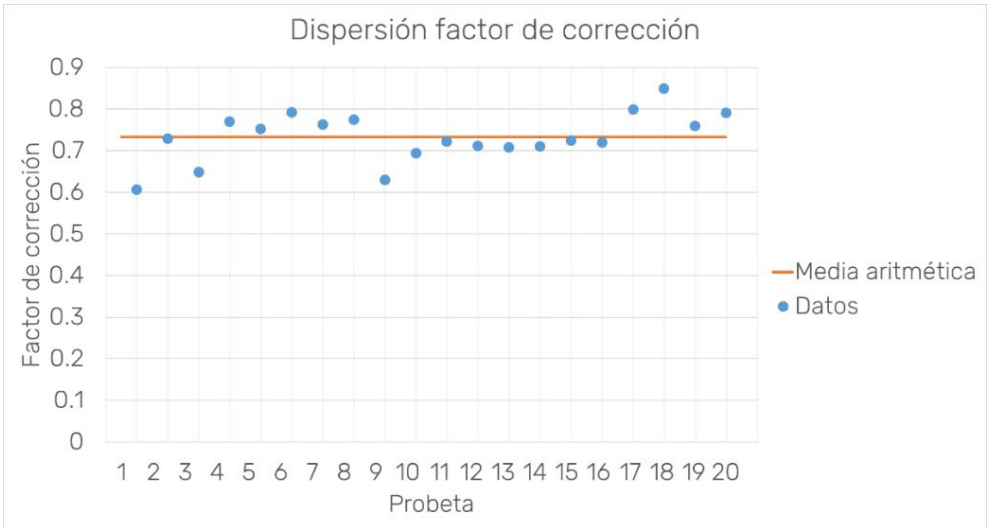


Figura 53. Dispersión de datos de factor de corrección

Teniendo en consideración que solamente el 65% de los datos se encuentra dentro del intervalo 0.732 ± 0.058, se ha decidido realizar un filtrado de datos, en el cual se eliminan los datos más dispersos y se calcula la media de los restantes. Este filtrado resulta en la distribución mostrada en la Figura 54, que presenta una media aritmética de 0.733.

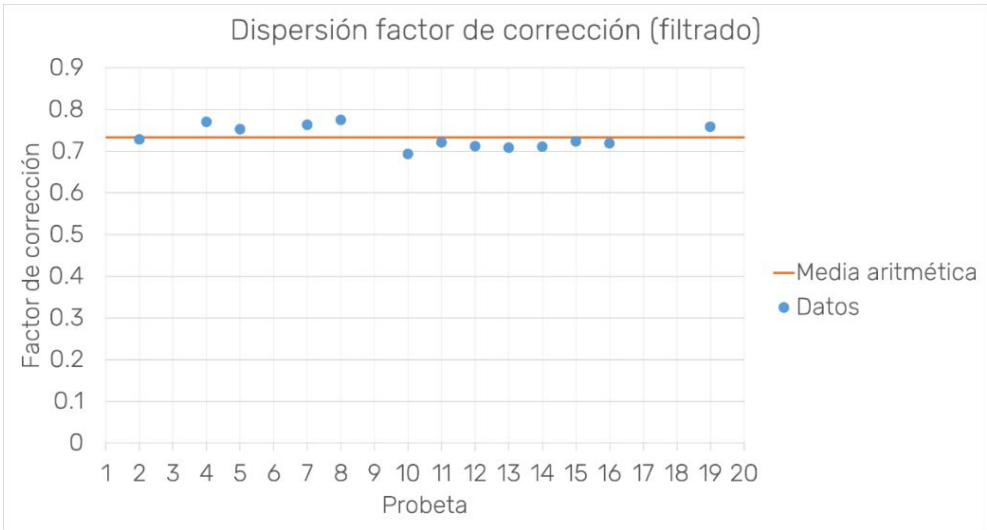


Figura 54. Dispersión de valores menos dispersos del factor de corrección

7.1.3. Conclusiones

La aplicación práctica de las miniprobetas de PLA ha reafirmado la dificultad de predecir el comportamiento mecánico real de la pieza impresa, aunque, en todos los casos, el módulo de elasticidad de esta es inferior al obtenido en las simulaciones. Aunque, eliminando los valores más dispersos, se puede afirmar que el módulo de Young equivalente de las piezas impresas será un 73.3% del módulo obtenido en las

simulaciones. Esta diferencia entre resultados mecánicos puede deberse a la definición de las propiedades del material en el proceso de AEF, ya que, para simplificar el proceso, se han introducido las propiedades del material según catálogo. Sin embargo, con un buen ajuste de las propiedades del material en el AEF sí sería posible aproximarse a los valores experimentales. Algunos autores, han obtenido el módulo de elasticidad del PLA tras ensayar el material a flexión. Por ejemplo, R. Paz et al. [80] han determinado que este valor es 2950.83 MPa, que representa un 76.43% del módulo de Young obtenido en el catálogo, 3861 MPa. Este valor se acerca al factor de corrección mencionado anteriormente.

El resto de resultados, el peso del modelo y el tiempo de impresión teórico, sí es posible predecirlos mediante la interpolación del modelo Kriging, para las diferentes configuraciones.

El diseño óptimo para una miniprobeta sometida a flexión, según la optimización mediante algoritmos genéticos, es la correspondiente a la miniprobeta 18: altura de capa 0.3 mm, 69.39% de porcentaje de relleno y un perímetro (incluyendo una capa sólida en la base y la cara superior). Esta configuración cumple con los requerimientos de módulo elástico, con el menor peso y tiempo de impresión.

Si no se tiene en cuenta la precisión del módulo de elasticidad, en aplicaciones con un rango amplio de valores aceptables y teniendo en cuenta el factor de corrección simulación-realidad, la optimización de parámetros de impresión mediante modelos teóricos puede ser beneficioso en el ahorro de material y tiempo. Esta afirmación es válida siempre que se pretenda obtener una configuración óptima y, para ello, se requiere imprimir y ensayar las piezas.

En el caso concreto de la miniprobeta a flexión, se ha demostrado que no requiere un mayor número de capas (no hay que recurrir a una altura de capa de 0.15 mm), tampoco un 100% de relleno ni más de una capa sólida exterior para conseguir las prestaciones mecánicas requeridas. Concretamente, si se compara la pieza óptima (miniprobeta 18) con la que presenta una altura de capa de 0.15 mm, 100% de relleno y 3 perímetros (miniprobeta 16), en valores reales, se ha reducido un 16.6% el peso y un 39.3% el tiempo de impresión, lo que también implica una reducción del módulo elástico de un 14.94%, pero manteniéndose en el rango de comportamiento mecánico válido.

7.2. SCAFFOLDS PARA ENSAYO BIOLÓGICO

La segunda aplicación práctica, para validar las metodologías de modelado y optimización desarrolladas en la presente tesis, es la de scaffolds cúbicos de PLA impresos en 3D. El objetivo es obtener la mejor configuración o diseño de estos scaffolds en referencia al patrón de relleno, la superficie disponible para la adherencia de las células, el tamaño de los poros y la interconectividad de estos, así como la rigidez del mismo, para maximizar el crecimiento celular y la regeneración de tejido óseo.

En la Figura 55, se muestra el proceso seguido para optimizar los scaffolds para regeneración de tejido, para los cuales se busca que tenga un patrón de relleno con curvatura para generar superficies cóncavas que favorezcan la proliferación celular.

En concreto, en referencia a los patrones de relleno, se utilizaron tres tipos: el giroide, el sinusoidal paralelo y el sinusoidal simétrico. El primero puede obtenerse en laminadores comerciales, mientras que, para obtener el resto ha sido necesario utilizar el programa FullControl Gcode Designer [20] y crearlos a través de modelos matemáticos. Con lo cual, para las geometrías diseñadas con Slic3r 1.3.0, sólo es necesario definir los parámetros de impresión (altura de capa y porosidad); mientras que, para las estructuras generadas con FullControl, se requieren tanto parámetros de diseño (amplitud, número de ciclos, ancho de extrusión, altura de capa, etc.) como ecuaciones matemáticas, para definir las trayectorias del relleno.

Atendiendo a las conclusiones anteriormente expuestas, se ha seleccionado VOLCO [31] como software para el modelado 3D, ya que se trata de geometrías pequeñas y complejas [72].

Además, se aplica la metodología de optimización a partir de los resultados del AEF de los modelos, con el fin de conseguir una pieza con un módulo de elasticidad y un tamaño de poro adecuados para aplicaciones óseas y, al mismo tiempo, maximizar la superficie de adherencia y la interconectividad entre los poros. Esta optimización se ha realizado en dos etapas. En la primera, se optimiza el scaffold sinusoidal paralelo para obtener la ecuación matemática (seno, coseno o seno + coseno) que genera el mejor patrón. Y, en la segunda etapa, se aplica esta ecuación como predeterminada a los scaffolds sinusoidales simétricos y se aplica la metodología de optimización a sus modelos y los de patrón giroide. Finalmente, se comparan los resultados de las tres configuraciones para elegir la óptima.

Por último, con el objetivo de validar las conclusiones de la optimización, ciertos diseños se imprimieron en 3D y se sometieron a un ensayo biológico.

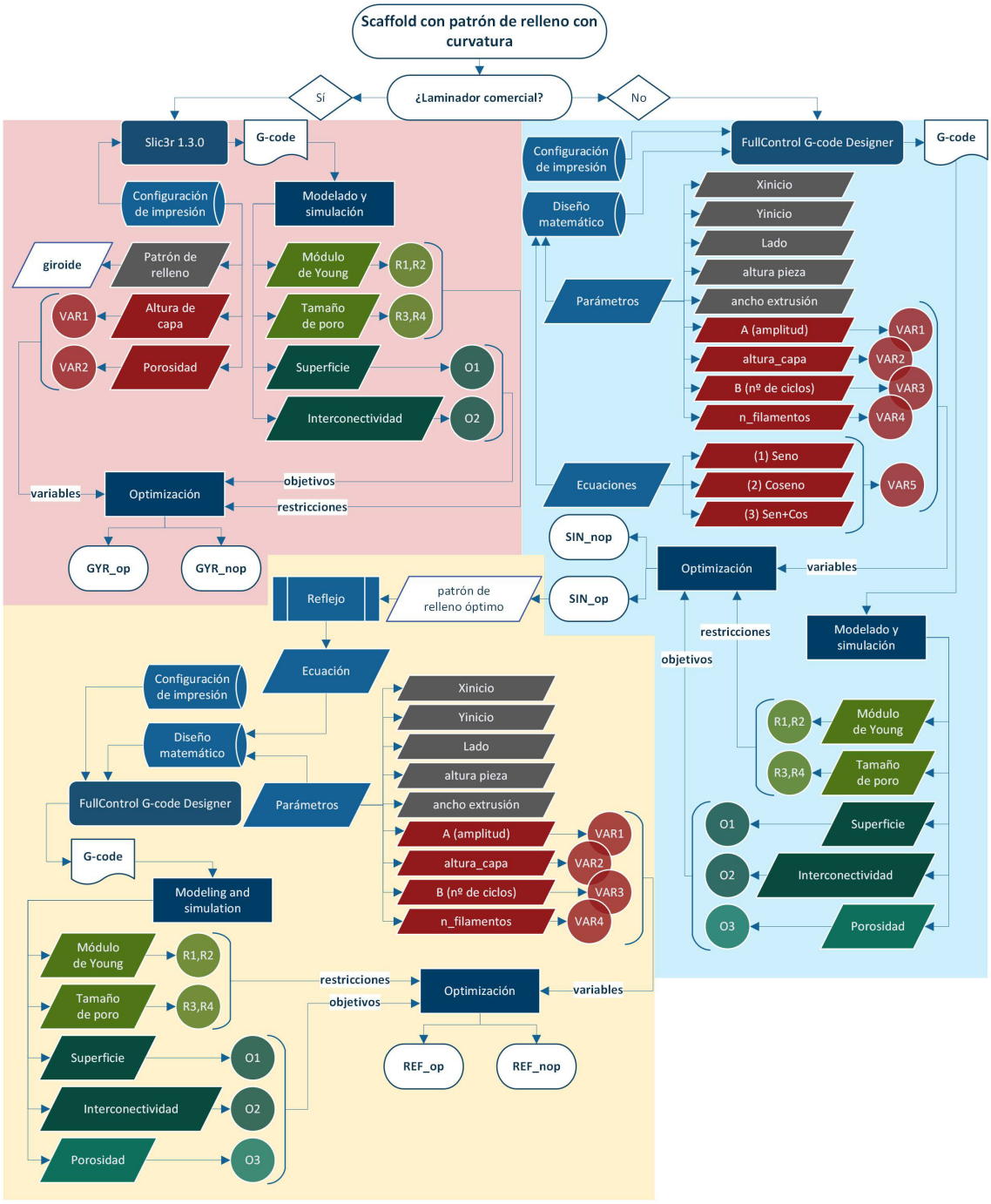


Figura 55. Diagrama optimización de scaffolds con patrón de relleno con curvatura: scaffold sinusoidal paralelo (azul), scaffold sinusoidal simétrico (amarillo) y giroide (rojo)

7.2.1. Selección de parámetros a optimizar

El objetivo de esta aplicación práctica es la obtención de los parámetros de impresión óptimos que resulten en una estructura porosa cuya superficie de adherencia e interconectividad de los poros, permita una proliferación de células y regeneración de tejido óseo eficiente. Además, los scaffolds deberán cumplir con ciertos requisitos mecánicos, de acuerdo al tejido nativo en el que será implantado, en este caso, del óseo trabecular.

En primer lugar, se parte de la premisa de que el crecimiento celular es favorecido por la curvatura de la estructura interna de los poros, sobre todo en las zonas cóncavas [66–70]. Con lo cual, se busca diseñar scaffolds cuyo patrón de relleno incluya curvas. Sin embargo, en los laminadores comerciales, el único patrón que cumple con esta premisa es el giroide. Por tanto, se ha recurrido al software FullControl Gcode Designer [20], que permite generar un código G sin restricciones, a partir de diseños matemáticos.

7.2.1.1. Fase preliminar

Como punto de partida y con la intención de conocer el patrón de relleno óptimo, se generaron tres tipos de patrones sinusoidales a partir de las fórmulas matemáticas del seno y el coseno, denominados coseno, seno y seno-coseno. El patrón coseno consiste en la repetición de forma paralela de filamentos diseñados a partir de la ecuación del coseno, el patrón seno, sería igual, pero siguiendo la ecuación del seno y, el patrón seno-coseno, presenta la ecuación seno en capas impares y coseno en las pares. Además, en el caso de los dos primeros patrones, las capas se someten a un giro de 90° con respecto a la anterior, y en el tercer patrón este giro se da cada dos capas. En la Figura 56 se pueden observar las diferencias geométricas de cada patrón de relleno.

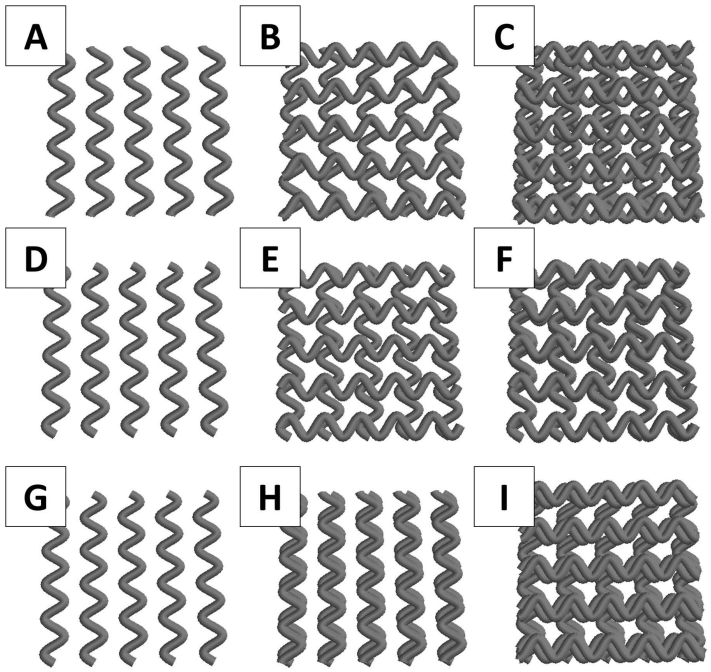


Figura 56. Diferencias entre los patrones de relleno sinusoidales. (A) Primera capa del patrón coseno. (B) Dos capas del patrón coseno. (C) Cuatro capas del patrón coseno. (D) Primera capa del patrón seno. (E) Dos capas del patrón seno. (F) Cuatro capas de patrón seno. (G) Primera capa del patrón seno-coseno. (H) Dos capas del patrón seno-coseno. (I) Cuatro capas del patrón seno coseno.

Al introducir el diseño en FullControl, se requiere la definición de los ajustes de impresión (temperatura de la boquilla, temperatura de la cama, velocidad de avance de la boquilla o diámetro del filamento) y el diseño matemático para generar las trayectorias. A su vez, el diseño matemático necesita que se definan las ecuaciones y se les dé valor a sus parámetros. Los parámetros que representan la geometría general de la pieza son *Lado*, cuyo valor es la longitud y el ancho del cuadrado (en mm), y *altura_pieza*, que es la altura del prisma (mm).

En este caso, como se seleccionó un scaffold cúbico para el estudio, todos estos valores fueron iguales (*Lado* y *altura_pieza*). Para definir la ecuación sinusoidal, A, B y n_filamentos representan la amplitud de la onda, su número de ciclos y el número de líneas de filamento en cada capa, respectivamente. Además, se requieren algunos parámetros de impresión, como Xinicio e Yinicio, que representan las coordenadas XY de origen de la pieza con respecto al origen de coordenadas de la cama de impresión (mm); altura_capa, que representa la altura de cada capa de filamentos (mm); y ancho_filamento, que es el ancho de extrusión del filamento (mm). Algunos de ellos se representan gráficamente en la Figura 57.

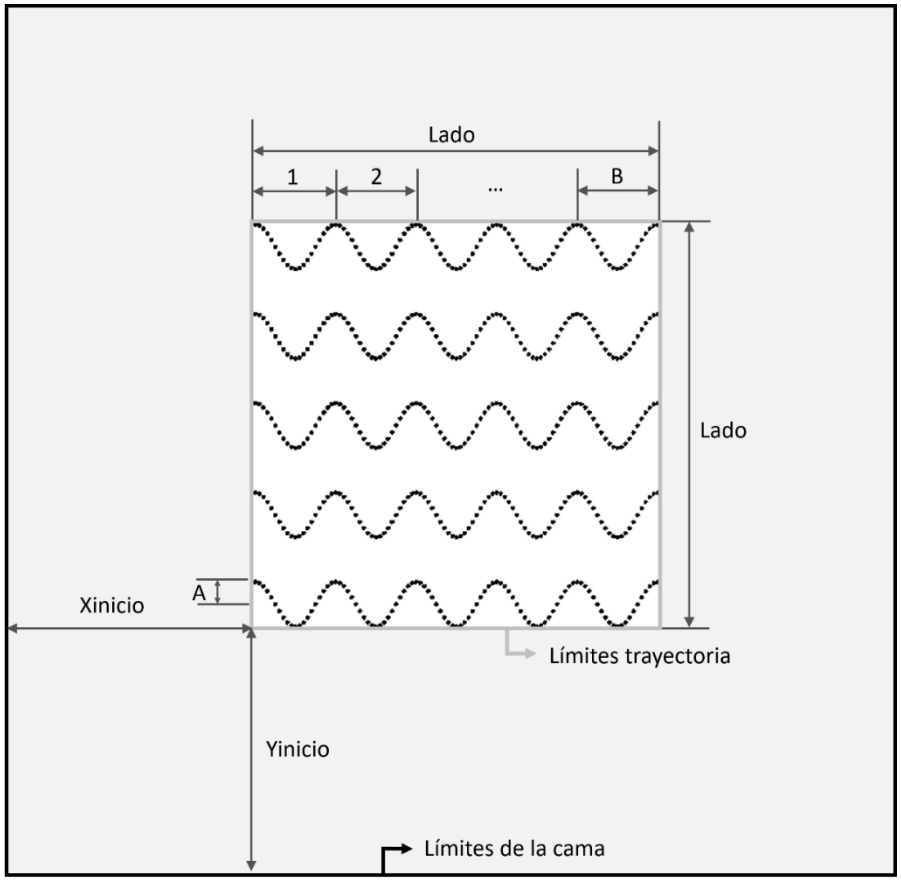


Figura 57. Parámetros de diseño para la generación de trayectorias sinusoidales

Teniendo en cuenta lo anterior, a la hora de optimizar hace falta incluir una variable para elegir el patrón óptimo (coseno, seno o seno-coseno) y cuatro variables que influirán en el diseño matemático: la amplitud (A), el número de ciclos (B), el número de filamentos por capa (n_filamentos) y la altura de capa (altura_capa).

Por otro lado, el principal requisito mecánico que debe cumplir un scaffold es tener un módulo de Young similar al del tejido en el que se va a implantar. En este caso, los scaffolds han sido diseñados específicamente para huesos trabeculares. Para este tipo de tejido, algunos autores definen su rigidez en el rango de 10-1500 MPa [35,84]. Por lo que el módulo de elasticidad de las piezas deberá cumplir con esta restricción.

Además, según la bibliografía sobre crecimiento celular y regeneración de tejido óseo a través de scaffolds, la superficie, forma, tamaño e interconectividad de los poros son relevantes para el diseño de los mismos [66–68]. En cuanto al tamaño de los poros, se establece que se buscan poros de más de 300 µm y del orden de la centena [66,85–88]. Además, se afirma que el valor óptimo para la formación de hueso es de 500 µm [88].

Atendiendo a lo descrito anteriormente, el tamaño de poro también se considerará una restricción en el proceso de optimización, mientras que, la superficie disponible y la interconectividad entre los poros se considerarán objetivos a maximizar.

7.2.1.2. Evaluación global

Como se ha indicado anteriormente, se realiza una primera aplicación de las metodologías de modelado y optimización a scaffolds con tres patrones de relleno sinusoidales para obtener el óptimo. En todos ellos, los filamentos son paralelos. Aunque, el objetivo final es comparar dos patrones más, el sinusoidal reflejado (con filamentos simétricos) y el giroide.

Después de obtener los resultados de los scaffolds sinusoidales, se introduce el patrón óptimo (coseno, seno o seno-coseno) como patrón por defecto de un nuevo scaffold sinusoidal reflejado con el fin de probar qué forma es mejor para el crecimiento celular. Se propone utilizar este nuevo patrón ya que, al enfrentar los filamentos sinusoidales, se generan zonas estrechas que favorecen el encuentro de células y la formación de pseudo-anillos, los cuales se ha comprobado que son idóneos para el crecimiento celular [66]. La diferencia entre filamentos paralelos y simétricos se muestra en la Figura 58.

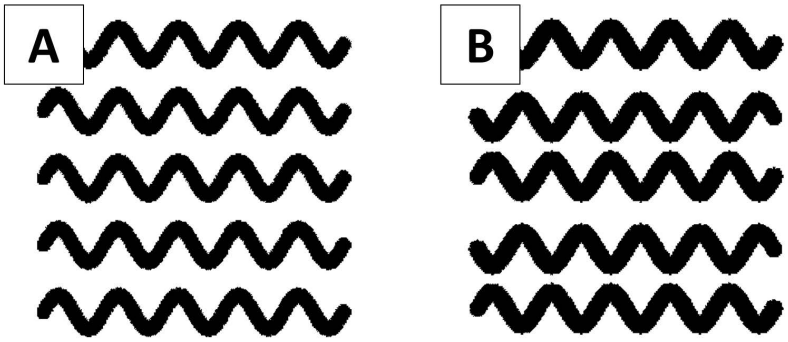


Figura 58. Diferencia entre las configuraciones de los filamentos. (A) Filamentos paralelos. (B) Filamentos simétricos

Teniendo en cuenta que en el patrón sinusoidal reflejado no se han probado diferentes ecuaciones matemáticas, sino que sólo se utiliza la obtenida como óptima en el estudio previo del scaffold sinusoidal paralelo, el número de variables para la optimización queda reducido a cuatro: A, B, n_filamentos y altura_capa.

A diferencia de los grupos anteriores, los giroides se diseñan mediante un laminador comercial (Slic3r 1.3.0); por tanto, las ecuaciones matemáticas y sus parámetros no son modificables. En consecuencia, las dos únicas variables seleccionables son la altura de capa y la porosidad, la cual modificará la disposición de los filamentos.

Tanto para los scaffolds sinusoidales reflejados como para los giroides, las restricciones y objetivos a maximizar serán los mismos que en el scaffold sinusoidal paralelo. Además, con el fin de conocer la porosidad en los scaffolds sinusoidales, paralelos y reflejados, se considerará esta como tercer objetivo en la optimización, aunque no influya en la función aptitud.

7.2.2. Aplicación de la metodología

Con el fin de optimizar las estructuras porosas para la regeneración del hueso trabecular, sin necesidad de imprimirlos o realizar ensayos experimentales, se requiere una metodología de modelado y optimización con la aplicación del AEF. Los pasos de la metodología seguida se presentan en la Figura 59 y en el Anexo H.

En resumen, las metodologías de modelado y optimización se aplican a tres grupos de scaffolds cuya diferencia es su patrón: sinusoidal, sinusoidal reflejado y giroide. Los dos primeros grupos se diseñaron con software no comercial, y la variación entre el patrón sinusoidal y el reflejado son las características geométricas de los filamentos: paralelos o simétricos, respectivamente.

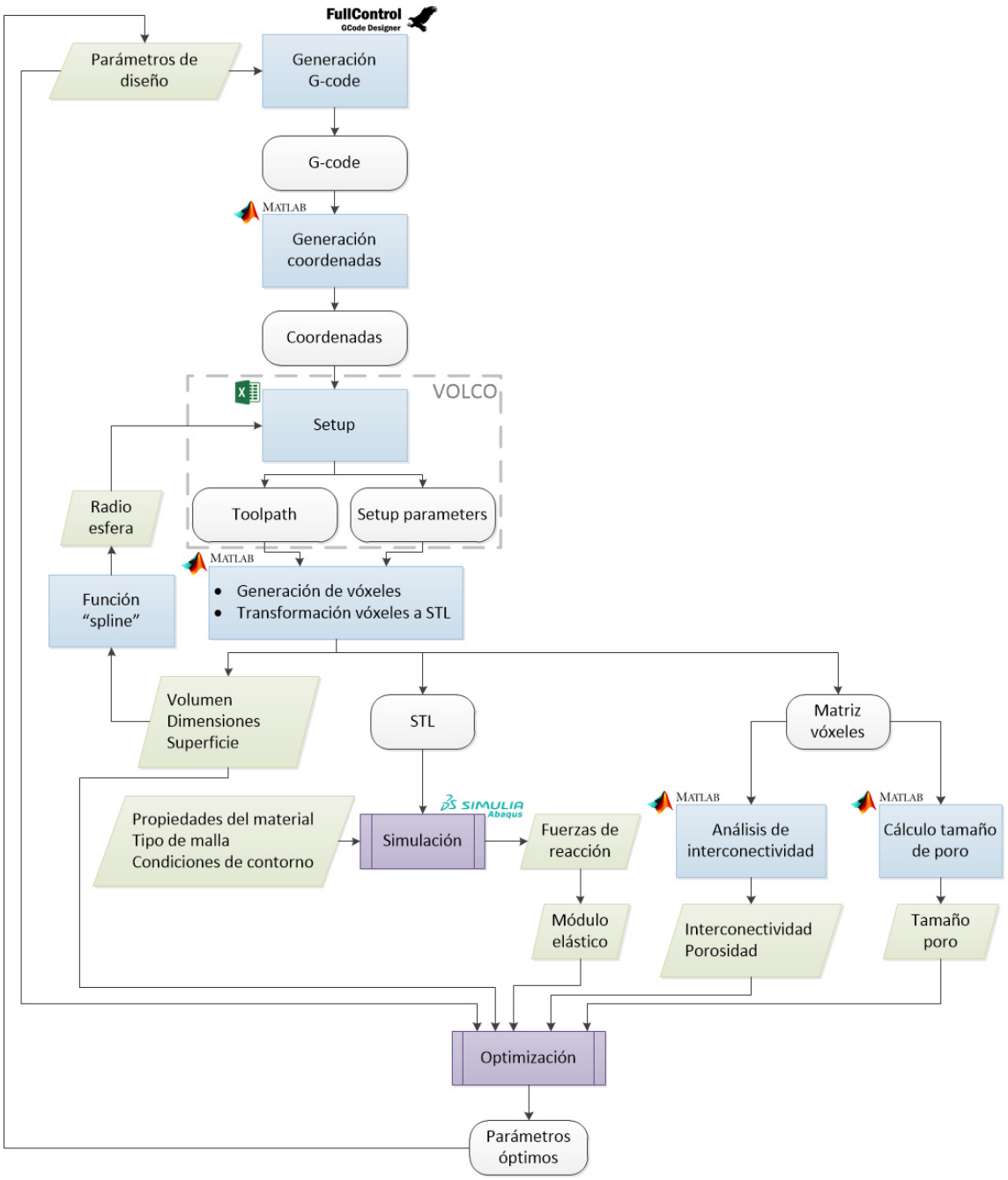


Figura 59. Metodología de modelado y optimización de scaffolds sinusoidales

7.2.2.1. Material

Los scaffolds diseñados serán impresos en PLA, un bioplástico con propiedades adecuadas para aplicaciones de Ingeniería Tisular. En particular, el PLA Smartfil, cuyas propiedades se mostraron anteriormente en la Tabla 12, pero se presentan de manera resumida en la Tabla 28.

Tabla 28. Propiedades de PLA Smartfil (resumidas)

Material Properties of PLA – Smartfil	
Diámetro	1.75 mm
Densidad	1.24 g/cm³
Módulo de elasticidad	3861 MPa
Temperatura de impresión	220 ± 20 °C
Temperatura de la cama	0 - 60 °C

Con el fin de hacer más sencillo el proceso de aplicación de la metodología de modelado y AEF, se utilizarán datos de catálogo para definir las propiedades del material. Sin embargo, cabe mencionar que otros investigadores han optado por realizar pruebas en el material para determinar sus propiedades [6,25].

7.2.2.2. Diseño de scaffolds sinusoidales

El objetivo principal de esta aplicación práctica es obtener scaffolds con curvatura para promover el crecimiento celular. Para ello, se generan patrones de relleno sinusoidales en FullControl Gcode Designer.

Para la creación de las trayectorias de los filamentos se pueden añadir diferentes eventos en la botón “Add Features” y parámetros en “Assign Parameters Names”, como se puede observar en la Figura 60. Las funciones del programa utilizadas para el diseño son “line equation” para definir las ecuaciones matemáticas de las líneas sinusoidales, “cartesian repeat” para repetir barridos idénticos (líneas y capas), “line (cartesian)” para definir los movimientos sin extrusión de material de la boquilla y “repeat rule” para girar las capas incrementalmente 90°, intercalando recorridos horizontales y verticales. Los movimientos sin extrusión de material se denominan “travels” y ha sido necesario programar su trayectoria alejándose de la pieza antes de imprimir una nueva capa, para asegurar que el material que siga saliendo de la boquilla no se adhiera a los filamentos ya depositados, modificando la estructura porosa.

Los parámetros asignados son los descritos anteriormente en el apartado 7.2.1 (*Xinicio*, *Yinicio, Lado, altura_pieza, A, B, n_filamentos, altura_capa* y *ancho_extrusión*). Tras incluir las funciones necesarias, las ecuaciones matemáticas para obtener un scaffold cúbico con un patrón sinusoidal y los valores de los parámetros, el software genera el archivo de código G del modelo.

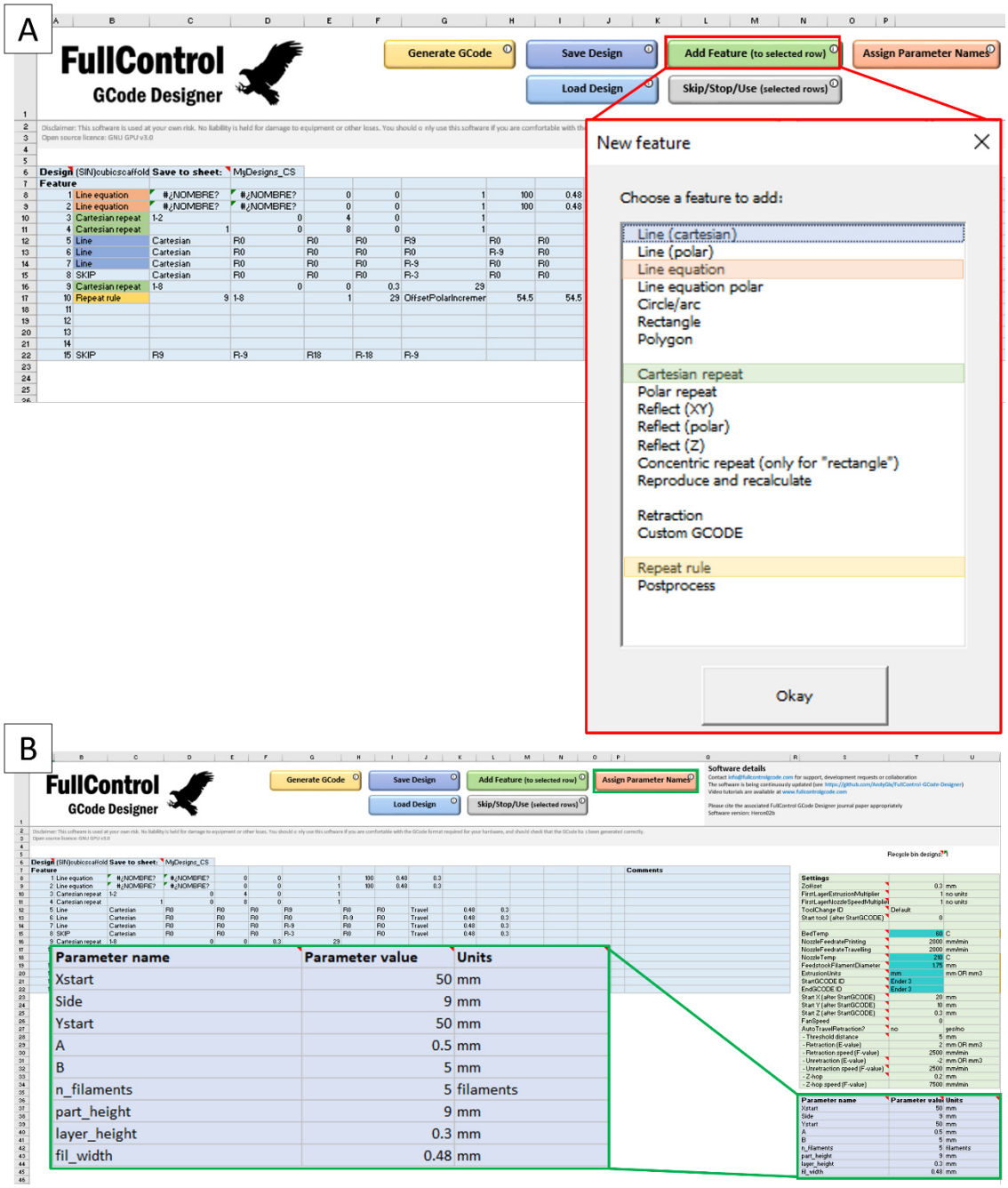


Figura 60. Creación de scaffolds sinusoidales en FullControl GCode Desginer: (A) añadir funciones y (B) asignar parámetros

Para la optimización se deberán generar varios modelos a partir de la misma ecuación, pero cuyos parámetros A, B, n_filamentos y altura_capa varíen. En la Figura 61(A) y la Figura 61(B) se muestran las diferencias en función del valor de la altura_capa. Las Figura 61(C) y Figura 61(F) muestran cómo el valor de la amplitud (A) modifica el relleno. La influencia del número de ciclos (B) se aprecia comparando la Figura 61(C) y la Figura 61(E). Por último, la Figura 61(C) y la Figura 61(D) muestran la variación del trazado modificando el número de filamentos por capa.

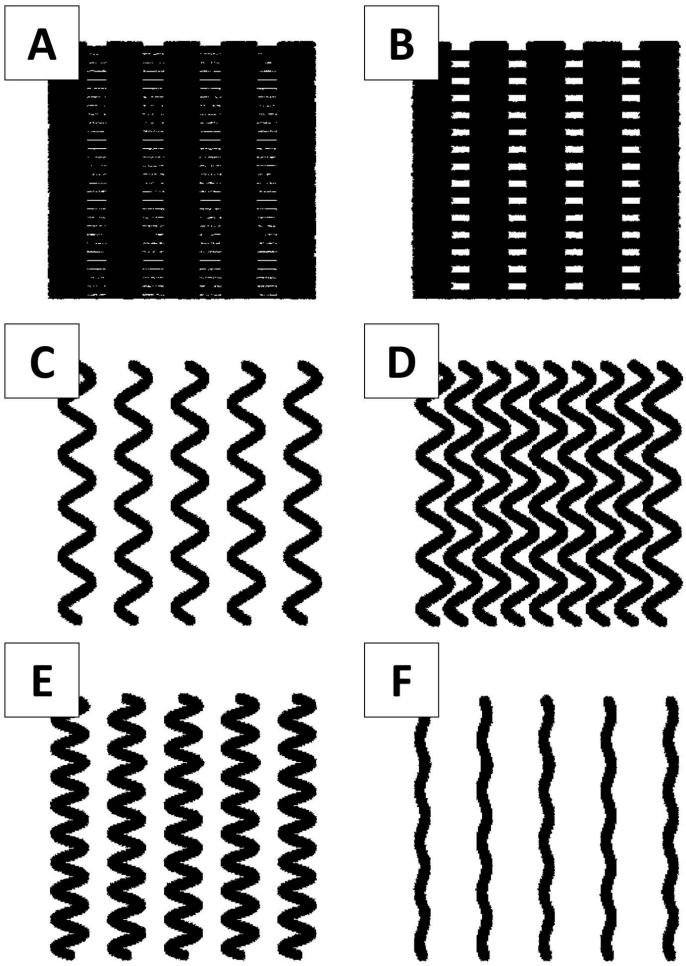


Figura 61. Influencia de los valores de las variables en la geometría. (A) Scaffold con una altura de capa de 0.15 mm. (B) Scaffold con altura de capa de 0.3 mm. (C) Capa con filamentos de amplitud 0.5 mm, 5 ciclos y 5 filamentos por capa. (D) Capa con filamentos de amplitud 0.5 mm, 5 ciclos y 9 filamentos por capa. (E) Capa con filamentos de amplitud 0.5 mm, 9 ciclos y 5 filamentos por capa. (F Capa con filamentos de amplitud 0.1 mm, 5 ciclos y 5 filamentos por capa.

7.2.2.3. Modelado

El siguiente paso para la obtención del modelo STL, es la obtención de las coordenadas de la trayectoria de la herramienta a partir del archivo de código G y, así, poder introducirlo en el software VOLCO [31]. Para ello, se ha utilizado la función *Coord_Gen_Slic3r*, independientemente (es decir, sin ser una función integrada en la subrutina Abaqus_model), en Matlab R2021a. Además de las coordenadas, el software VOLCO necesita otros parámetros como la región del modelo STL, el tamaño del vóxel, la altura de la capa y el radio de la esfera.

En este trabajo se ha realizado un estudio previo del radio de esfera óptimo para cada altura de capa. Para ello, se modeló una línea depositada para cada altura de capa, y se aplicó un paso adicional para mejorar la precisión del volumen voxelizado resultante. Este nuevo paso consiste en un proceso iterativo en el que se aplica

manualmente una interpolación cúbica, mediante la función "spline" de Matlab. Esta interpolación toma como datos el radio de la esfera y el volumen del modelo, dando como resultado el valor del radio de la esfera necesario para que el volumen del filamento voxelizado sea similar al teórico, es decir, el volumen presentado en el código G.

En concreto, este proceso se aplicó a los scaffolds obtenidos mediante FullControl Gcode Designer, los scaffolds sinusoidales y sinusoidales reflejados. A partir de los resultados de la iteración, se seleccionaron los radios de esfera: 134 µm para los scaffods con una altura de capa de 0.15 mm y 188.2 µm para aquellos con una altura de capa de 0.3 mm. Con estos valores de radio de esfera teórico en VOLCO, se consigue un volumen del modelo voxelizado muy similar al volumen teórico de acuerdo al código G.

Por otra parte, en el caso de los giroides, la función "spline" se aplicó iterativamente al volumen del modelo completo para ajustar el radio de la esfera de cada configuración. En concreto, los radios de esfera para los giroides, según sus parámetros de impresión, es decir, variables de diseño usadas en el diseño de experimentos (apartado 7.2.2.9), en este caso la altura de capa y porcentaje de relleno, fueron: 199.57 µm (0.3 mm y 50 %), 199.18 µm (0.3 mm y 60%), 199.24 µm (0.15 mm y 50%) y 199.17 µm (0.15 mm y 60%).

Una vez definidos los valores de los parámetros, VOLCO genera la matriz de vóxeles y el STL, que serán utilizados para la obtención de resultados del tamaño de poro, el módulo de Young, la interconectividad, la porosidad y la superficie de los scaffolds.

7.2.2.4. Interconectividad y porosidad

En la ingeniería tisular, se desean estructuras porosas para el crecimiento celular, pero el valor óptimo de la porosidad no puede ser fijado en las estructuras creadas en FullControl, puesto que se no es una variable del diseño del scaffold como sí que ocurre en el caso del giroide; por tanto, se considera un resultado de la geometría (variable de respuesta que se obtiene a partir del modelo 3D). Por otra parte, la interconectividad de los poros es esencial para la conducción de fluidos, la migración celular y el crecimiento interior de tejido.

El método más utilizado para el análisis de la interconectividad [89] se basa en el concepto de índice de conectividad efectiva de los poros (EPCI) [90]. Este índice indica

la proporción de vóxeles conectados entre la última capa y la primera.

No obstante, para superar las limitaciones de EPCI, algunos autores decidieron aplicar el algoritmo de los 6 vecindarios conectados utilizando la función "bwconncomp" de Matlab [89]. Otros investigadores han utilizado un enfoque similar [91,92]. Unos consideran la interconectividad como una fracción del volumen de huecos accesibles desde el exterior y el volumen total de la estructura [91], mientras que otros adicionalmente lo restringen a conexiones estrechas [92].

A diferencia de estudios anteriores que calculan la interconectividad a partir de una matriz binaria 3D generada a partir de imágenes TC, este trabajo parte de la matriz de vóxeles obtenida en VOLCO. Para ello se ha desarrollado una subrutina de cálculo de interconectividad (Anexo J). El primer paso consiste en invertir la matriz y establecer el valor cero para el material y el uno para los huecos (vóxeles vacíos). A continuación, se define la interconectividad de los poros mediante el valor del índice de volumen conectado (CVI), concretamente, el criterio de dos caras, que se calcula considerando el volumen de los grupos de huecos que conectan dos caras externas cualesquiera, utilizando la Ecuación (12). Nótese que dos vóxeles pertenecen a un mismo grupo de huecos si tienen al menos una cara adyacente.

$$\text{CVI}(\%) = \frac{\text{Volumen de grupos de huecos que unen dos caras externas}}{\text{Volumen total}} \qquad (12)$$

Por último, la porosidad se calcula en la misma subrutina de Matlab 2021a, dividiendo el volumen de los huecos y el volumen total, ya que no es un valor conocido en el proceso de diseño de los scaffolds en FullControl Gcode Designer.

7.2.2.5. Tamaño de los poros

El tamaño de los poros es relevante para el crecimiento celular. No es deseable tener poros ni demasiado pequeños ni demasiado grandes. Teniendo en cuenta esta premisa y la bibliografía, se determinó que el rango aceptable de tamaño de poro estaría entre 300 y 700 µm [66,85–88].

En este trabajo se ha desarrollado una herramienta automática de medición del tamaño de poro (Anexo K) para comprobar si los modelos cumplen con los requisitos dimensionales. Esta herramienta se ha implementado para tratar de establecer un criterio único y para agilizar el proceso de determinación del tamaño de poro, lo cual no es nada sencillo en geometrías tan complejas y voxelizadas. La metodología propuesta consiste básicamente en proyectar los vóxeles en el eje Z, de modo que la

proyección resultante es la que analiza. A modo de símil, si se aplica una fuente de luz desde la parte superior del scaffold y en dirección vertical descendente, la luz que se proyecta en la base sería la proyección que se analiza. De esa proyección, se toman los grupos de luz interiores y se determina el diámetro equivalente medio de todos los grupos, que sería el tamaño de poro.

El proceso se resume en el diagrama mostrado en la Figura 62. Se parte de la misma matriz 3D utilizada para el análisis de interconectividad, que define si cada vóxel representa un hueco o material. A continuación, se utiliza una operación booleana (AND) capa por capa en la coordenada Z para obtener su proyección en una matriz 2D. La Figura 63(A) muestra un ejemplo de la proyección en Z.

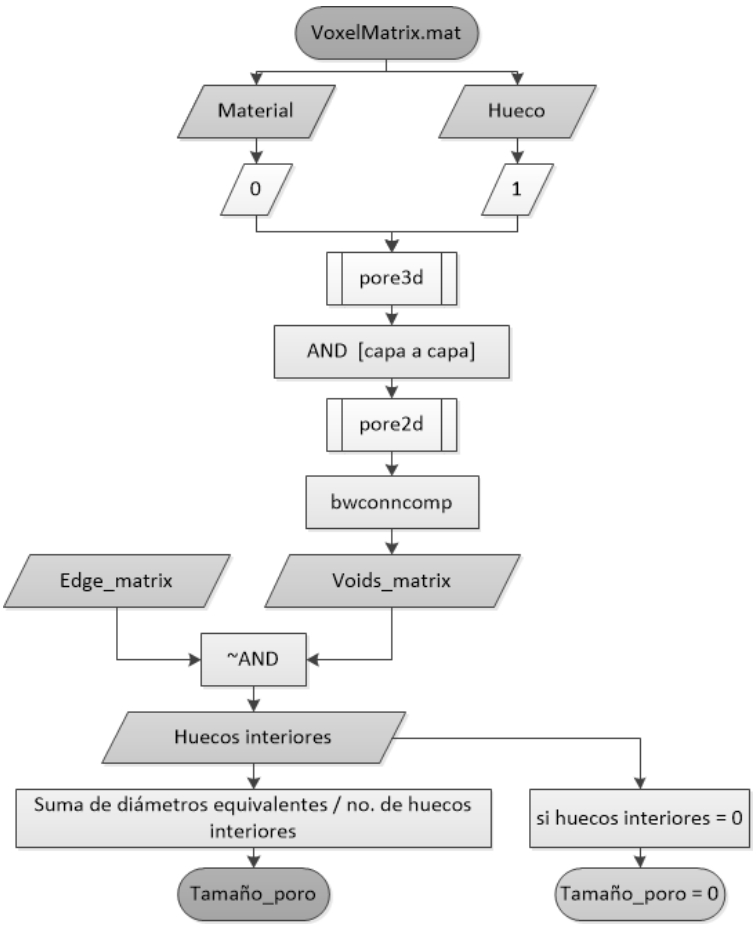


Figura 62. Proceso de cálculo de tamaño de poro

Una vez obtenida la proyección Z, los poros de esta matriz 2D se identifican con la función "bwconncomp" [89]. A continuación, se analizan para determinar si son adyacentes a los bordes (poros exteriores), o no (poros interiores). Solamente se calculará el tamaño de los poros interiores. Se determina el diámetro equivalente de cada uno y, a continuación, se suman todos los resultados y se divide por el número de poros interiores, con lo que se obtiene el diámetro equivalente medio y, por tanto, el tamaño de poro. La Figura 63(B) muestra una proyección de un scaffold poroso y el

diámetro medio equivalente (tamaño de poro).

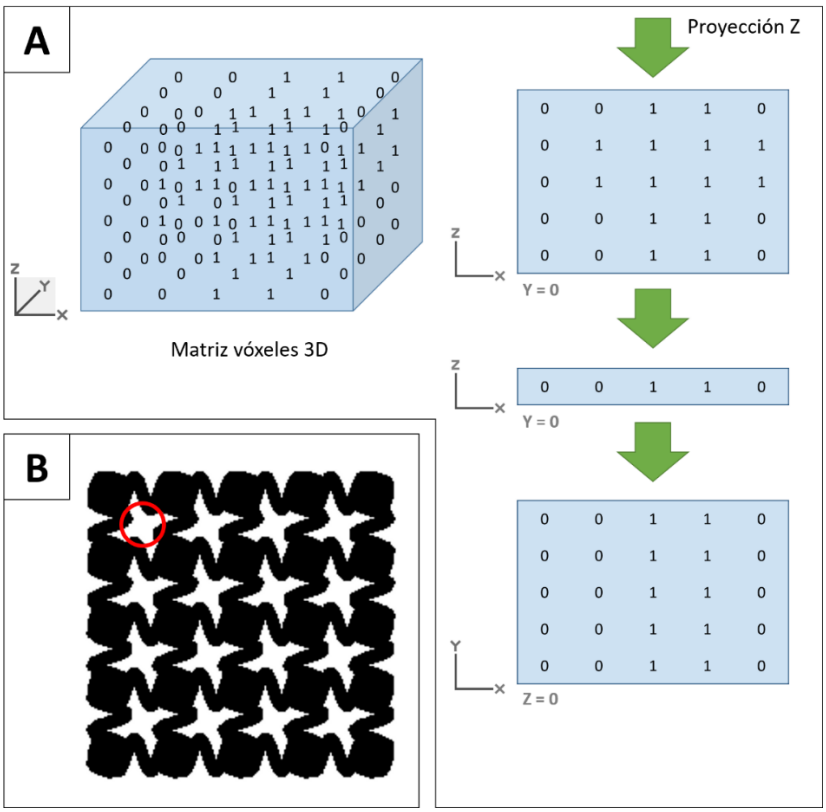


Figura 63. Determinación del tamaño de los poros. (A) Matriz de vóxeles 3D y su proyección Z. (B) Diámetro de poro equivalente.

7.2.2.6. Superficie

Otro objetivo a maximizar en las estructuras porosas es la superficie o área de disponible para la adherencia de las células. Este valor puede extraerse del archivo STL del modelo o directamente en Abaqus/CAE 6.14-1 antes de mallar la pieza (analizando las propiedades de masa con la herramienta "query").

7.2.2.7. Análisis de elementos finitos

Tras obtener el modelo voxelizado a partir del fichero G-code, se importa el STL a Abaqus/CAE 6.14-1 para aplicar el AEF.

El primer paso es convertir los elementos triangulares en tetraedros, obteniendo el modelo mallado. A continuación, se asignan las propiedades del material PLA a toda la pieza. Seguidamente, se crea un nuevo paso (Paso-1) para establecerlo como la etapa final de la simulación. Seguidamente, se definen las condiciones de contorno para simular un ensayo de compresión. Para ello, se limita el desplazamiento Z en la cara inferior de la pieza (base); además, se atribuye la restricción de unión fija o "encastre" a dos puntos de la base para evitar su desplazamiento o rotación, y se aplica

un desplazamiento de 0.2 mm en la dirección Z negativa en la cara superior de la pieza (top), para emular una compresión.

Los resultados significativos de la simulación son el desplazamiento (U) y las fuerzas de reacción (RF), obtenidos en el scaffold deformado. El módulo de Young equivalente se calcula a partir de las fuerzas de reacción obtenidas en la cara inferior (base) en la dirección Z, utilizando la Ecuación (13), extraída de la norma ISO 604 (Plásticos-determinación de las propiedades en compresión).

$$E = \frac{\sigma}{\varepsilon} = \frac{F \bullet L_0}{A \bullet \Delta L} \tag{13}$$

Donde E es el módulo de Young equivalente o módulo de elasticidad (MPa), σ es el esfuerzo de compresión (MPa), ε es la deformación, F es la fuerza de reacción (N), A es el área equivalente (mm^2), ΔL es el desplazamiento (mm) y L_0 es la altura original (mm). Téngase en cuenta que el área equivalente corresponde al área de la base suponiendo que fuera una pieza sólida. Por consiguiente, el módulo de elasticidad equivalente representa la rigidez mecánica general del scaffold, incluyendo el efecto de los poros.

7.2.2.8. Optimización preliminar

Como se ha comentado con anterioridad, antes de la evaluación final del diseño del scaffold con curvatura óptimo, ha sido necesaria la aplicación de la metodología a los scaffolds sinusoidales con filamentos paralelos para obtener el mejor de los tres patrones propuestos: coseno, seno, seno-coseno.

Asimismo, se especificó que para la aplicación de la optimización se partiría de cinco variables. Concretamente, la primera variable (VAR1) es la amplitud de onda (A), de la cual se valorará su influencia atribuyéndole los valores 0.1 y 0.5 mm. La VAR2 es la altura de capa (altura_capa), a la que se le dará los valores 0.15 y 0.3 mm. Para la VAR3, el número de ciclos (B), se barajará incluir 5 y 9 ciclos a lo largo de la curva. En la VAR 4, el número de filamentos por capa (n_filamentos), también se probará a añadir 5 y 9 filamentos por capa. Por último, en la VAR5, el patrón de relleno, se definirá cada uno con un número discreto, 1 (coseno), 2 (seno) y 3 (seno-coseno). La combinación de los valores de las variables seleccionadas (diseño factorial 2^4 x 3) da como resultado cuarenta y ocho muestras (Tabla 29).

Por otro lado, en la metodología se deben definir las restricciones, definiendo si se trata de un máximo o un mínimo. Sin embargo, en este estudio, las restricciones se

definen como un intervalo, por lo que debe establecerse un mínimo y un máximo para cada una. En consecuencia, cada restricción debe dividirse en dos; los resultados del módulo de Young del modelo se utilizan en las restricciones R1 y R2, y el tamaño de los poros en R3 y R4. Concretamente, el módulo elástico equivalente debe ser superior a 10 MPa (R1) e inferior a 1500 MPa (R2), y el tamaño de poro debe ser superior a 300 µm (R3) e inferior a 700 µm (R4). Los resultados de ambos datos para cada configuración, siguiendo la metodología descrita, se presentan en la Tabla 29. Nótese, que todas aquellas muestras que no cumplen con las restricciones, concretamente porque no tienen el tamaño de poro requerido, están sombreadas en la tabla.

Además, los objetivos a maximizar son la superficie (O1) y la interconectividad (O2). Sin embargo, la porosidad (O3) se considerará también un objetivo, ya que se desea conocer su valor para los scaffolds sinusoidales, aunque no se incluirá en la función aptitud. Los resultados obtenidos para los objetivos se presentan en la Tabla 29.

Tabla 29. Resultados de los scaffolds sinusoidales

Nº	VAR1 (mm)	VAR2 (mm)	VAR3	VAR4	VAR5	R1, R2 (MPa)	R3, R4 (µm)	O1 (mm^2)	O2 (%)	O3 (%)
1	0.5	0.3	5	5	1	104.37	352.90	3227.82	70.58	70.59
2	0.5	0.3	5	5	2	262.71	635.61	3369	70.59	70.59
3	0.5	0.3	5	5	3	50.06	549.62	3262.06	70.59	70.59
4	0.5	0.15	5	5	1	63.63	351.72	4289.07	69.69	69.70
5	0.5	0.15	5	5	2	226.87	627.40	4550.87	69.69	69.70
6	0.5	0.15	5	5	3	11.59	143.11	4342.28	69.69	69.70
7	0.5	0.3	5	9	1	919.59	133.02	5098.52	47.04	47.06
8	0.5	0.3	5	9	2	1241.58	231.03	5118.21	47.05	47.06
9	0.5	0.3	5	9	3	648.44	0.00	5382.06	47.05	47.06
10	0.5	0.15	5	9	1	649.85	156.46	6073.25	45.44	45.47
11	0.5	0.15	5	9	2	1182.57	245.09	6023.88	45.44	45.47
12	0.5	0.15	5	9	3	30.33	59.18	6697.14	45.40	45.46
13	0.5	0.3	9	5	1	516.05	242.82	4012.04	56.83	56.85
14	0.5	0.3	9	5	2	760.54	1241.40	4070.59	56.60	56.62
15	0.5	0.3	9	5	3	1076.63	793.13	3703.97	56.84	56.85
16	0.5	0.15	9	5	1	361.19	307.99	5235.3	55.46	55.47
17	0.5	0.15	9	5	2	652.57	255.49	5415.32	55.70	55.71
18	0.5	0.15	9	5	3	270.14	912.24	4562.62	55.69	55.71
19	0.5	0.3	9	9	1	564.31*	56.42	4609.76	22.16	22.33
20	0.5	0.3	9	9	2	644.57*	0.00	4914.08	21.58	21.91
21	0.5	0.3	9	9	3	610.35*	0.00	4546.22	21.36	22.33
22	0.5	0.15	9	9	1	496.53*	83.50	5321.86	19.75	19.85
23	0.5	0.15	9	9	2	619.24*	58.61	6048.98	20.07	20.28
24	0.5	0.15	9	9	3	409.47*	0.00	5032.75	20.08	20.28

Nº	VAR1 (mm)	VAR2 (mm)	VAR3	VAR4	VAR5	R1, R2 (MPa)	R3, R4 (μm)	O1 (mm^2)	O2 (%)	O3 (%)
25	0.1	0.3	5	9	1	446.87	606.92	3528.91	64.42	64.42
26	0.1	0.3	5	9	2	544.52	731.10	3527.88	64.42	64.42
27	0.1	0.3	5	9	3	551.10	594.93	3142.19	64.42	64.42
28	0.1	0.15	5	9	1	330.80	732.59	4540.49	63.44	63.44
29	0.1	0.15	5	9	2	517.13	827.24	4521.83	63.44	63.44
30	0.1	0.15	5	9	3	410.48	659.61	3735.44	63.44	63.44
31	0.1	0.3	9	9	1	535.30	610.42	3829.16	62.38	62.39
32	0.1	0.3	9	9	2	605.39	637.16	3841.97	62.38	62.39
33	0.1	0.3	9	9	3	606.23	564.31	3401.87	62.39	62.39
34	0.1	0.15	9	9	1	430.24	705.68	4865.84	61.33	61.33
35	0.1	0.15	9	9	2	573.01	748.56	4886.62	61.33	61.33
36	0.1	0.15	9	9	3	496.30	631.74	3999.08	61.33	61.33
37	0.1	0.3	5	5	1	114.34	1887.70	2121.09	80.24	80.24
38	0.1	0.3	5	5	2	165.76	2015.30	2118.5	80.23	80.24
39	0.1	0.3	5	5	3	148.50	1842.70	1829.11	81.16	81.17
40	0.1	0.15	5	5	1	88.24	1991.70	2855.25	79.69	79.69
41	0.1	0.15	5	5	2	155.94	2101.40	2847.43	79.69	79.69
42	0.1	0.15	5	5	3	112.98	1815.60	2268.07	81.07	81.07
43	0.1	0.3	9	5	1	147.83	1587.90	2315.75	79.10	79.10
44	0.1	0.3	9	5	2	182.56	1776.70	2318.08	79.10	79.10
45	0.1	0.3	9	5	3	176.06	1559.40	1987.34	80.11	80.12
46	0.1	0.15	9	5	1	119.39	1860.20	3110.26	78.52	78.52
47	0.1	0.15	9	5	2	172.49	1755.30	3119.82	78.52	78.52
48	0.1	0.15	9	5	3	141.86	1706.30	2453.67	80.01	80.01

*Resultados del módulo de Young equivalente obtenidos por interpolación

En el caso de las muestras sinusoidales de 19 a 24, presentaron problemas de simulación, por lo que los resultados del módulo de Young tuvieron que predecirse con el metamodelo creado para el resto de resultados, realizando una interpolación.

La función aptitud utilizada para este caso se presenta en la Ecuación (14). En ella se pretende hallar por valores máximos de los objetivos, en este caso, de superficie (O1) e interconectividad (O2), ponderándolos al 50% y normalizando sus valores al rango 0-1. Además, durante el proceso, se le aplica un factor de penalización a las configuraciones que no cumplen con las restricciones, reduciendo su valor de aptitud.

$$\text{Función de aptitud} = \text{máx}\left(0.5 \cdot \frac{O1}{\overline{O1}} + 0.5 \cdot \frac{O2}{\overline{O2}}\right) \qquad (14)$$

Tras introducir las cuarenta y ocho muestras de los scaffolds sinusoidales en el algoritmo genético, la quinta muestra ha resultado ser la óptima. Se trata de un scaffold con cinco filamentos por capa y una altura de capa de 0.15 mm. Cabe destacar que la

ecuación matemática de los filamentos corresponde a una ecuación seno, con cinco ciclos y una amplitud de 0.5 mm. Por tanto, para el diseño de los scaffolds sinusoidales reflejados se utilizará la ecuación seno.

7.2.2.9. Optimización final

A partir de este punto, se considerarán tres grupos de scaffolds, según su patrón: scaffold sinusoidal, scaffold sinusoidal reflejado y giroide. Las variables y el valor de cada una de ellas, así como las restricciones y objetivos asignados a cada uno de los grupos para la optimización se presentan en la Tabla 30.

Tabla 30. Diseño de experimentos

Grupo	Variables		Valores	Restricciones		Objetivos	
Sinusoidal	VAR1 VAR2 VAR3 VAR4 VAR5	A (mm) altura_capa (mm) B n_filamentos Patrón de relleno	0.1, 0.5 0.15, 0.3 5, 9 5, 9 1 (cos), 2 (sen), 3 (sen-cos)	R1,R2 R3,R4 	Módulo de Young Tamaño de poro 	O1 O2 O3 	Superficie Interconectividad Porosidad
Sinusoidal reflejado	VAR1 VAR2 VAR3 VAR4	A (mm) altura_capa (mm) B n_filamentos	0.1, 0.5 0.15, 0.3 5, 9 5, 9	R1,R2 R3,R4	Módulo de Young Tamaño de poro	O1 O2 O3 	Superficie Interconectividad Porosidad
Giroide	VAR1 VAR2	altura_capa (mm) Porosidad (%)	0.15, 0.3 50, 70	R1,R2 R3,R4	Módulo de Young Tamaño de poro	O1 O2 	Superficie Interconectividad

En el caso de los scaffolds sinusoidales reflejados, el número de variables se redujo a cuatro, frente a las cinco variables del diseño anterior, porque el patrón de relleno óptimo se utilizó como predeterminado en esta nueva configuración. Por lo tanto, el número de muestras diseñadas se redujo a dieciséis, como se muestra en la Tabla 31. Además, también se presentan los resultados del módulo elástico equivalente (R1, R2), tamaño de poro (R3, R4), superficie (O1), interconectividad (O2) y porosidad (O3), apareciendo sombreadas las muestras que no cumplen con las restricciones.

Tabla 31. Resultados de los scaffolds sinusoidales reflejados

N°	VAR1 (mm)	VAR2 (mm)	VAR3	VAR4	R1, R2 (MPa)	R3, R4 (μm)	O1 (mm^2)	O2 (%)	O3 (%)
1	0.1	0.15	5	5	384.75	1095.5	3063.87	75.61	75.61
2	0.1	0.3	5	5	99.84	1015.8	2313.63	75.87	75.87
3	0.1	0.15	5	9	1071.58	1015.8	4524.43	75.62	75.62
4	0.1	0.3	5	9	667.33	559.39	3631.86	56.56	56.56
5	0.1	0.15	9	5	405.02	925.04	3342.48	73.93	73.93

Nº	VAR1 (mm)	VAR2 (mm)	VAR3	VAR4	R1, R2 (MPa)	R3, R4 (µm)	O1 (mm^2)	O2 (%)	O3 (%)
6	0.1	0.3	9	5	107.72	890.29	2515.22	74.49	74.49
7	0.1	0.15	9	9	1175.36	584.00	4822.78	53.08	53.08
8	0.1	0.3	9	9	745.06	465.47	3913.4	54.08	54.09
9	0.5	0.3	5	5	380.68	981.94	3368.68	64.12	64.13
10	0.5	0.15	5	9	2250.45	244.24	4029.16	35.18	35.25
11	0.5	0.3	5	9	1352.73	227.11	3920.02	36.08	36.10
12	0.5	0.15	9	5	1464.82	422.17	4607.59	46.78	46.87
13	0.5	0.3	9	5	968.80	269.26	3765.03	47.39	47.40
14	0.5	0.15	9	9	3413.70	0	861.47	12.65	12.67
15	0.5	0.3	9	9	3365.57	0	960.14	10.56	10.57
16	0.1	0.15	5	5	384.75	1095.5	3063.87	75.61	75.61

Tras aplicar el algoritmo genético y, tomando como función de aptitud la misma empleada en los scaffold sinusoidales paralelos, Ecuación (14), el diseño óptimo del scaffold sinusoidal reflejado es el de la muestra 7, con una amplitud de 0.1 mm, 9 ciclos, 9 filamentos por capa y 0.15 mm de altura de capa.

Por otro lado, al considerar únicamente dos variables de interés en el análisis de los giroides, la altura de capa y la porosidad, se obtuvieron un total de cuatro muestras. Los resultados correspondientes a estas muestras después del proceso de modelado y simulación se presentan en la Tabla 32. En este caso, solamente la muestra 3 no cumple con los requisitos de tamaño de poro.

Tabla 32. Resultados de los giroides

Nº	VAR1 (mm)	VAR2 (%)	R1, R2 (MPa)	R3, R4 (µm)	O1 (mm^2)	O2 (%)
1	0.3	50	567.45	500.83	2703.41	63.77
2	0.3	70	239.37	408.31	1702.67	77.45
3	0.15	50	1234.88	732.89	2090.98	42.11
4	0.15	70	534.22	545.91	1460.05	62.82

Al aplicar la optimización por algoritmos genéticos, aplicando la misma función aptitud que en los casos anteriores, Ecuación (14), la configuración de la muestra 1 ha resultado ser la óptima, altura de capa 0.3 mm y porosidad 50%.

Los resultados del algoritmo genético, es decir, las combinaciones óptimas para cada configuración, ya habían sido modeladas y simuladas en todos los casos (ya incluidas en el diseño de experimentos). Por lo tanto, no fue necesario realizar cuatro iteraciones en la optimización ni verificar el MAPE como criterio de parada.

Por otro lado, para comparar el rendimiento biológico de las muestras y verificar el correcto funcionamiento del proceso de optimización, también se determinó la peor

combinación evaluada entre las que cumplían con las restricciones, para cada caso. Las configuraciones óptimas y no óptimas, ordenadas desde la más a la menos óptima, de acuerdo al valor de la función de aptitud, se muestran en la Tabla 33 y en la Figura 64. Con esto, se consigue combinar los tres grupos y decidir el mejor patrón curvo para la proliferación celular, de manera teórica.

Tabla 33. Muestras óptimas y no óptimas

Configuración (n° muestra)	Módulo Young (MPa)	Tamaño poro (µm)	Superficie (mm^2)	Interconectividad (%)	Porosidad (%)	Función aptitud	Nombre
Reflejado (7)	1175.36	584.00	4822.78	53.08	53.08	1.23	REF_op
Giroide (1)	567.45	500.83	2703.41	63.77	50.00	1.20	GYR_op
Sinusoidal (5)	226.87	627.40	4550.87	69.69	69.70	1.15	SIN_op
Reflejado (4)	667.33	559.39	3631.86	56.56	56.56	1.08	REF_nop
Sinusoidal (27)	551.10	594.93	3142.19	64.42	64.42	0.93	SIN_nop
Giroide (4)	534.22	545.91	1460.05	62.82	70.00	0.88	GYR_nop

Analizando el resultado final, la configuración óptima, según la metodología seguida, es la sinusoidal reflejada, con una amplitud de 0.1 mm y 9 ciclos, y 9 filamentos en cada capa de 0.15 mm de altura. El módulo elástico equivalente del modelo es de 1175.36 MPa, que cumple con el requisito de ser entre 10 y 1500 MPa. El tamaño de poro es de 584 µm, que se encuentra entre 300 y 700 µm y está cerca del óptimo, 500 µm. Además, se obtiene que la superficie disponible para que se depositen las células es de 4822.78 mm^2, y los valores de interconectividad de poros y porosidad son del 53.08%.

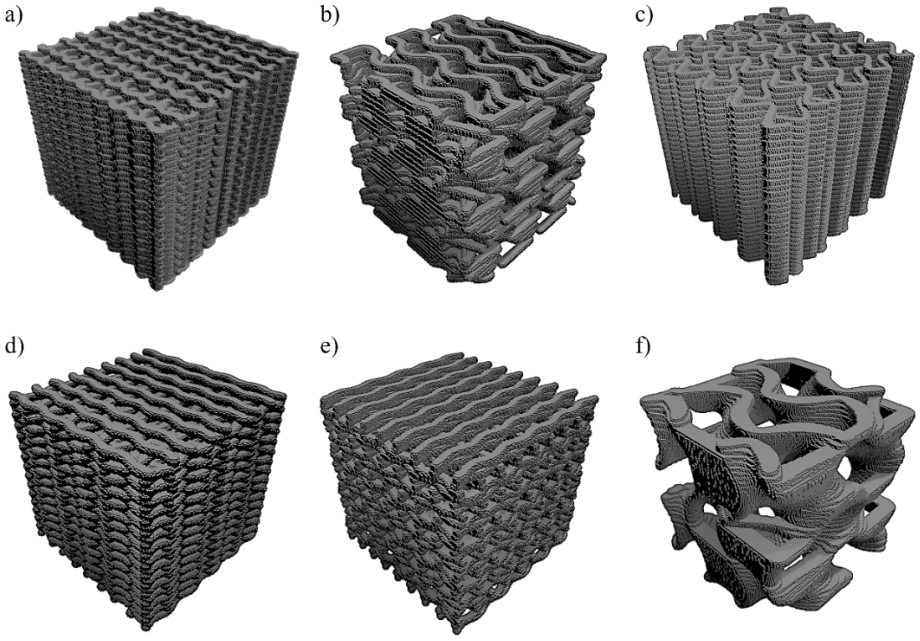


Figura 64. Geometría de scaffolds óptimos y no óptimos: REF_op (a), GYR_op (b), SIN_op (c), REF_nop (d), SIN_nop (e) y GYR_nop (f)

La geometría, proyección en Z y los resultados de la simulación de la configuración óptima se muestran en la Figura 65.

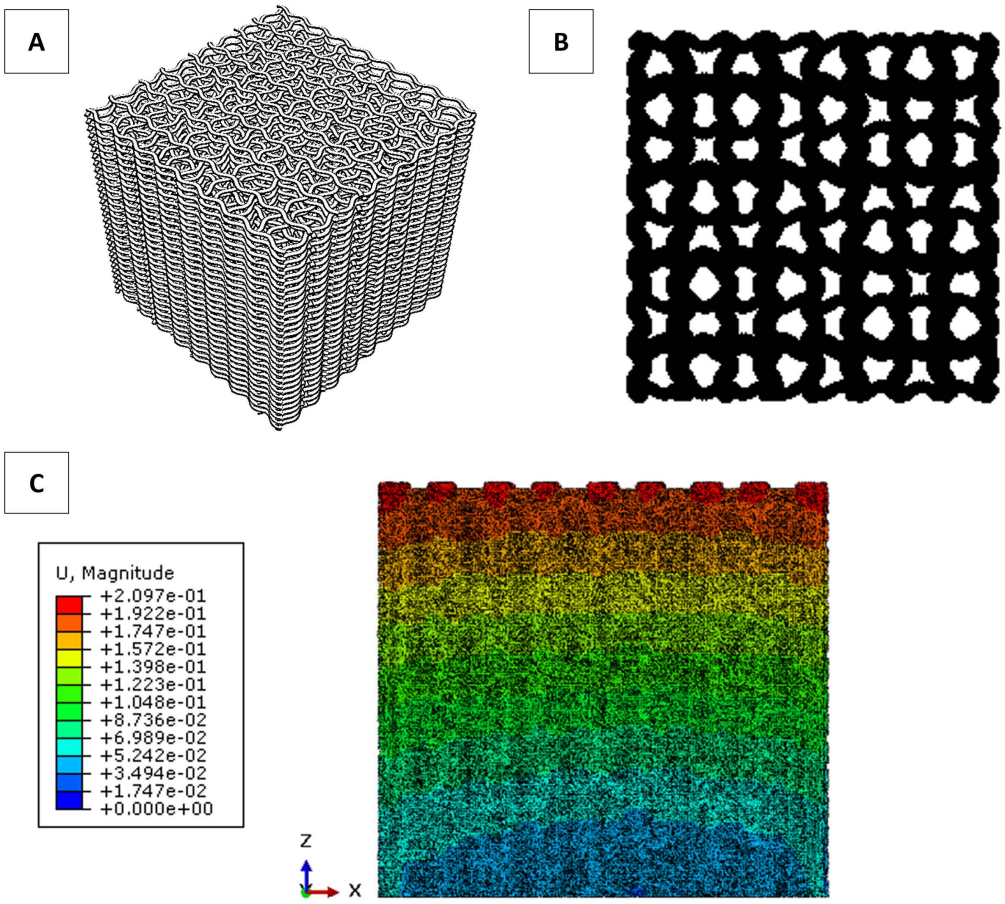


Figura 65. Configuración óptima del scaffold cúbico. (A) Geometría. (B) Proyección Z. (C) Desplazamiento del scaffold deformado en el plano XZ.

7.2.3. Ensayos biológicos

Con el fin de validar la metodología de optimización, se procedió a imprimir en PLA las seis muestras descritas en la Tabla 33, es decir, el diseño óptimo de cada uno de los grupos y el diseño denominado no óptimo, que corresponde a la muestra menos óptima de aquellas que cumplían con las restricciones. En la Figura 66, a modo de ejemplo, se muestra el modelo y la pieza impresa de uno de los scaffolds (REF_op). El diseño sinusoidal incluye travels alejados de la pieza, para evitar que el filamento que sigue saliendo de la boquilla en estos movimientos se adhieran a los filamentos depositados y modifiquen la estructura porosa; así que, deben eliminarse a posteriori, quedando como en la Figura 66(C).

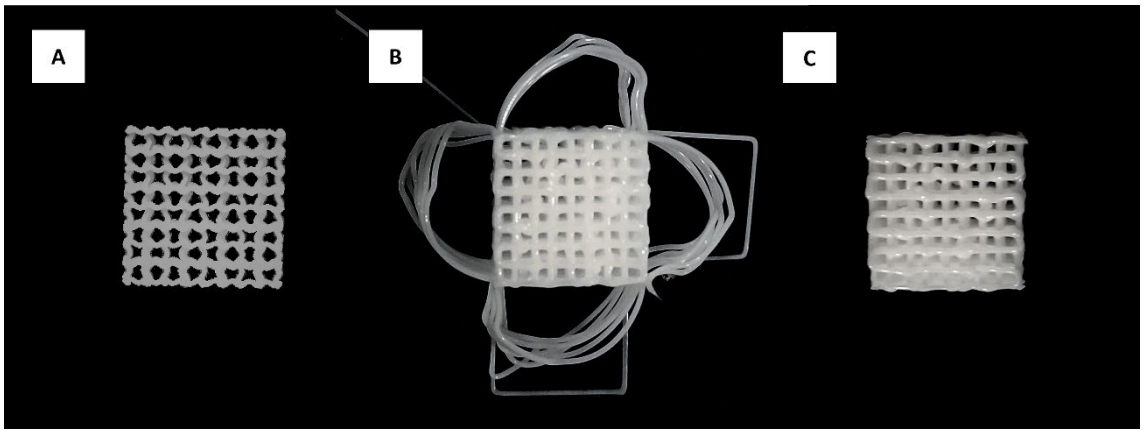


Figura 66. Ejemplo de scaffold REF_op. (A) Modelo. (B) Pieza impresa con travels. (C) Pieza impresa sin travels.

Debido a la naturaleza del ensayo, se decidió imprimir las muestras hasta la altura de 3.5 mm, en la cual se obtiene una representación aceptable del scaffold en su totalidad, a la vez que se ahorra medio de cultivo y reactivo.

7.2.3.1. Cultivo celular y siembra

Las células madre mesenquimales de médula ósea humana (CMM-MO) (SCC034, Merck) se cultivaron en un medio de formulación bajo en suero (Human Mesenchymal-LS Expansion Medium, SCM023, Merck), suplementado con un 1% de solución antibiótica-antimicótica (A5955-100ML, Merck). Las células se cultivaron en matraces de 75 cm^2 (T75, Sarstedt) previamente recubiertos con una solución de gelatina al 0.1% (ES-006-B, EmbryoMax, Merck). Para el desprendimiento celular, se utilizó tripsina-EDTA en solución salina equilibrada de Hank (0.25% de tripsina y 1 mM de EDTA, sin Ca^{2+} ni Mg^{2+}, SM-2003-C, Merck) a una confluencia del 80-90%.

Los scaffolds a sembrar tenían 3.5 mm de altura y una base cuadrada de 9x9 mm. Se fabricaron mediante MEX en una impresora 3D Ender-3, con una temperatura de cama de 60 °C y una temperatura de boquilla de 210 °C. Además, el diámetro del filamento de PLA Smartfil es 1.75 mm y el ancho de extrusión se definió como 0.48 mm.

Se utilizaron cuatro réplicas por grupo: SIN_op, GYR_op, REF_op, SIN_nop, GYR_nop y REF_nop. SIN_op y SIN_nop corresponden a los scaffolds óptimos y no óptimos de patrón sinusoidal (filamentos paralelos); REF_op y REF_nop, a los scaffolds de patrón sinusoidal reflejado; y GYR_op y GYR_nop, a los giroides.

Los scaffolds se sumergieron en una solución de etanol al 75% v/v, se colocaron en una placa de 24 pocillos no tratada (Thermo Scientific™ Nunc™) y posteriormente

se expusieron a luz UV durante una hora, bajo la campana extractora. A continuación, se lavaron dos veces con medio de cultivo para eliminar cualquier rastro de etanol. Tras colocar las muestras en una nueva placa de pocillos, los scaffolds se mantuvieron en una incubadora (a 37 °C y 5% de CO_2) sumergidos en 1 mL de medio de cultivo fresco durante una hora. Tras este periodo de hidratación de las muestras, el medio de cultivo de cada una de ellas se sustituyó por una suspensión de 1 mL que contenía 30000 CMM-MO. El medio de cultivo se cambió después de las primeras 24 h.

7.2.3.2. Evaluación de la actividad metabólica celular

La actividad metabólica celular se evaluó mediante el protocolo CCK-8 (Cell Counting Kit-8, Dojindo Molecular Technologies). Tras 4 días de cultivo celular, los scaffolds se transfirieron a una nueva placa de 24 pocillos no tratada y se añadieron 0.7 mL de una solución al 10% v/v del reactivo CCK-8 en medio de cultivo. La misma solución se vertió en 4 pocillos vacíos que se utilizaron como controles negativos. La placa se incubó a 37 °C y 5% de CO_2 durante 3.5 h. A continuación, se transfirieron dos alícuotas de 100 µl de cada pocillo a una placa lectora de 96 pocillos (Thermo Scientific™ Nunc™-MicroWell™). Se utilizó un lector BioTek ELx800 (Bio Tek Instruments Inc., Winooski, VT, EE.UU.) para medir la absorbancia de las muestras a una longitud de onda de excitación de 450 nm. Por último, el resultado del grupo de control negativo se restó de la absorbancia obtenida para cada grupo de muestras analizado.

7.2.3.3. Análisis estadístico

El análisis estadístico se realizó utilizando Matlab (MATLAB and Statistics Toolbox Release 2021a). Para el análisis de datos se utilizó la prueba de Kruskal-Wallis. El nivel de significancia se fijó en *$p<0.05$, **$p<0.01$ y ***$p<0.001$, para diferencias estadísticamente significativas, altamente significativas y muy altamente significativas, respectivamente. Las figuras de los resultados estadísticos muestran los valores medios de cada grupo y sus desviaciones estándar representadas con barras de error.

7.2.3.4. Resultados

Según los resultados del ensayo CCK-8 (Figura 67), tras 5 días de cultivo, las células fueron capaces de adherirse y proliferar en todos los grupos analizados. Los valores medios más altos de absorbancia, y por tanto las actividades metabólicas más altas de las CMM-MO cultivadas en los scaffolds, se observaron en los grupos GYR_op y REF_op. Ambos grupos mostraron diferencias estadísticamente significativas con

respecto a sus homólogos no optimizados, es decir, los grupos GYR_nop y REF_nop. Cabe destacar que se obtuvo una diferencia estadísticamente muy significativa (*p<0.001) entre los scaffolds REF_op y GYR_nop, que mostraron los valores medios de absorbancia más altos y más bajos, respectivamente.

Atendiendo a los resultados del ensayo biológico, puede afirmarse que la optimización es válida, ya que los diseños óptimos han presentado datos de absorbancia superior a los no óptimos de cada grupo, aunque estas diferencias sean significativas solamente en los scaffolds sinusoidales reflejados y giroides. Además, el que ha presentado un mayor valor de absorbancia es el scaffold con la mejor configuración entre todos los grupos, según la metodología.

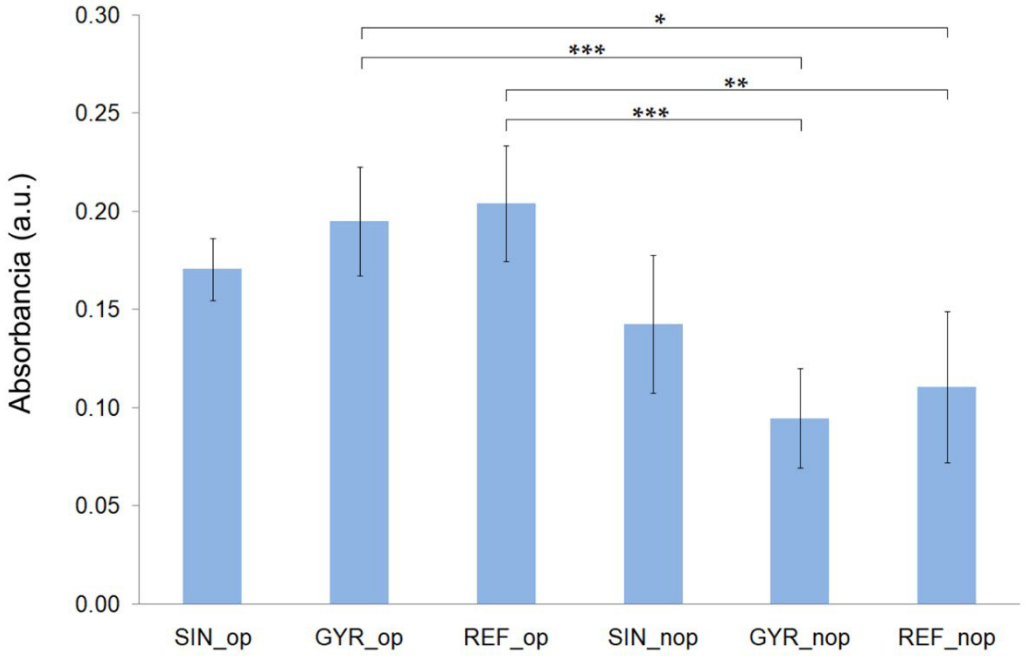


*Figura 67. Actividad metabólica de las CMM-MO cultivadas en los diferentes grupos de scaffolds, determinada por el ensayo CCK-8 en el día 5 (*p<0.05, **p<0.01 y ***p<0.001)*

7.2.4. Conclusiones

En resumen, la aplicación práctica de las metodologías desarrolladas en la tesis a scaffolds cúbicos de PLA ha sido viable. Además, la aplicación del ensayo biológico a las piezas impresas ha demostrado la validez de la optimización de los modelos 3D, obtenidos mediante VOLCO, a partir de ficheros G-code, y simulados mediante AEF en Abaqus/CAE 6.14-1.

En esta aplicación, se pretendían obtener scaffolds cúbicos de 9x9x9 mm, impresos en PLA Smartfil de 1.75 mm de diámetro, en la impresora Ender-3, con una

temperatura de cama de 60 °C y de boquilla de 210 °C, considerando su ancho de extrusión como 0.48 mm. Para el resto de parámetros geométricos y de impresión, se aplicó la metodología de optimización, comparando la influencia del patrón de relleno, la altura de capa y la porosidad intrínseca de las estructuras (seleccionable en los giroides desde el laminador, pero obtenida como resultado en los scaffolds sinusoidales, según la disposición de los filamentos).

El resultado de la optimización ha sugerido que el mejor patrón de relleno para las proliferación y adherencia celular y, por tanto, la regeneración de tejido óseo trabecular, es el sinusoidal reflejado. Es decir, el patrón curvilíneo cuyos filamentos presentan un trazado que corresponde a la ecuación del seno y son simétricos con respecto a su contiguo. Además, el valor de la función aptitud de la muestra 7 es superior al resto, concluyendo que la configuración óptima es la que presenta una altura de capa es de 0.15 mm, una longitud de onda de 0.1 mm, 9 ciclos en cada filamentos y 9 filamentos en cada capa. La geometría de una capa del scaffold óptimo se presenta en la Figura 68.

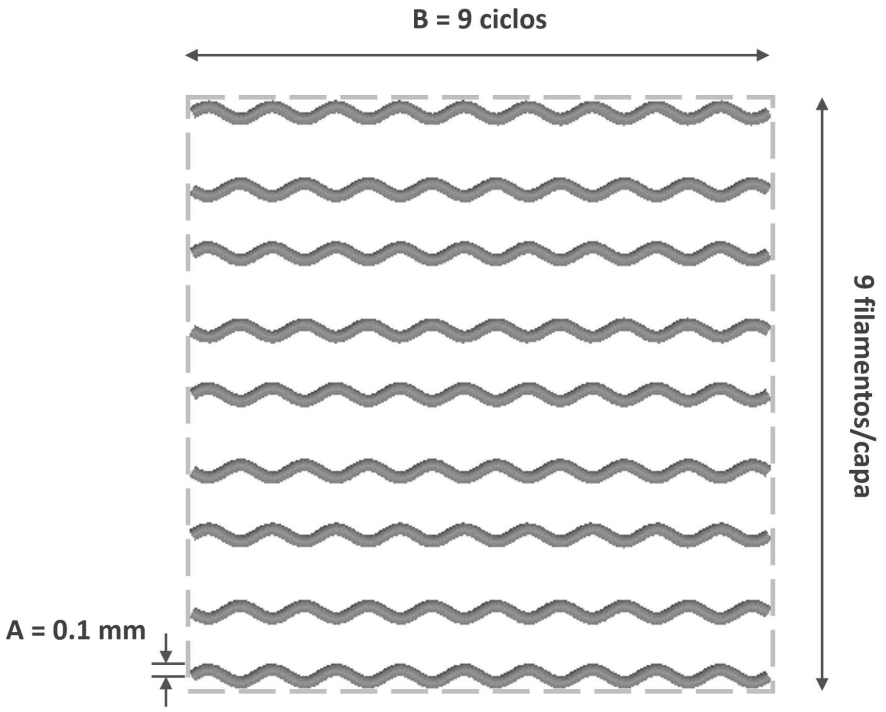


Figura 68. Primera capa de la configuración óptima (REF_op)

Capítulo 8. Conclusiones, difusión y líneas futuras

8.1. CONCLUSIONES

En conclusión, a lo largo de esta investigación se ha llevado a cabo un exhaustivo análisis y estudio de la optimización de parámetros geométricos para la fabricación de piezas mediante la tecnología MEX. Se han explorado diferentes enfoques, metodologías y técnicas para abordar la falta de herramientas para la predicción y optimización de piezas impresas con esta tecnología (MEX), sin tener que recurrir a ensayos experimentales para alcanzar los objetivos propuestos.

A partir del análisis detallado de los resultados obtenidos al aplicar las metodologías de modelado y optimización, se han identificado patrones, relaciones y limitaciones que han proporcionado una sólida base de conocimiento. Estos hallazgos han permitido extraer conclusiones significativas y brindar nuevas perspectivas, especialmente en el ámbito de la ingeniería tisular. A continuación, se presentan las principales conclusiones derivadas de este estudio, destacando su relevancia y contribuciones al campo de la ingeniería:

- Ha sido posible crear una nueva herramienta de modelado 3D basada en la geometría (DECODE) para piezas a fabricar mediante MEX, a partir de un archivo G-code con comentarios. Dicha metodología se basa en replicar, mediante operaciones de barrido en software CAD, las trayectorias de filamento depositado de acuerdo al código G. Actualmente, se puede iniciar el proceso de modelado con códigos G generados desde Slic3r 1.3.0, Simplify3D 4.1.1 o FullControl G-code Designer, utilizando una combinación de Matlab R2021a y Abaqus/CAE 6.14-1 para la generación de modelos. Por tanto, se ha cumplido así con el primer objetivo específico de esta tesis, creando una metodología para el modelado de piezas a partir de código G que posibilita su posterior simulación.
- Tras comparar varias alternativas de software para el modelado basado en geometría, DECODE ha resultado ser la mejor alternativa por la capacidad de automatizar el proceso y evitar errores al generar el modelo (auto-intersecciones, cierres de trayectoria defectuosos, etc.).
- En la comparación de varias alternativas de modelado basadas en diferentes enfoques, geometría (DECODE) o vóxeles (VOLCO), se ha concluido que ambas son

válidas para simular la deposición de filamentos de la pieza impresa; pero la mejor alternativa para modelar piezas de pequeño tamaño y patrón complejo es VOLCO; mientras que, para piezas con un entramado simple y de mayor tamaño, DECODE es más eficiente, el proceso es más rápido y simple.

- Estas técnicas de modelado combinadas con el análisis de elementos finitos conforman unas herramientas de gran utilidad para la optimización de piezas antes de ser fabricadas mediante MEX, reduciendo así el trabajo experimental y mejorando la rentabilidad del proceso.
- Para aprovechar el potencial que ofrecen estas herramientas de modelado y simulación en la optimización de parámetros geométricos en piezas MEX, se ha propuesto una metodología de optimización (segundo objetivo específico de esta tesis). Esta combina el diseño de experimentos factorial (niveles o intensidad de muestreo adaptados a cada caso) con el uso de metamodelos (Kriging) y algoritmos genéticos, consiguiendo una metodología robusta para diferentes problemas, como suele ocurrir con los algoritmos genéticos, y viable en términos de recursos computacionales gracias a la combinación de diseño de experimentos y el uso de metamodelos. El diseño de experimentos inicial permite obtener, a partir de las herramientas de modelado y simulación, una base de datos de comportamiento del sistema, la cual se usa posteriormente para ajustar o entrenar los metamodelos Kriging (interpolación). Estos, a su vez, se emplean para estimar internamente las respuestas de los diferentes individuos o diseños propuestos durante la evolución del algoritmo genético (sin tener que modelar/simular cada uno de ellos). Además, la metodología propuesta incluye varias iteraciones de optimización, de modo que el diseño óptimo de cada iteración es evaluado y añadido a la base de datos, mejorando así la precisión de los metamodelos en cada iteración.
- Las aplicaciones prácticas llevadas cabo demuestran la validez de la metodología de optimización, siempre que se definan correctamente las variables y objetivos a optimizar. Es necesario establecer un diseño de experimentos adecuado y seguir los pasos descritos, para obtener el diseño óptimo, después de la aplicación de algoritmos genéticos.
- Las metodologías de modelado y optimización son aplicables a piezas relativamente simples, como se vio en el caso de las probetas de ensayo, para la toma de decisiones sobre los parámetros geométricos que se utilizarán al generar el código G. En estas aplicaciones, se debe tener en cuenta que los valores de las simulaciones y los ensayos experimentales pueden diferir si las propiedades del

material en el modelo de simulación no se corresponden con las del material real, como puede ocurrir cuando se utilizan las características de catálogo del material de fabricación (normalmente con propiedades bastante más favorables a las reales). Sin embargo, incluso en estos casos, la metodología de modelado y optimización permite deducir qué piezas serán mejores en base a los objetivos de la optimización.

- Se ha encontrado que las metodologías de modelado y optimización pueden contribuir significativamente a la fabricación de estructuras porosas en la ingeniería tisular. En esta investigación se ha conseguido predecir la mejor configuración (entre varios patrones con curvatura) para la proliferación celular y regeneración de tejido óseo trabecular, haciendo uso de estas metodologías y sin necesidad de imprimir y ensayar los scaffolds. En este sentido, se ha combinado el uso de un laminador innovador (como es el FullControl GCode Designer, con capacidad para definir patrones sinusoidales), junto con un método de modelado basado en vóxeles (VOLCO) y simulación mediante AEF, así como varias herramientas desarrolladas específicamente para determinar algunas variables de respuesta (porosidad e interconectividad). Todas estas herramientas, combinadas e integradas correctamente en la metodología de optimización y con una correcta planificación y definición del problema de optimización, han permitido encontrar el diseño óptimo para la proliferación celular y regeneración de tejido óseo trabecular, lo cual se ha contrastado con ensayos biológicos.

8.2. DIFUSIÓN DE RESULTADOS

En el presente apartado, se presenta una visión general de las actividades de difusión llevadas a cabo como parte de la tesis doctoral. Durante el desarrollo de la investigación, se han realizado diversas presentaciones y se han redactado varios artículos científicos que abordan los resultados y hallazgos obtenidos. Estas acciones de difusión han permitido compartir los avances y contribuciones realizadas en el campo de estudio, así como fomentar el intercambio de conocimientos con la comunidad científica y académica.

8.2.1. Artículos

Durante el desarrollo de esta tesis se ha publicado un artículo en una revista indexada y se ha redactado un segundo, que se desea publicar próximamente, titulado *Curve-Based Infill Pattern Optimization for 3D Printed Polymeric Scaffolds to Promote*

Trabecular Bone Regeneration.

8.2.1.1. Artículo de comparación de metodologías

Tipo: Artículo científico.

Nombre de la publicación: Comparison of CAD and Voxel-Based Modelling Methodologies for the Mechanical Simulation of Extrusion-Based 3D Printed Scaffolds.

Autores: Gisela Vega, Rubén Paz, Andrew Gleadall, Mario Monzón, María Elena Alemán Domínguez.

Revista: Materials.

Editorial: MDPI - Multidisciplinary Digital Publishing Institute.

Año: 2021.

DOI: 10.3390/ma14195670.

8.2.2. Ponencias en congresos

Los resultados de esta investigación se han presentado en cinco congresos internacionales.

8.2.2.1. 2019 Internacional Conference on Materials & Nanomaterials

Título: Comparison between geometry-based and voxel-based modelling techniques for the prediction of the mechanical behaviour of polymeric scaffolds obtained by extrusion-based Additive Manufacturing.

Autores: Gisela del Carmen Vega Rodríguez, Rubén Paz Hernández, Mario Domingo Monzón Verona, Andy Gledall.

Nombre del congreso: 2019 Internacional Conference on Materials & Nanomaterials.

Entidad organizadora: Sciknowledge European Conferences.

Lugar: París, Île de France, Francia.

Año: 2019.

8.2.2.2. Autumn Webinar BAMOS 2020

Título: Comparison of 3D modelling techniques for the prediction of the mechanical behaviour of polymeric scaffolds obtained by additive manufacturing.

Autores: Gisela del Carmen Vega.

Nombre del congreso: Autumn Webinar BAMOS 2020.

Entidad organizadora: Universidad de las Palmas de Gran Canaria.

Lugar: Las Palmas de Gran Canaria, Canarias, España.

Año: 2020.

8.2.2.3. Brain IT Brain Revealed: innovative Technologies in Neurosurgery Study

Título: Modelling methodologies for Finite Element Analysis of material extrusion 3D printed scaffolds.

Autores: Gisela del Carmen Vega Rodríguez, Rubén Paz Hernández, Mario Monzón Verona, Andrew Gleadall.

Nombre del congreso: Brain IT Brain Revealed: innovative Technologies in Neurosurgery Study.

Entidad organizadora: Sibiu Emergency County Clinical Hospital.

Lugar: Sibiu, Rumanía.

Año: 2021.

8.2.2.4. 10th International Conference on Biomedical Engineering and Biotechnology

Título: Modelling Methodologies for the Mechanical Simulation of Polymeric Scaffolds obtained by Material Extrusion Additive Manufacturing.

Autores: Gisela del Carmen Vega Rodríguez, Rubén Paz Hernández, Mario Domingo Monzón Verona, Andrew Gleadall.

Nombre del congreso: 10th International Conference on Biomedical Engineering and Biotechnology (ICBEB 2021).

Entidad organizadora: ICBEB2021 Organizing Committee.

Lugar: China.

Año: 2021.

8.2.2.5. 2nd International Conference on Cells and Extracellular Templates

Título: Custom-made Implants and Additive Manufacturing in the International and European Medical Device.

Autores: Mario Monzón, Rubén Paz, Pablo R. Bordón, Ricardo Donate, Joshua García-Montagut, Gisela Vega.

Nombre del congreso: 2nd International Conference on Cells and Extracellular Templates (CET 2023).

Entidad organizadora: Università degli Studi Niccolò Cusano.

Lugar: Roma, Italia.

Año: 2023.

8.3. LÍNEAS FUTURAS

La investigación llevada a cabo en esta tesis doctoral ha proporcionado importantes contribuciones al campo de estudio de la ingeniería, abriendo nuevas vías de investigación. Sin embargo, es evidente que aún existen áreas que requieren mayor exploración y desarrollo para avanzar en la comprensión y aplicación de los conceptos estudiados. Por tanto, en este apartado se presentan las líneas futuras de investigación que pueden ser consideradas para ampliar el conocimiento y seguir impulsando el avance en el área abordada en esta tesis doctoral.

Las líneas futuras propuestas se enfocan en diferentes aspectos que han surgido como resultado de esta investigación y que han revelado oportunidades prometedoras para futuros estudios. Entre estas líneas, se encuentran:

- Optimización y mejora de subrutinas y modelos: Es posible investigar la optimización de las subrutinas utilizadas en las diferentes metodologías de modelado, así como mejorar los modelos matemáticos y de simulación empleados. El objetivo principal sería obtener modelos de piezas de mayor dimensión de manera eficiente y así, explorar nuevas posibilidades de aplicación.
- Investigación de aplicaciones en diferentes sectores: En este trabajo se ha comprobado la eficiencia de los procesos para la aplicación en la ingeniería tisular. Sin embargo, se puede explorar la aplicación de los conceptos estudiados en otros sectores de la ingeniería. Esto abriría la puerta a nuevas áreas de aplicación y permitiría evaluar su eficacia y rendimiento en diferentes contextos.

- Análisis de casos de estudio más complejos: Se podría llevar a cabo un análisis más detallado y exhaustivo de casos más complejos, que implicaran el análisis de otras variables de entrada y objetivos en la optimización. Esto incluiría la recolección de datos adicionales, la realización de pruebas experimentales más avanzadas y el análisis de resultados en un contexto más amplio.
- Exploración de nuevos materiales y técnicas: Se puede investigar la aplicación de otros materiales o el uso de técnicas innovadoras en el contexto de la ingeniería estudiada. Esto permitiría evaluar sus propiedades y comportamiento, así como analizar su viabilidad y ventajas en comparación con los enfoques estudiados.

Referencias

1. Prince, J.D. 3D Printing: An Industrial Revolution. *J. Electron. Resour. Med. Libr.* **2014**, *11*, 39–45, doi:10.1080/15424065.2014.877247.
2. Tofail, S.A.M.; Koumoulos, E.P.; Bandyopadhyay, A.; Bose, S.; O'Donoghue, L.; Charitidis, C. Additive manufacturing: scientific and technological challenges, market uptake and opportunities. *Mater. Today* **2018**, *21*, 22–37, doi:10.1016/j.mattod.2017.07.001.
3. Singh, T.; Kumar, S.; Sehgal, S. 3D printing of engineering materials: A state of the art review. *Mater. Today Proc.* **2020**, *28*, 1927–1931, doi:10.1016/j.matpr.2020.05.334.
4. Hendrikson, W.J.; van Blitterswijk, C.A.; Rouwkema, J.; Moroni, L. The Use of Finite Element Analyses to Design and Fabricate Three-Dimensional Scaffolds for Skeletal Tissue engineering. *Front. Bioeng. Biotechnol.* **2017**, *5*, 1–13, doi:10.3389/fbioe.2017.00030.
5. Velasco, M.A.; Narváez-Tovar, C.A.; Garzón-Alvarado, D.A. Design, Materials, and Mechanobiology of Biodegradable Scaffolds for Bone Tissue Engineering. *Biomed Res. Int.* **2015**, *2015*.
6. Naghieh, S.; Karamooz Ravari, M.R.; Badrossamay, M.; Foroozmehr, E.; Kadkhodaei, M.; Karamooz Ravari, M.R.; Badrossamay, M.; Foroozmehr, E.; Kadkhodaei, M. Numerical investigation of the mechanical properties of the additive manufactured bone scaffolds fabricated by FDM: The effect of layer penetration and post-heating. *J. Mech. Behav. Biomed. Mater.* **2016**, *59*, 241–250, doi:10.1016/j.jmbbm.2016.01.031.
7. Seol, Y.-J.; Kang, H.-W.; Lee, S.J.; Atala, A.; Yoo, J.J. Bioprinting technology and its applications. *Eur. J. Cardio-Thoracic Surg. 46* **2014**, *46*, 342–348, doi:10.1093/ejcts/ezu148.
8. Ribeiro, J.F.M.; Oliveira, S.M.; Alves, J.L.; Pedro, A.J.; Reis, R.L.; Fernandes, E.M.; Mano, J.F. Structural monitoring and modeling of the mechanical deformation of three-dimensional printed poly (ε-caprolactone) scaffolds. *Biofabrication* **2017**, *9*.
9. Qu, H. Additive manufacturing for bone tissue engineering scaffolds. *Mater. Today Commun.* **2020**, *24*, 1–16, doi:10.1016/j.mtcomm.2020.101024.

10. Saygili, E.; Dogan-Gurbuz, A.A.; Yesil-Celiktas, O.; Draz, M.S. 3D bioprinting : A powerful tool to leverage tissue engineering and microbial systems. *Bioprinting* 2020, *18*, e00071, doi:10.1016/j.bprint.2019.e00071.

11. Rahim, T.N.A.T.; Abdullah, A.M.; Md Akil, H. Recent Developments in Fused Deposition Modeling-Based 3D Printing of Polymers and Their Composites. *Polym. Rev.* 2019, *59*, 589–624, doi:10.1080/15583724.2019.1597883.

12. Dudescu, C.; Racz, L. Effects of Raster Orientation, Infill Rate and Infill Pattern on the Mechanical Properties of 3D Printed Materials. *ACTA Univ. Cibiniensis* 2017, *69*, 23–30, doi:10.1515/aucts-2017-0004.

13. Singh, D.; Babbar, A.; Jain, V.; Gupta, D.; Saxena, S.; Dwibedi, V. Synthesis , characterization , and bioactivity investigation of biomimetic biodegradable PLA scaffold fabricated by fused filament fabrication process. *J. Brazilian Soc. Mech. Sci. Eng.* 2019, *41*, 1–13, doi:10.1007/s40430-019-1625-y.

14. Temple, J.P.; Hutton, D.L.; Hung, B.P.; Huri, P.Y.; Cook, C.A.; Kondragunta, R.; Jia, X.; Grayson, W.L. Engineering anatomically shaped vascularized bone grafts with hASCs and 3D-printed PCL scaffolds. *J. Biomed. Mater. Res. - Part A* 2014, *102*, 4317–4325, doi:10.1002/jbm.a.35107.

15. Roopavath, U.K.; Malferrari, S.; Haver, A. Van; Verstreken, F.; Rath, S.N.; Kalaskar, D.M. Optimization of extrusion based ceramic 3D printing process for complex bony designs. *Mater. Des.* 2019, *162*, 263–270, doi:10.1016/j.matdes.2018.11.054.

16. Loh, G.H.; Pei, E.; Harrison, D.; Monzón, M.D. An overview of functionally graded additive manufacturing. *Addit. Manuf.* 2018, *23*, 34–44, doi:10.1016/j.addma.2018.06.023.

17. McCaw, J.C.S.; Cuan-Urquizo, E. Curved-Layered Additive Manufacturing of non-planar, parametric lattice structures. *Mater. Des.* 2018, *160*, 949–963, doi:10.1016/j.matdes.2018.10.024.

18. Jung, J.W.; Lee, J.S.; Cho, D.W. Computer-Aided multiple-head 3D printing system for printing of heterogeneous organ/tissue constructs. *Sci. Rep.* 2016, *6*, 1–9, doi:10.1038/srep21685.

19. Kang, H.-W.W.; Lee, S.J.; Ko, I.K.; Kengla, C.; Yoo, J.J.; Atala, A. A 3D bioprinting system to produce human-scale tissue constructs with structural integrity. *Nat. Biotechnol.* 2016, *34*, 312–319, doi:10.1038/nbt.3413.

20. Gleadall, A. FullControl GCode Designer: Open-source software for

unconstrained design in additive manufacturing. *Addit. Manuf.* 2021, *46*, 102109, doi:10.1016/j.addma.2021.102109.

21. Sahai, N.; Saxena, K.K.; Gogoi, M. Modelling and simulation for fabrication of 3D printed polymeric porous tissue scaffolds. *Adv. Mater. Process. Technol.* 2020, 1–10, doi:10.1080/2374068X.2020.1728643.

22. Souness, A.; Zamboni, F.; Walker, G.M.; Collins, M.N. Influence of scaffold design on 3D printed cell constructs. *J. Biomed. Mater. Res. Part B* 2018, *106B*, 533–545, doi:10.1002/jbm.b.33863.

23. Soufivand, A.A.; Abolfathi, N.; Hashemi, S.A.; Lee, S.J. Prediction of mechanical behavior of 3D bioprinted tissue-engineered scaffolds using finite element method (FEM) analysis. *Addit. Manuf.* 2020, *33*, 2214–8604, doi:10.1016/j.addma.2020.101181.

24. Schipani, R.; Nolan, D.R.; Lally, C.; Kelly, D.J. Integrating finite element modelling and 3D printing to engineer biomimetic polymeric scaffolds for tissue engineering. *Connect. Tissue Res.* 2020, *61:2*, 174–189, doi:10.1080/03008207.2019.1656720.

25. Miranda, P.; Pajares, A.; Guiberteau, F. Finite element modeling as a tool for predicting the fracture behavior of robocast scaffolds. *Acta Biomater.* 2008, *4*, 1715–1724, doi:10.1016/j.actbio.2008.05.020.

26. Entezari, A.; Fang, J.; Sue, A.; Zhang, Z.; Swain, M. V; Li, Q. Yielding behaviors of polymeric scaffolds with implications to tissue engineering. *Mater. Lett.* 2016, *184*, 108–111, doi:10.1016/j.matlet.2016.07.149.

27. Melancon, D.; Bagheri, Z.S.; Johnston, R.B.; Liu, L.; Tanzer, M.; Pasini, D. Mechanical characterization of structurally porous biomaterials built via additive manufacturing : experiments , predictive models , and design maps for load-bearing bone replacement implants. *Acta Biomater.* 2017, *63*, 350–368, doi:10.1016/j.actbio.2017.09.013.

28. Kadkhodapour, J.; Montazerian, H.; Darabi, A.C.; Anaraki, A.P.; Ahmadi, S.M.; Zadpoor, A.A.; Schmauder, S. Failure mechanisms of additively manufactured porous biomaterials : Effects of porosity and type of unit cell. *J. Machanical Behav. Biomed. Mater.* 2015, *50*, 180–191.

29. Campoli, G.; Borleffs, M.S.; Amin Yavari, S.; Wauthle, R.; Weinans, H.; Zadpoor, A.A. Mechanical properties of open-cell metallic biomaterials manufactured

using additive manufacturing. *Mater. Des.* 2013, *49*, 957–965, doi:10.1016/j.matdes.2013.01.071.

30. Ravari, M.R.K.; Kadkhodaei, M.; Badrossamay, M.; Rezaei, R. Numerical investigation on mechanical properties of cellular lattice structures fabricated by fused deposition modeling. *Int. J. Mech. Sci.* 2014, *88*, 154–161, doi:10.1016/j.ijmecsci.2014.08.009.

31. Gleadall, A.; Ashcroft, I.; Segal, J. VOLCO: A predictive model for 3D printed microarchitecture. *Addit. Manuf.* 2018, *21*, 605–618, doi:10.1016/j.addma.2018.04.004.

32. Sanz-Herrera, J.A.; García-Aznar, J.M.; Doblaré, M. On scaffold designing for bone regeneration: A computational multiscale approach. *Acta* 2009, *5*, 219–229, doi:10.1016/j.actbio.2008.06.021.

33. Rupal, B.S.; Mostafa, K.G.; Wang, Y.; Qureshi, A.J. A Reverse CAD Approach for Estimating Geometric and Mechanical A Reverse CAD Approach for Estimating Geometric and Mechanical Behavior of FDM Conference Printed Parts Behavior of FDM Printed Parts Costing models for capacity optimization Trade-off between. *Procedia Manuf.* 2019, *34*, 535–544, doi:10.1016/j.promfg.2019.06.217.

34. Sukindar, N.A.; Dahan, A.A.A.; Halim, N.F.H.A.; Kamaruddin, S.; Sulaiman, M.H.; Ariffin, M.K.A.M. The Effect of Printing Parameters on Scaffold Structure Using Low-cost 3D Printer. *Test Eng. Manag.* 2020, *82*, 8255–8263.

35. Hollister, S.J. Porous scaffold design for tissue engineering. *Nat. Mater.* 2005, *4*, 518–524.

36. Giannitelli, S.M.; Accoto, D.; Trombetta, M.; Rainer, A. Current trends in the design of scaffolds for computer-aided tissue engineering. *Acta Biomater.* 2014, *10*, 580–594, doi:10.1016/j.actbio.2013.10.024.

37. Gómez, S.; Vlad, M.D.; López, J.; Fernández, E. Design and properties of 3D scaffolds for bone tissue engineering. *Acta Biomater.* 2016, *42*, 341–350, doi:10.1016/j.actbio.2016.06.032.

38. Wang, G.; Shen, L.; Zhao, J.; Liang, H.; Xie, D.; Tian, Z.; Wang, C. Design and Compressive Behavior of Controllable Irregular Porous Scaffolds : Based on Voronoi-Tessellation and for Additive Manufacturing. *ACS Biomater. Sci. Eng.* 2018, *4*, 719–727, doi:10.1021/acsbiomaterials.7b00916.

39. Baikerikar, P.J.; Turner, C.J. Comparison of as-built FEA simulations and

experimental results for additively manufactured dogbone geometries. In Proceedings of the Proceedings of the ASME Design Engineering Technical Conference; Cleveland, Ohio, USA, 2017; pp. 1–14.

40. Alafaghani, A.; Qattawi, A.; Alrawi, B.; Guzman, A. Experimental Optimization of Fused Deposition Modelling Processing Parameters: A Design-for-Manufacturing Approach. *Procedia Manuf.* 2017, *10*, 791–803, doi:10.1016/j.promfg.2017.07.079.

41. Cuan-Urquizo, E.; Barocio, E.; Tejada-Ortigoza, V.; Pipes, R.; Rodriguez, C.; Roman-Flores, A. Characterization of the Mechanical Properties of FFF Structures and Materials: A Review on the Experimental, Computational and Theoretical Approaches. *Materials (Basel).* 2019, *12*, 895, doi:10.3390/ma12060895.

42. Brenken, B.; Barocio, E.; Favaloro, A.; Kunc, V.; Pipes, R.B. Development and validation of extrusion deposition additive manufacturing process simulations. *Addit. Manuf.* 2019, *25*, 218–226, doi:10.1016/j.addma.2018.10.041.

43. Domingo-Espin, M.; Puigoriol-Forcada, J.M.; Garcia-Granada, A.A.; Llumà, J.; Borros, S.; Reyes, G. Mechanical property characterization and simulation of fused deposition modeling Polycarbonate parts. *Mater. Des.* 2015, *83*, 670–677, doi:10.1016/j.matdes.2015.06.074.

44. Madhukar Somireddy; Aleksander Czekanski Mechanical Characterization of Additively Manufactured Parts by FE Modeling of Mesostructure. *J. Manuf. Mater. Process.* 2017, *1*, 18, doi:10.3390/jmmp1020018.

45. Miranda, P.; Pajares, A.; Saiz, E.; Tomsia, A.P.; Guiberteau, F. Fracture modes under uniaxial compression in hydroxyapatite scaffolds fabricated by robocasting. 2006, *10*, doi:10.1002/jbm.a.

46. Aydin, L.; Kucuk, S. A method for more accurate FEA results on a medical device developed by 3D technologies. *Polym. Adv. Technol.* 2018, *29*, 2281–2286, doi:10.1002/pat.4339.

47. Lužanin, O.; Movrin, D.; Plan, M. Effect of Layer Thickness , Deposition Angle , and Infill on Maximum Flexural Force in Fdm-Built Specimens. *J. Technol. Plast.* 2014, *39*.

48. Rayegani, F.; Onwubolu, G.C. Fused deposition modelling (fdm) process parameter prediction and optimization using group method for data handling

(gmdh) and differential evolution (de). *Int. J. Adv. Manuf. Technol.* 2014, *73*, 509–519, doi:10.1007/s00170-014-5835-2.

49. Mohamed, O.A.; Masood, S.H.; Bhowmik, J.L.; Nikzad, M.; Azadmanjiri, J. Effect of Process Parameters on Dynamic Mechanical Performance of FDM PC/ABS Printed Parts Through Design of Experiment. *J. Mater. Eng. Perform.* 2016, *25*, 2922–2935, doi:10.1007/s11665-016-2157-6.

50. Pandey, P.M.; Thrimurthulu, K.; Reddy, N.V. Optimal part deposition orientation in FDM by using a multicriteria genetic algorithm. *Int. J. Prod. Res.* 2004, *42*, 4069–4089, doi:10.1080/00207540410001708470.

51. Paz, R.; Monzón, M.D.; Benítez, A.N.; González, B. New lightweight optimisation method applied in parts made by selective laser sintering and Polyjet technologies. *Int. J. Comput. Integr. Manuf.* 2016, *29*, 462–472, doi:10.1080/0951192X.2015.1066033.

52. Anitha, R.; Arunachalam, S.; Radhakrishnan, P. Critical parameters influencing the quality of prototypes in fused deposition modelling. *J. Mater. Process. Technol.* 2001, *118*, 385–388, doi:10.1016/S0924-0136(01)00980-3.

53. Liu, X.; Zhang, M.; Li, S.; Si, L.; Peng, J.; Hu, Y. Mechanical property parametric appraisal of fused deposition modeling parts based on the gray Taguchi method. *Int. J. Adv. Manuf. Technol.* 2017, *89*, 2387–2397, doi:10.1007/s00170-016-9263-3.

54. Sood, A.K.; Ohdar, R.K.; Mahapatra, S.S. Improving dimensional accuracy of Fused Deposition Modelling processed part using grey Taguchi method. *Mater. Des.* 2009, *30*, 4243–4252, doi:10.1016/j.matdes.2009.04.030.

55. Chacón, J.M.; Caminero, M.A.; García-Plaza, E.; Núñez, P.J. Additive manufacturing of PLA structures using fused deposition modelling: Effect of process parameters on mechanical properties and their optimal selection. *Mater. Des.* 2017, *124*, 143–157, doi:10.1016/j.matdes.2017.03.065.

56. Lanzotti, A.; Grasso, M.; Staiano, G.; Martorelli, M. The impact of process parameters on mechanical properties of parts fabricated in PLA with an open-source 3-D printer. *Rapid Prototyp. J.* 2015, *21*, 604–617, doi:10.1108/RPJ-09-2014-0135.

57. Popescu, D.; Zapciu, A.; Amza, C.; Baciu, F.; Marinescu, R. FDM process parameters influence over the mechanical properties of polymer specimens: A

review. *Polym. Test.* **2018**, *69*, 157–166, doi:10.1016/j.polymertesting.2018.05.020.

58. Rao, R.V.; Rai, D.P. Optimization of fused deposition modeling process using teaching-learning-based optimization algorithm. *Eng. Sci. Technol. an Int. J.* **2016**, *19*, 587–603, doi:10.1016/j.jestch.2015.09.008.

59. Perez, R.A.; Mestres, G. Role of pore size and morphology in musculo-skeletal tissue regeneration. *Mater. Sci. Eng. C* **2016**, *61*, 922–939, doi:10.1016/j.msec.2015.12.087.

60. Chuenjitkuntaworn, B.; Inrung, W.; Damrongsri, D.; Mekaapiruk, K.; Supaphol, P.; Pavasant, P. Polycaprolactone / Hydroxyapatite composite scaffolds : Preparation , characterization , and in vitro and in vivo biological responses of human primary bone cells. **2010**, *10*, doi:10.1002/jbm.a.32657.

61. Gómez-Lizárraga, K.K.; Flores-Morales, C.; Prado-Audelo, M.L. Del; Álvarez-Pérez, M.A. Polycaprolactone- and polycaprolactone/ceramic-based 3D-bioplotted porous scaffolds for bone regeneration : A comparative study. **2020**, *79*, 326–335, doi:10.1016/j.msec.2017.05.003.

62. Chen, S.; Guo, Y.; Liu, R.; Wu, S.; Fang, J.; Huang, B.; Li, Z.; Chen, Z.; Chen, Z. Tuning surface properties of bone biomaterials to manipulate osteoblastic cell adhesion and the signaling pathways for the enhancement of early osseointegration. *Colloids Surfaces B Biointerfaces* **2018**, *164*, 58–69, doi:10.1016/j.colsurfb.2018.01.022.

63. Karageorgiou, V.; Kaplan, D. Porosity of 3D biomaterial scaffolds and osteogenesis. **2005**, *26*, 5474–5491, doi:10.1016/j.biomaterials.2005.02.002.

64. Agrawal, C.M.; Mckinney, J.S.; Lanctot, D.; Athanasiou, K.A. Effects of fluid flow on the in vitro degradation kinetics of biodegradable scalolds for tissue engineering. **2000**, *21*, 2443–2452.

65. Bosworth, L.A.; Downes, S. Physicochemical characterisation of degrading polycaprolactone scaffolds. *Polym. Degrad. Stab.* **2010**, *95*, 2269–2276, doi:10.1016/j.polymdegradstab.2010.09.007.

66. Zadpoor, A.A. Bone tissue regeneration: The role of scaffold geometry. *Biomater. Sci.* **2015**, *3*, 231–245, doi:10.1039/c4bm00291a.

67. Guyot, Y.; Papantoniou, I.; Luyten, F.P.; Geris, L. Coupling curvature-dependent and shear stress-stimulated neotissue growth in dynamic bioreactor cultures: a

3D computational model of a complete scaffold. *Biomech. Model. Mechanobiol.* 2016, *15*, 169–180, doi:10.1007/s10237-015-0753-2.

68. Egan, P.F.; Shea, K.A.; Ferguson, S.J. Simulated tissue growth for 3D printed scaffolds. *Biomech. Model. Mechanobiol.* 2018, *17*, 1481–1495, doi:10.1007/s10237-018-1040-9.

69. Paris, M.; Götz, A.; Hettrich, I.; Bidan, C.M.; Dunlop, J.W.C.; Razi, H.; Zizak, I.; Hutmacher, D.W.; Fratzl, P.; Duda, G.N.; et al. Scaffold curvature-mediated novel biomineralization process originates a continuous soft tissue-to-bone interface. *Acta Biomater.* 2017, *60*, 64–80, doi:10.1016/j.actbio.2017.07.029.

70. Zhang, Y.; Wang, P.; Jin, J.; Li, L.; He, S. yuan; Zhou, P.; Jiang, Q.; Wen, C. In silico and in vivo studies of the effect of surface curvature on the osteoconduction of porous scaffolds. *Biotechnol. Bioeng.* 2022, *119*, 591–604, doi:10.1002/bit.27976.

71. Vega Rodríguez, G. del C. AutomateD SwEep CAD modeller Of Extrusion-based G-coDE Available online: https://www.mathworks.com/matlabcentral/fileexchange/100576-automated-sweep-cad-modeller-of-extrusion-based-g-code (accessed on Apr 13, 2023).

72. Vega, G.; Paz, R.; Gleadall, A.; Monzón, M.; Alemán-Domínguez, M.E. Comparison of CAD and voxel-based modelling methodologies for the mechanical simulation of extrusion-based 3D printed scaffolds. *Materials (Basel).* 2021, *14*, doi:10.3390/ma14195670.

73. Geuzaine, C.; Remacle, J.F. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Methods Eng.* 2009, *79*, 1309–1331, doi:10.1002/nme.2579.

74. Wan, C.; Chen, B. Reinforcement and interphase of polymer/graphene oxide nanocomposites. *J. Mater. Chem.* 2012, *22*, 3637–3646, doi:10.1039/c2jm15062j.

75. Iannace, S.; De Luca, N.; Nicolais, L.; Carfagna, C.; Huang, S.J. Physical characterization of incompatible blends of polymethylmethacrylate and polycaprolactone. *J. Appl. Polym. Sci.* 1990, *41*, 2691–2704, doi:10.1002/app.1990.070411116.

76. Northcutt, L.A.; Orski, S. V.; Migler, K.B.; Kotula, A.P. Effect of processing conditions on crystallization kinetics during materials extrusion additive manufacturing. *Polym. (United Kingdom)* 2018, *154*, 182–187,

doi:10.1016/j.polymer.2018.09.018.

77. Paz Hernández, R.; Monzón Verona, M.D.; Benítez Vega, A.N.; González Landín, B. Optimización y experimentación del diseño con algoritmos genéticos y metamodelos en la minimización del peso en piezas obtenidas mediante fabricación aditiva, Universidad de Las Palmas de Gran Canaria, 2014.

78. Lophaven, S.N.; Nielsen, H.B.; Søndergaard, J. *DACE - A Matlab Kriging Toolbox, Version 2.0*; Kgs. Lyngby - Denmark, 2002;

79. Bordón, P.; Paz, R.; Monzón, M.D. Evaluation of the Performance of Atomic Diffusion Additive Manufacturing Electrodes in Electrical Discharge Machining. *Materials (Basel).* 2022, *15*, doi:10.3390/ma15175953.

80. Paz, R.; Pei, E.; Monzón, M.; Ortega, F.; Suárez, L. Lightweight parametric design optimization for 4D printed parts. *Integr. Comput. Aided. Eng.* 2017, *24*, 225–240, doi:10.3233/ICA-170543.

81. Es-Said, O.S.; Foyos, J.; Noorani, R.; Mendelson, M.; Marloth, R.; Pregger, B.A. Effect of layer orientation on mechanical properties of rapid prototyped samples. *Mater. Manuf. Process.* 2000, *15*, 107–122, doi:10.1080/10426910008912976.

82. Nickel, A.H.; Barnett, D.M.; Prinz, F.B. Thermal stresses and deposition patterns in layered manufacturing. *Mater. Sci. Eng. A* 2001, *317*, 59–64, doi:10.1016/S0921-5093(01)01179-0.

83. Akhoundi, B.; Behravesh, A.H. Effect of Filling Pattern on the Tensile and Flexural Mechanical Properties of FDM 3D Printed Products. *Exp. Mech.* 2019, *59*, 883–897, doi:10.1007/s11340-018-00467-y.

84. Yánez, A.; Herrera, A.; Martel, O.; Monopoli, D.; Afonso, H. Compressive behaviour of gyroid lattice structures for human cancellous bone implant applications. *Mater. Sci. Eng. C* 2016, *68*, 445–448, doi:10.1016/j.msec.2016.06.016.

85. Zein, I.; Hutmacher, D.W.; Tan, K.C.; Teoh, S.H. Fused deposition modeling of novel scaffold architectures for tissue engineering applications. *Biomaterials* 2002, *23*, 1169–1185, doi:10.1016/S0142-9612(01)00232-0.

86. Hollister, S.J.; Maddox, R.D.; Taboas, J.M. Optimal design and fabrication of scaffolds to mimic tissue properties. *Biomaterials* 2002, *23*, 4095–4103.

87. Iordache, F. Bioprinted scaffolds. In *Materials for Biomedical Engineering*;

Elsevier Inc., 2019; pp. 35–60 ISBN 9780128169018.

88. Collins, M.N.; Ren, G.; Young, K.; Pina, S.; Reis, R.L.; Oliveira, J.M. Scaffold Fabrication Technologies and Structure/Function Properties in Bone Tissue Engineering. *Adv. Funct. Mater.* 2021, *31*, 1–22, doi:10.1002/adfm.202010609.

89. Monzón, M.; Paz, R.; Wang, L.; Donate, R. The effect of the manufacturing process on the properties of freeze- dried cellulose reinforced alginate-based scaffolds. 2019, *84*.

90. Sun, H.; Al-Marzouqi, H.; Vega, S. EPCI: A new tool for predicting absolute permeability from computed tomography images. *Geophysics* 2019, *84*, F97–F102, doi:10.1190/geo2018-0653.1.

91. Wang, L.; Xu, M.; Zhang, L.; Zhou, Q.; Luo, L. Automated quantitative assessment of three-dimensional bioprinted hydrogel scaffolds using optical coherence tomography. *Biomed. Opt. Express* 2016, *7*, 894, doi:10.1364/boe.7.000894.

92. Moore, M.J.; Jabbari, E.; Ritman, E.L.; Lu, L.; Currier, B.L.; Windebank, A.J.; Yaszemski, M.J. Quantitative analysis of interconnectivity of porous biodegradable scaffolds with micro-computed tomography. *J. Biomed. Mater. Res. - Part A* 2004, *71*, 258–267, doi:10.1002/jbm.a.30138.

Anexos

```
%NOMBRE ARCHIVOS
filename_gcode=input('Introduzca el nombre del archivo G-code (terminado
en .gcode): ','s');
filename_py=input('Introduzca el nombre del archivo Python (terminado
en .py): ','s');
%LECTURA ENCABEZADO
ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
HEAD=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
%s %s %s',[1,20]);
fclose(ARCH); %Cierre del archivo

%LECTURA DE COORDENADAS
for i=1:1:20
  sim=cell2mat(strfind(HEAD{1,i},'Simplify3D(R)')); %Busca en cada
  celda 'Simplify3D'
  sli=cell2mat(strfind(HEAD{1,i},'Slic3r')); %Busca en cada celda
  'Slic3r'
  fc=cell2mat(strfind(HEAD{1,i},'FLAVOR')); %Busca en cada celda el
  encabezado típico de FullControl
  if sim>0
    [coordinates,fil_D]=Coord_Gen_Simplify(filename_gcode);
  elseif sli>0
    [coordinates,fil_D]=Coord_Gen_Slic3r(filename_gcode);
  elseif fc>0
    [coordinates,fil_D]=Coord_Gen_FullControl(filename_gcode);
  end
end

[m,n]=size(coordenadas); %Dimensiones tabla; m = número de coordenadas
COORD_mm=coordenadas*1000; %Coordenadas en mm

%GENERAR CORTES PARA EVITAR AUTO-INTERSECCIONES Y TRAYECTORIAS CERRADAS
COORD_mm(m+1,4)=0; %Añadir una fila al final, cuya cuarta columna es un
"0"
[row,col]=find(COORD_mm(:,4) == 0); %Localizar el inicio de un tramo +
la posición de la última fila de ceros
row_size=size(row,1); %Número de tramos + última fila de ceros

for j=1:1:row_size-1 %Recorre tramo a tramo

  %Determinación de width
  height=COORD_mm(row(j,1),3); %Determinación altura de capa
  %Sumatoria distancias
  dist=0; %Inicialización de variable dist
  for i=row(j,1):1:row(j+1,1)-2 %Recorre el tramo
    p1=COORD_mm(i,1:2);
    p2=COORD_mm(i+1,1:2);
    dist_n=sqrt(sum((p1-p2).^2)); %Calcula distancia entre puntos
    dist=dist+dist_n; %Sumatoria distancias
  end
  %Diferencia E
  AE=(COORD_mm(row(j+1,1)-1,5)-COORD_mm(row(j,1),5))/1000;
  %Ancho de extrusión
  width=height+(AE*pi*0.875^2/dist-pi*height^2/4)/height;

  %INTERSECCIONES
  %Inicialización de variables
  inter_x=0;
  inter_y=0;
```

```
if row(j+1,1)-row(j,1)>2 %Si hay más de dos puntos en el tramo

  %Se analizan las coordenadas de la matriz entre el inicio de un
  tramo y el fin de este (el inicio del siguiente menos uno)
  datax=COORD_mm(row(j,1):row(j+1,1)-1,1);
  datay=COORD_mm(row(j,1):row(j+1,1)-1,2);

  %Busca los máximos y mínimos de las coordenadas del tramo (valor y
  localización)tanto en X como en Y
  [max_x,locs_max_x] = findpeaks(datax);
  [max_y,locs_max_y] = findpeaks(datay);
  [min_x,locs_min_x] = findpeaks(-datax);
  [min_y,locs_min_y] = findpeaks(-datay);

  %Se determina el número de máximos y mínimos
  size_max_x=size(max_x);
  size_min_x=size(min_x);
  size_max_y=size(max_y);
  size_min_y=size(min_y);

  %Comparación de cada punto máximo con los puntos mínimos en X
  for i=1:1:size_max_x
    p1=COORD_mm(row(j,1)+locs_max_x(i)-1,1:2);

    %Coordenadas del máximo
    for w=1:1:size_min_x
      p2=COORD_mm(row(j,1)+locs_min_x(w)-1,1:2);  %Coordenadas del
      mínimo
      dist=sqrt(sum((p1-p2).^2)); %Distancia entre el máximo y el
      mínimo

      %Si dicha distancia es inferior al ancho de extrusión y hay
      más de 3 puntos entre ellos, existe una auto-intersección
      if (dist<width)&&(abs(locs_max_x(i)-locs_min_x(w))>3)
        inter_x=1;

      %Si dicha distancia es superior al ancho de extrusión, pero
      inferior al 150% de este ancho y hay más de 3 puntos entre
      ellos; se comparan las distancias entre el punto mínimo y los
      dos anteriores y dos posteriores con el punto máximo y los dos
      anteriores y dos posteriores
      elseif (dist<1.5*width)&&(abs(locs_max_x(i)-locs_min_x(w))>3)
        Pmin=[COORD_mm(row(j,1)+locs_min_x(w)-3,1:2);
        COORD_mm(row(j,1)+locs_min_x(w)-2,1:2);
        COORD_mm(row(j,1)+locs_min_x(w)-1,1:2);
        COORD_mm(row(j,1)+locs_min_x(w),1:2);
        COORD_mm(row(j,1)+locs_min_x(w)+1,1:2)];

        Pmax=[COORD_mm(row(j,1)+locs_max_x(i)-3,1:2);
        COORD_mm(row(j,1)+locs_max_x(i)-2,1:2);
        COORD_mm(row(j,1)+locs_max_x(i)-1,1:2);
        COORD_mm(row(j,1)+locs_max_x(i),1:2);
        COORD_mm(row(j,1)+locs_max_x(i)+1,1:2)];

        for k=1:1:5
          for w=1:1:5
            dist=sqrt(sum((Pmax(w,1:2)-Pmin(k,1:2)).^2));
            if (dist<width)
              inter_x=1;
            end
          end
```

```
        end
      end
    end
end

%Comparación de cada punto máximo con los puntos mínimos en Y (se
repiten los pasos anteriores)
for i=1:1:size_max_y
 p1=COORD_mm(row(j,1)+locs_max_y(i)-1,1:2);
  for w=1:1:size_min_y
    p2=COORD_mm(row(j,1)+locs_min_y(w)-1,1:2);
    dist=sqrt(sum((p1-p2).^2));

    if (dist<width)&&(abs(locs_max_y(i)-locs_min_y(w))>3)
      inter_y=1;
    elseif (dist<1.5*width)&&(abs(locs_max_y(i)-locs_min_y(w))>3)
      Pmin=[COORD_mm(row(j,1)+locs_min_y(w)-3,1:2);
      COORD_mm(row(j,1)+locs_min_y(w)-2,1:2);
      COORD_mm(row(j,1)+locs_min_y(w)-1,1:2);
      COORD_mm(row(j,1)+locs_min_y(w),1:2);
      COORD_mm(row(j,1)+locs_min_y(w)+1,1:2)];

      Pmax=[COORD_mm(row(j,1)+locs_max_y(i)-3,1:2);
      COORD_mm(row(j,1)+locs_max_y(i)-2,1:2);
      COORD_mm(row(j,1)+locs_max_y(i)-1,1:2);
      COORD_mm(row(j,1)+locs_max_y(i),1:2);
      COORD_mm(row(j,1)+locs_max_y(i)+1,1:2)];

      for k=1:1:5
        for w=1:1:5
          dist=sqrt(sum((Pmax(w,1:2)-Pmin(k,1:2)).^2));
          if (dist<width)
            inter_y=1;
          end
        end
      end
    end
  end
end

%Si hay una intersección en Y, se genera el corte en los máximos y
mínimos de X (dándole valor -1 a la cuarta columna en la fila de
esa coordenada); y viceversa, si hay interacción en X, el corte se
establece en los máximos y mínimos de Y
if inter_y==1
  if size_max_x(1,1)>0
    for i=1:1:size_max_x(1,1)
      COORD_mm(row(j,1)+locs_max_x(i)-1,4)=-1;
    end
  end
  if size_min_x(1,1)>0
    for i=1:1:size_min_x(1,1)
      COORD_mm(row(j,1)+locs_min_x(i)-1,4)=-1;
    end
  end
elseif inter_x==1
  if size_max_y(1,1)>0
    for i=1:1:size_max_y(1,1)
      COORD_mm(row(j,1)+locs_max_y(i)-1,4)=-1;
    end
  end
```

```
        if size_min_y(1,1)>0
          for i=1:1:size_min_y(1,1)
            COORD_mm(row(j,1)+locs_min_y(i)-1,4)=-1;
          end
        end
      end
    end

    %TRAYECTORIAS CERRADAS
    %Si la distancia entre el primer y el último punto del plano es menor
    a la mitad del ancho de extrusión, se considera que es una trayectoria
    cerrada. Cuando ocurre esto, se separa la última línea recta del
    tramo, asignando a la cuarta columna de la penúltima coordenada de la
    trayectoria el valor -1
    dist_ext=COORD_mm(row(j),1:2)-COORD_mm(row(j+1)-1,1:2);

    if sqrt(sum(dist_ext.^2))<(width/2)
      COORD_mm(row(j+1)-2,4)=-1;
    end
end

row,col]=find(COORD_mm(:,4) <= 0);
row=[row;m+1];
row_size=size(row,1);

%CREACIÓN ARCHIVO PHYTON
FILE=fopen(filename_py,'w');
head='from part import *\nfrom material import *\nfrom section import
*\nfrom assembly import *\nfrom step import *\nfrom interaction import
*\nfrom load import *\nfrom mesh import *\nfrom optimization import
*\nfrom job import *\nfrom sketch import *\nfrom visualization import
*\nfrom connectorBehavior import *\n';

sheetSize=20;
Model_name='''Model-1''';
Sketch_type='''__sweep__''';
Part_name='''Part-1''';
gridSpacing=0.74;
Profile='''__profile__''';

%PRIMER BARRIDO
%Croquis
%Escribe la cabecera
fprintf(FILE,head);
fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
sheetSize=%3.3f)\n',Model_name,Sketch_type,sheetSize);

j=1; %Inicialización variable

%Define la fila de la matriz coordenadas en la que empieza la nueva
trayectoria y en la fila en la que acaba, que dependerá de si son
trayectorias que no auto-intersectan y son abiertas, donde el final
sería en la fila anterior al comienzo de la siguiente (cuyo valor en la
cuarta columna es un "0") o si es una de las trayectorias cortadas por
auto-intersección o ser cerrada, en la cual llegaría hasta el punto
donde comienza la siguiente (su valor en la cuarta columna es "-1".
START=row(j,1);
if COORD_mm(row(j+1,1),4)==0
  END=row(j+1,1)-1;
else
  END=row(j+1,1);
```

```
end

%Recorre toda la trayectoria, escribiendo en el archivo Python la línea
de programación correspondiente a la creación de una nueva línea del
croquis (definida por dos puntos).
for i=START:1:END-1
   point=COORD_mm(i:i+1,1:2);
   point1=point(1,1:2);
   point2=point(2,1:2);
   fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,%3.3f),
   point2=(%3.3f, %3.3f))\n',Model_name,Sketch_type,point1,point2);
end

%Rotación de ejes para la correcta orientación del croquis con respecto
al plano y escritura de dichas líneas de programación en el Python.
var=COORD_mm(2,1:2)-COORD_mm(1,1:2);
tx=-90;
ty=0;
tz=atand(var(1)/var(2));
rot_x=[1 0 0;0 cosd(tx) -sind(tx);0 sind(tx) cosd(tx)];
rot_y=[cosd(ty) 0 sind(ty);0 1 0;-sind(ty) 0 cosd(ty)];
rot_z=[cosd(tz) -sind(tz) 0;sind(tz) cosd(tz) 0;0 0 1];
I=[1 0 0;0 1 0;0 0 1];
ROT=I*rot_x*rot_y*rot_z;

fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
%1.6f, %1.6f, %1.6f, %3.3f, %3.3f,%3.3f))\n',Model_name,Profile,
sheetSize,ROT(1,1:3),ROT(2,1:3),ROT(3,1:3),COORD_mm(1,1:2),
COORD_mm(1,3)-COORD_mm(1,3));

fprintf(FILE,'mdb.models[%s].sketches[%s].ConstructionLine(point1=
(%3.3f,0.0), point2=(%3.3f, 0.0))\n
mdb.models[%s].sketches[%s].ConstructionLine(point1=(0.0,%3.3f),
point2=(0.0, %3.3f))\n',Model_name,Profile,-sheetSize,sheetSize,
Model_name,Profile,-sheetSize,sheetSize);

%Perfil
height=COORD_mm(1,3); %Determinación alto de capa
dist=0; %Inicialización variable
%Determinación de la distancia total recorrida en la trayectoria
for i=START:1:END-1
   p1=COORD_mm(i,1:2);
   p2=COORD_mm(i+1,1:2);
   dist_n=sqrt(sum((p1-p2).^2)); %Cálculo de distancia
   dist=dist+dist_n; %Sumatoria de distancia
end
%Diferencia E
AE=(COORD_mm(END,5)-COORD_mm(START,5))/1000;
%Ancho de extrusión
width=height+(AE*pi*0.875^2/dist-pi*height^2/4)/height;

%Definición de perfil en PY
fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f, %3.3f),
point2=(%3.3f, %3.3f))\n
mdb.models[%s].sketches[%s].Line(point1=(%3.3f, %3.3f), point2=(%3.3f,
%3.3f))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f,%3.3f), vector=(1.0, 0.0))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f), vector=(-1.0, 0.0))\n',Model_name,
```

```
Profile,-(width-height)/2,height,(width-height)/2,height,Model_name,
Profile,-(width-height)/2,0,(width-height)/2,0,Model_name,Profile,
(width-height)/2,0,(width-height)/2,height,Model_name,Profile,      -
(width-height)/2,height,-(width-height)/2,0);

%Definición de barrido en PY
fprintf(FILE,'mdb.models[%s].Part(dimensionality=THREE_D,name=%s,
type=DEFORMABLE_BODY)\nmdb.models[%s].parts[%s].BaseSolidSweep(path=md
b.models[%s].sketches[%s], sketch=mdb.models[%s].sketches[%s])\n',
Model_name,Part_name,Model_name,Part_name,Model_name,Sketch_type,
Model_name,Profile);

fprintf(FILE,'\n');

%Definición de ejes en PY
fprintf(FILE,'mdb.models[%s].parts[%s].DatumAxisByPrincipalAxis(princi
palAxis=YAXIS)\nmdb.models[%s].parts[%s].DatumAxisByPrincipalAxis(prin
cipalAxis=ZAXIS)\n',Model_name,Part_name,Model_name,Part_name);

%Definición de Plano 0 en PY
fprintf(FILE,'mdb.models[%s].parts[%s].DatumPlaneByPrincipalPlane(
offset=%3.3f,principalPlane=XYPLANE)\n',Model_name,Part_name,0);

%Definición de resto de planos en PY
fprintf(FILE,'mdb.models[%s].parts[%s].DatumPlaneByPrincipalPlane(
offset=%3.3f,principalPlane=XYPLANE)\n',Model_name,Part_name,
COORD_mm(row(1),3)); %Para r=1

for r=2:1:row_size-2
  if COORD_mm(row(r),3)==COORD_mm(row(r-1),3)
  else %Si la coordenada Z cambia
    fprintf(FILE,'mdb.models[%s].parts[%s].DatumPlaneByPrincipalPlane(o
    ffset=%3.3f,principalPlane=XYPLANE)\n',Model_name,Part_name,
    COORD_mm(row(r),3));
  end
end

fprintf(FILE,'\n');

%SIGUIENTES BARRIDOS
datum=4; %Inicialización variable
h_layer=COORD_mm(row(1),3); %Inicialización altura de capa
h_newlayer=h_layer; %Inicialización nueva altura de capa

%Recorre el resto de la matriz, excluyendo el primer barrido
for j=2:1:row_size-2
  %Definición de inicio y fin de la trayectoria, teniendo en cuenta si
  el valor de la cuarta columna es "0" o "-1".
  START=row(j,1);
  if COORD_mm(row(j+1,1),4)==0
    END=row(j+1,1)-1;
  else
    END=row(j+1,1);
  end

  %Si se indica que el principio y el fin es el mismo punto, es que ha
  terminado. Si no, se continúa generando barridos.
  if START==END
    fprintf(FILE,'\n');
  else
    h_newlayer=COORD_mm(START,3); %Toma el valor de altura de capa
```

```
%Si la altura de capa es igual a la del primer barrido quiere decir
que se sitúa en la 1ª capa, en el Plano 0 y, por tanto, Datum 4
if h_newlayer==COORD_mm(row(1),3)
   fprintf(FILE,'mdb.models[%s].ConstrainedSketch(gridSpacing=
   %3.3f,name=%s,sheetSize=%3.3f,transform=mdb.models[%s].
   parts[%s].MakeSketchTransform(sketchPlane=mdb.models[%s].
   parts[%s].datums[%d],sketchPlaneSide=SIDE1,sketchUpEdge=
   mdb.models[%s]. parts[%s].datums[%d],sketchOrientation=LEFT,
   origin=(0.0, 0.0,%3.3f)))\n',Model_name,gridSpacing,
   Sketch_type,sheetSize,Model_name,Part_name,Model_name,
   Part_name,4,Model_name, Part_name,2,0);

%Si no está en la primera capa, pero se mantiene en la misma capa
que el barrido anterior
elseif h_layer==h_newlayer
   cont=0;
   for w=j:-1:1
     if cont<1
       if h_newlayer==COORD_mm(row(w),3)
       else
          fprintf(FILE,'mdb.models[%s].ConstrainedSketch(
          gridSpacing=%3.3f, name=%s,sheetSize=%3.3f,
          transform=mdb.models[%s].parts[%s].
          MakeSketchTransform(sketchPlane=mdb.models[%s].
          parts[%s].datums[%d],sketchPlaneSide=SIDE1,
          sketchUpEdge=mdb.models[%s].parts[%s].datums[%d],
          sketchOrientation= LEFT, origin=(0.0, 0.0,%3.3f)))\n',
          Model_name,gridSpacing,Sketch_type,sheetSize,
          Model_name,Part_name,Model_name,Part_name,datum,
          Model_name,Part_name,2, COORD_mm(row(w),3));

          cont=1;
       end
     end
   end

%Si existe un cambio de capa
else
   datum=datum+1;
   fprintf(FILE,'mdb.models[%s].ConstrainedSketch(gridSpacing=%3.3
   f, name=%s,sheetSize=%3.3f,transform=mdb.models[%s].
   parts[%s].MakeSketchTransform(sketchPlane=mdb.models[%s].
   parts[%s].datums[%d],sketchPlaneSide=SIDE1,sketchUpEdge=
   mdb.models[%s]. parts[%s].datums[%d],sketchOrientation=LEFT,
   origin=(0.0, 0.0,%3.3f)))\n',Model_name,gridSpacing,
   Sketch_type,sheetSize,Model_name,Part_name,Model_name,
   Part_name,datum,Model_name, Part_name,2,h_layer);

end

%Línea de programación
fprintf(FILE,'mdb.models[%s].parts[%s].
projectReferencesOntoSketch(filter=COPLANAR_EDGES,sketch=
mdb.models[%s].sketches[%s])\n',Model_name,Part_name,Model_name,Sk
etch_type);

%Definición de croquis
for i=START:1:END-1
   point=COORD_mm(i:i+1,1:2);
   point1=point(1,1:2);
```

```
   point2=point(2,1:2);
   fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,
   %3.3f), point2=(%3.3f, %3.3f))\n',Model_name,Sketch_type,
   point1,point2);
end

%Rotación de ejes
var=COORD_mm(row(j)+1,1:2)-COORD_mm(row(j),1:2);
tx=-90;
ty=0;
tz=atand(var(1)/var(2));
rot_x=[1 0 0;0 cosd(tx) -sind(tx);0 sind(tx) cosd(tx)];
rot_y=[cosd(ty) 0 sind(ty);0 1 0;-sind(ty) 0 cosd(ty)];
rot_z=[cosd(tz) -sind(tz) 0;sind(tz) cosd(tz) 0;0 0 1];
I=[1 0 0;0 1 0;0 0 1];
ROT=I*rot_x*rot_y*rot_z;

if h_newlayer==COORD_mm(row(1),3) %Capa 1
   fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
   sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
   %1.6f, %1.6f, %1.6f, %1.6f, %3.3f,%3.3f,%3.3f))\n',
   Model_name,Profile,sheetSize,ROT(1,1:3),ROT(2,1:3),
   ROT(3,1:3),COORD_mm(START,1:2),0);

elseif h_layer==h_newlayer %Misma capa
   cont=0;
   for w=j:-1:1
      if cont<1
         if h_newlayer==COORD_mm(row(w),3)
         else
            fprintf(FILE,'mdb.models[%s].ConstrainedSketch( name=%s,
            sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f,
            %1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
            %3.3f,%3.3f,%3.3f))\n',Model_name,Profile,sheetSize,
            ROT(1,1:3),ROT(2,1:3),ROT(3,1:3),COORD_mm(START,1:2),
            COORD_mm(row(w),3));
            cont=1;
         end
      end
   end

else %Cambio de capa
   fprintf(FILE,'mdb.models[%s].ConstrainedSketch(name=%s,
   sheetSize=%3.3f,transform=(%1.6f, %1.6f, %1.6f, %1.6f, %1.6f,
   %1.6f, %1.6f, %1.6f, %1.6f, %3.3f,%3.3f,%3.3f))\n',
   Model_name,Profile,sheetSize,ROT(1,1:3),ROT(2,1:3),
   ROT(3,1:3),COORD_mm(START,1:2),h_layer);
end

fprintf(FILE,'mdb.models[%s].sketches[%s].ConstructionLine(
point1=(%3.3f,0.0),point2=(%3.3f,0.0))\nmdb.models[%s].
sketches[%s]. ConstructionLine(point1=(0.0,%3.3f), point2=(0.0,
%3.3f))\nmdb.models[%s].parts[%s].projectReferencesOntoSketch(
filter=COPLANAR_EDGES,sketch=mdb.models[%s].sketches[%s])\n',
Model_name,Profile,-sheetSize,sheetSize,Model_name,Profile,     -
sheetSize,sheetSize,Model_name,Part_name,Model_name,Profile);

%Perfil
%Determinación altura de capa
if h_newlayer==COORD_mm(1,3)
   height=h_newlayer;
```

```
elseif h_newlayer==h_layer
   height=height;
else
   height=h_newlayer-h_layer;
end

%Determinación de ancho de extrusión
dist=0;
for i=START:1:END-1
   p1=COORD_mm(i,1:2);
   p2=COORD_mm(i+1,1:2);
   dist_n=sqrt(sum((p1-p2).^2)); %Cálculo distancia
   dist=dist+dist_n; %Sumatoria distancias
end
%Diferencia E
AE=(COORD_mm(END,5)-COORD_mm(START,5))/1000;
%Ancho de extrusión
width=height+(AE*pi*0.875^2/dist-pi*height^2/4)/height;

%Línea de programación para definir perfil
fprintf(FILE,'mdb.models[%s].sketches[%s].Line(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f))\nmdb.models[%s].sketches[%s].
Line(point1=(%3.3f, %3.3f), point2=(%3.3f, %3.3f))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f,%3.3f), vector=(1.0, 0.0))\n
mdb.models[%s].sketches[%s].ArcByStartEndTangent(point1=(%3.3f,
%3.3f), point2=(%3.3f, %3.3f), vector=(-1.0, 0.0))\n',Model_name,
Profile,-(width-height)/2,height,(width-height)/2,height,
Model_name,Profile,-(width-height)/2,0,(width-height)/2,0,
Model_name,Profile,(width-height)/2,0,(width-height)/2,height,
Model_name,Profile,-(width-height)/2,height,-(width-height)/2,0);

%Definición barrido
x2=COORD_mm(START+1,1);
x1=COORD_mm(START,1);
y2=COORD_mm(START+1,2);
y1=COORD_mm(START,2);
x=((-y1)/(y2-y1))*(x2-x1)+x1;

if ((x2>x1)&&(y2==y1))||((x2==x1)&&(y2<y1))||
((x2>x1)&&(y2<y1))||((x>0)&&(x2<x1)&&(y2<y1))||((x<0)&&(x2>x1)&&(y
2>y1))

   fprintf(FILE,'mdb.models[%s].parts[%s].SolidSweep(path=
   mdb.models[%s].sketches[%s],pathOrientation=LEFT,pathPlane=
   mdb.models[%s].parts[%s].datums[%d],pathUpEdge=
   mdb.models[%s].parts[%s].datums[%d],profile=mdb.models[%s].
   sketches[%s],sketchOrientation=RIGHT,sketchUpEdge=
   mdb.models[%s].parts[%s].datums[%d])\n',Model_name,Part_name,Mo
   del_name,Sketch_type,Model_name,Part_name,datum,Model_name,Part
   _name,2,Model_name, Profile,Model_name,Part_name,3);

else
   fprintf(FILE,'mdb.models[%s].parts[%s].SolidSweep(path=
   mdb.models[%s].sketches[%s],pathOrientation=LEFT,pathPlane=
   mdb.models[%s].parts[%s].datums[%d],pathUpEdge= mdb.models[%s].
   parts[%s].datums[%d],profile=mdb.models[%s].
   sketches[%s],sketchOrientation=LEFT,sketchUpEdge=
   mdb.models[%s].parts[%s].datums[%d])\n',Model_name,Part_name,Mo
   del_name,Sketch_type,Model_name,Part_name,datum,Model_name,Part
   _name,2,Model_name, Profile,Model_name,Part_name,3);
```

```
    end
  end

  h_layer=h_newlayer; %La altura de capa toma el valor de la última capa
  en la que se ha generado el barrido
  fprintf(FILE,'\n');
end

fclose(FILE);
```

ANEXO B. FUNCIÓN COORD_GEN_SLIC3R

```
function [coordinates,fil_D]=Coord_Gen_Slic3r(filename_gcode)
  %INTRODUCCIÓN ARCHIVO
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  CI=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[inf,20]); %Lectura de todas las filas y 20 columnas
  kc=size(CI{2},1); %Determinación de la dimensión máxima de filas de la matriz (kc)
  fclose(ARCH); %Cierre de archivo
  ARCH=fopen(filename_gcode,'r'); %Nueva apertura
  C=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[kc,20]); %Lectura de 'kc' filas y 20 columnas
  c=[C{1} C{2} C{3} C{4} C{5}]; %Creación de matriz de tipo string con todo el archivo
  fclose(ARCH); %Cierre del archivo
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  COM=textscan(ARCH,'%s %s','Delimiter',';');
  comments=[COM{2}];
  fclose(ARCH); %Cierre de archive

  %Diámetro del filamento
  [fil,col]=size(comments);
  for i=1:1:fil
    dia=strfind(comments{i},'filament_diameter');
    if isempty(dia)==0
      fil_dia=strrep(comments{i},'filament_diameter = ','');
      fil_D=str2num(fil_dia);
    end
  end

  %ELIMINACIÓN DE COMENTARIOS
  k=size(c,1); %Número de columnas de la matriz 'c'
  for i=1:1:k
    for j=1:1:5
      f=strfind(c{i,j},';'); %Busca en cada celda un ';'
      ff=strfind(c{i,j},'='); %Busca en cada celda un '='
      if ff>0
        c{i,j}=' '; %Si lo encuentra borra esa celda
      elseif f>0
        for m=j:1:5
          c{i,m}=' '; %Si lo encuentra borra esa celda y todas las siguientes de la fila
        end
      end
    end
  end

  %ELIMINACIÓN DE COMANDOS 'M', 'S', 'G28', 'G' y 'F'
  for i=1:1:k
    for j=1:1:5
      M=strfind(c{i,j},'M'); %Busca el comando 'M' celda por celda
      H=strfind(c{i,j},'G00'); %Busca el comando 'G00' celda por celda
      V=strfind(c{i,j},'G28'); %Busca el comando 'G28' celda por celda
      G=strfind(c{i,j},'G'); %Busca el comando 'G' celda por celda
      S=strfind(c{i,j},'S'); %Busca el comando 'S' celda por celda
      F=strfind(c{i,j},'F'); %Busca el comando 'F' celda por celda
      if M==1
        c{i,j}=''; %Borra la celda donde se encuentra el comando 'M'
      elseif H==1
        c{i,j}=''; %Borra G0
```

```
    elseif V==1 %Borra G28 y todo lo que sigue al comando
      c{i,j}='';
      c{i,j+1}='';
      c{i,j+2}='';
    elseif G==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'G'
    elseif S==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'S'
    elseif F==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'F'
    end
  end
end

%VECTOR POSICIÓN + E
pos=zeros(1,5); %Inicialización matriz 'pos'
for i=1:1:k
  for j=1:1:5
    X=strfind(c{i,j},'X'); %Busca 'X' celda por celda
    Y=strfind(c{i,j},'Y'); %Busca 'Y' celda por celda
    Z=strfind(c{i,j},'Z'); %Busca 'Z' celda por celda
    E=strfind(c{i,j},'E'); %Busca el comando 'E' celda por celda
    if X==1
      cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
      únicamente los números
      pos(i,1)=str2num(cp{i,1}); %Pasa los números al vector
      posición (pos)
    elseif Y==1
      cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
      únicamente los números
      pos(i,2)=str2num(cp{i,2}); %Pasa los números al vector
      posición (pos)
    elseif Z==1
      cp{i,3}=strrep(c{i,j},'Z',''); %Borra la 'Z' para dejar
      únicamente los números
      pos(i,3)=str2num(cp{i,3}); %Pasa los números al vector
      posición (pos)
    elseif E==1
      cp{i,4}=strrep(c{i,j},'E',''); %Borra la 'E' para dejar
      únicamente los números
      pos(i,5)=str2num(cp{i,4}); %Pasa los números al vector
      posición (pos)
    end
  end
end

%ASIGNACIÓN VALORES CUARTA COLUMNA
[k,y]=size(pos);
for i=1:1:k
  inwards=strfind(comments(i),'move inwards before travel');
  skirt=strfind(comments(i),'skirt');
  if isequal(skirt,{[1]})
    pos(i,4)=2;
  elseif isequal(skirt,{[15]})
    pos(i,4)=2;
  elseif isequal(inwards,{[1]})
    pos(i,4)=2;
    pos(i+1,3)=pos(i,3);
  elseif isequal(comments(i),{'lift nozzle'})
    pos(i,4)=2;
  elseif isequal(comments(i),{'move to first infill point'})
```

```
      pos(i,4)=0;
    elseif isequal(comments(i),{'move to first infill (bridge) point'})
      pos(i,4)=0;
    elseif isequal(comments(i),{'move to first perimeter point'})
      pos(i,4)=0;
    else
      pos(i,4)=1;
    end
end

%BÚSQUEDA DE Z EN CAMBIO DE CAPA Y AJUSTAR EL VALOR "E"
for i=1:1:k
    layer=strfind(comments(i),'move to next layer');
    if isequal(layer,{[1]})
      pos(i,4)=2;
      fir=find(pos(i:k,4)==0);
      pos(i+fir(1)-1,3)=pos(i,3);
    elseif isequal(comments(i),{'reset extrusion distance'})
      pos(i+1,5)=2;
    end
end

%LIMPIEZA
fila=1;
pos_3=pos(:,1:3);
com=sum(pos_3,2);
[k,y]=size(com);
posf=zeros(1,5);
for i=1:1:k
    if com(i,1)>0 && pos(i,4)<2
      posf(fila,1:5)=pos(i,1:5);
      fila=fila+1;
    end
end

%Si en la fila ha encontrado una X, una Y o/y una Z, con lo cual
%existe una posición, pero alguna de las coordenadas no aparece, toma
%el valor anterior (no es cero)
[filf,colf]=size(posf);
for i=2:1:filf
    if posf(i,1)==0 %Si no se especifica el valor de X
      posf(i,1)=posf(i-1,1); %Toma el valor X de la posición anterior
    end
    if posf(i,2)==0 %Si no se especifica el valor de Y
      posf(i,2)=posf(i-1,2); %Toma el valor Y de la posición anterior
    end
    if posf(i,3)==0 %Si no se especifica el valor de Z
      posf(i,3)=posf(i-1,3); %Toma el valor Z de la posición anterior
    end
    if posf(i,5)==0 %Si no se especifica el valor de E
      posf(i,5)=posf(i-1,5); %Toma el valor E de la posición anterior
    end
end

%PREPARACIÓN MATRIZ COORDENADAS
coordenadas=posf(1:filf,1:5);
[filc,colc]=size(coordenadas);
minx=coordenadas(1,1);
miny=coordenadas(1,2);
for i=1:1:filc
    if coordenadas(i,1)<minx
```

```
        minx=coordenadas(i,1);
      end
      if coordenadas(i,2)<miny
        miny=coordenadas(i,2);
      end
    end
    minx=minx-2;
    miny=miny-2;
    for i=1:1:filc
      coordenadas(i,1)=(coordenadas(i,1)-minx)/1000;
      coordenadas(i,2)=(coordenadas(i,2)-miny)/1000;
      coordenadas(i,3)=coordenadas(i,3)/1000;
    end

end %Final de función
```

ANEXO C. FUNCIÓN COORD_GEN_SIMPLIFY

```
function [coordinates,fil_D]=Coord_Gen_Simplify(filename_gcode)
  %INTRODUCCIÓN ARCHIVO
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  CI=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[inf,20]); %Lectura de todas las filas y 20 columnas
  kc=size(CI{2},1); %Determinación de las dimensión máxima de filas de la matriz (kc)
  fclose(ARCH); %Cierre de archivo
  ARCH=fopen(filename_gcode,'r'); %Nueva apertura
  C=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',[kc,20]); %Lectura de 'kc' filas y 20 columnas
  c=[C{1} C{2} C{3} C{4} C{5}]; %Creación de matriz de tipo cadena con todo el archivo
  fclose(ARCH); %Cierre del archivo

  %Diámetro de filamento
  [fil,col]=size(c);
  for i=1:1:fil
    dia=strfind(c{i,2},'filamentDiameters');
    if isempty(dia)==0
      fil_1=strfind(c{i,2},'1.75');
      fil_3=strfind(c{i,2},'3');
      if isempty(fil_1)==0
        fil_D=1.75;
      elseif isempty(fil_3)==0
        fil_D=3;
      end
    end
  end

  %MODIFICIÓN DE COMENTARIOS
  k=size(c,1); %Número de columnas de la matriz 'c'
  cont=0;
  for i=1:1:k
    feature=strfind(c(i,2),'feature'); %Busca comentarios 'feature'
    if cont==0&&isequal(feature,{[1]})
      inicio=i;
      cont=cont+1;
    end
  end
  for i=inicio:1:k
    for j=1:1:5
      perimeter=strfind(c{i,j},'perimeter');    %Busca    comentarios 'perimeter'
      infill=strfind(c{i,j},'infill'); %Busca comentarios 'infill'
      solid=strfind(c{i,j},'solid'); %Busca comentarios 'solid layer'
      fill=strfind(c{i,j},'fill'); %Busca comentarios 'fill'
      support=strfind(c{i,j},'support'); %Busca comentarios 'support'
      if perimeter>0
        for m=1:1:5
          c{i,m}=' '; %Si lo encuentra borra esa celda y todas las siguientes de la fila
        end
        c{i,2}='part';
      elseif infill>0
        for m=1:1:5
          c{i,m}=' '; %Si lo encuentra borra esa celda y todas las siguientes de la fila
        end
```

```
      c{i,2}='part';
    elseif solid>0
      for m=1:1:5
        c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
        siguientes de la fila
      end
      c{i,2}='part';
    elseif fill>0
      for m=1:1:5
        c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
        siguientes de la fila
      end
      c{i,2}='part';
    elseif support>0
      for m=1:1:5
        c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
        siguientes de la fila
      end
      c{i,2}='support';
    end
  end
end

%ELIMINACIÓN DE COMENTARIOS
for i=1:1:k
  for j=1:1:5
    comment=strfind(c{i,j},';'); %Busca en cada celda un ';'
    if comment>0
      for m=j:1:5
        c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
        siguientes de la fila
      end
    end
  end
end

%ELIMINACIÓN DE COMANDOS 'M', 'S', 'G28', 'G' y 'F'
for i=1:1:k
  for j=1:1:5
    M=strfind(c{i,j},'M'); %Busca el comando 'M' celda por celda
    H=strfind(c{i,j},'G00'); %Busca el comando 'G00' celda por celda
    V=strfind(c{i,j},'G28'); %Busca el comando 'G28' celda por celda
    G=strfind(c{i,j},'G'); %Busca el comando 'G' celda por celda
    S=strfind(c{i,j},'S'); %Busca el comando 'S' celda por celda
    F=strfind(c{i,j},'F'); %Busca el comando 'S' celda por celda
    if M==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'M'
    elseif H==1
      c{i,j}=''; %Borra G0
    elseif V==1 %Borra G28 y todo lo que sigue al comando
      c{i,j}='';
      c{i,j+1}='';
      c{i,j+2}='';
    elseif G==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'G'
    elseif S==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'S'
    elseif F==1
      c{i,j}=''; %Borra la celda donde se encuentra el comando 'F'
    end
  end
```

```
end

%VECTOR POSICIÓN
pos=NaN(k,5); %Inicialización matriz pos
for i=1:1:k
   %Busca el comando 'E' en la fila
   E3=strfind(c{i,3},'E');
   E4=strfind(c{i,4},'E');
   for j=1:1:5
      X=strfind(c{i,j},'X'); %Busca 'X' celda por celda
      Y=strfind(c{i,j},'Y'); %Busca 'Y' celda por celda
      Z=strfind(c{i,j},'Z'); %Busca 'Z' celda por celda
      E=strfind(c{i,j},'E'); %Busca 'E' celda por celda
      if E3==1
         if X==1
            cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
            únicamente los números
            pos(i,1)=str2num(cp{i,1}); %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
         elseif Y==1
            cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
            únicamente los números
            pos(i,2)=str2num(cp{i,2}); %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
         elseif E==1
            cp{i,3}=strrep(c{i,j},'E',''); %Borra la 'E' para dejar
            únicamente los números
            pos(i,5)=str2num(cp{i,3}); %Pasa los números al vector
            posición (pos)
         end
      elseif E4==1
         if X==1
            cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
            únicamente los números
            pos(i,1)=str2num(cp{i,1}); %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
         elseif Y==1
            cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
            únicamente los números
            pos(i,2)=str2num(cp{i,2}); %Pasa los números al vector
            posición (pos)
            pos(i,4)=1;
         elseif E==1
            cp{i,4}=strrep(c{i,j},'E',''); %Borra la 'E' para dejar
            únicamente los números
            pos(i,5)=str2num(cp{i,4}); %Pasa los números al vector
            posición (pos)
         end
      else
         if X==1
            cp{i,1}=strrep(c{i,j},'X',''); %Borra la 'X' para dejar
            únicamente los números
            pos(i,1)=str2num(cp{i,1}); %Pasa los números al vector
            posición (pos)
            pos(i,4)=0;
            pos(i,5)=0;
         elseif Y==1
            cp{i,2}=strrep(c{i,j},'Y',''); %Borra la 'Y' para dejar
```

```
            únicamente los números
            pos(i,2)=str2num(cp{i,2});  %Pasa  los  números  al  vector
            posición (pos)
            pos(i,4)=0;
            pos(i,5)=0;
          end
        end
        if Z==1
          cp{i,3}=strrep(c{i,j},'Z','');  %Borra  la  'Z'  para  dejar
          únicamente los números
          pos(i,3)=str2num(cp{i,3});  %Pasa  los  números  al  vector
          posición (pos)
        end
    end
end

%COMPLETAR CUARTA FILA
cont=0;
fil_support=1;
fil_part=1;
for i=1:1:k
    part=strfind(c{i,2},'part');
    support=strfind(c{i,2},'support');
    if part==1
        if cont==0
            pos(1:i,4)=2;
            cont=cont+1;
        else
            fil_part=i;
        end
    elseif support==1
        if cont==0
            pos(1:i,4)=2;
            cont=cont+1;
        else
            fil_support=i;
        end
    end
    if fil_support<fil_part&&fil_support>1
        pos(fil_support:fil_part,4)=2;
        fil_support=1;
        fil_part=1;
    end
end

%Si en la fila ha encontrado una X, una Y o/y una Z, con lo cual
%existe una posición, pero alguna de las coordenadas no aparece, toma
%el valor anterior (no es cero)
for i=2:1:k
    if isnan(pos(i,1))==1 %Si no se especifica el valor de X
        pos(i,1)=pos(i-1,1); %Toma el valor X de la posición anterior
    end
    if isnan(pos(i,2))==1 %Si no se especifica el valor de Y
        pos(i,2)=pos(i-1,2); %Toma el valor Y de la posición anterior
    end
    if isnan(pos(i,3))==1 %Si no se especifica el valor de Z
        pos(i,3)=pos(i-1,3); %Toma el valor Z de la posición anterior
    end
    if isnan(pos(i,5))==1 %Si no se especifica el valor de E
        pos(i,5)=pos(i-1,5); %Toma el valor E de la posición anterior
    end
```

```
    end

    %LIMPIEZA
    fila=1;
    posf=zeros(1,5);
    for i=1:1:k
      if isnan(pos(i,4))==0 && pos(i,4)<2
        posf(fila,1:5)=pos(i,1:5);
        fila=fila+1;
      end
    end

    %PREPARACIÓN MATRIZ COORDENADAS
    [filf,colf]=size(posf);
    inicio=find(posf(:,5)==0);
    coordenadas=posf(inicio(1):filf,1:5);
    [filc,colc]=size(coordenadas);
    minx=coordenadas(1,1);
    miny=coordenadas(1,2);
    for i=1:1:filc
      if coordenadas(i,1)<minx
        minx=coordenadas(i,1);
      end
      if coordenadas(i,2)<miny
        miny=coordenadas(i,2);
      end
    end
    minx=minx-2;
    miny=miny-2;
    for i=1:1:filc
      coordenadas(i,1)=(coordenadas(i,1)-minx)/1000;
      coordenadas(i,2)=(coordenadas(i,2)-miny)/1000;
      coordenadas(i,3)=coordenadas(i,3)/1000;
    end

end %Fin de la función
```

ANEXO D. FUNCIÓN COORD_GEN_FULLCONTROL

```
function [coordinates,fil_D]=Coord_Gen_FullControl(filename_gcode)
  % Entrada de diámetro de filament
  fil_D=str2num(input('Enter the FeedstockFilamentDiameter defined in
  FullControl (only number in mm, e.g. 1.75): ','s'));

  %INTRODUCCIÓN ARCHIVO
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  CI=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
  %s %s %s',[inf,20]); %Lectura de todas las filas y 20 columnas
  kc=size(CI{2},1); %Determinación de la dimensión máxima de filas de
  la matriz (kc)
  fclose(ARCH); %Cierre de archivo
  ARCH=fopen(filename_gcode,'r'); %Nueva apertura
  C=textscan(ARCH,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
  %s %s %s',[kc,20]); %Lectura de 'kc' filas y 20 columnas
  c=[C{1} C{2} C{3} C{4} C{5}]; %Creación de matriz de tipo string con
  todo el archivo
  fclose(ARCH); %Cierre del archivo
  ARCH=fopen(filename_gcode,'r'); %Apertura del archivo
  COM=textscan(ARCH,'%s %s','Delimiter',';');
  comments=[COM{2}];
  fclose(ARCH); %Cierre de archive

  %ELIMINACIÓN DE COMENTARIOS
  k=size(c,1); %Número de columnas de la matriz 'c'
  for i=1:1:k
    for j=1:1:5
      f=strfind(c{i,j},';'); %Busca en cada celda un ';'
      if f>0
        for m=j:1:5
          c{i,m}=' '; %Si lo encuentra borra esa celda y todas las
          siguientes de la fila
        end
      end
    end
  end

  %ELIMINACIÓN DE COMANDOS 'M', 'S', 'G28', 'G' y 'F'
  for i=1:1:k
    for j=1:1:5
      M=strfind(c{i,j},'M'); %Busca el comando 'M' celda por celda
      V=strfind(c{i,j},'G28'); %Busca el comando 'G28' celda por celda
      S=strfind(c{i,j},'S'); %Busca el comando 'S' celda por celda
      F=strfind(c{i,j},'F'); %Busca el comando 'F' celda por celda
      if M==1
        c{i,j}=''; %Borra la celda donde se encuentra el comando 'M'
      elseif V==1 %Borra G28 y todo lo que sigue al comando
        c{i,j}='';
        c{i,j+1}='';
        c{i,j+2}='';
      elseif S==1
        c{i,j}=''; %Borra la celda donde se encuentra el comando 'S'
      elseif F==1
        c{i,j}=''; %Borra la celda donde se encuentra el comando 'F'
      end
    end
  end

  %VECTOR POSICIÓN + E
```

```
pos=zeros(k,5); %Inicialización matriz 'pos'
for i=1:1:k
   for j=1:1:5
      X=strfind(c{i,j},'X'); %Busca 'X' celda por celda
      Y=strfind(c{i,j},'Y'); %Busca 'Y' celda por celda
      Z=strfind(c{i,j},'Z'); %Busca 'Z' celda por celda
      E=strfind(c{i,j},'E'); %Busca 'E' celda por celda
      H=strfind(c{i,j},'G0'); %Busca 'G0' celda por celda
      G=strfind(c{i,j},'G1'); %Busca 'G1' celda por celda
      if X==1
         cp{i,1}=strrep(c{i,j},'X',''); %Deletes 'X' para solamente
         conservar los números
         pos(i,1)=str2double(cp{i,1}); %Trasfiere los números al vector
         de posición (pos)
      elseif Y==1
         cp{i,2}=strrep(c{i,j},'Y',''); %Elimina 'Y' para solamente
         conservar los números
         pos(i,2)=str2double(cp{i,2}); %Trasfiere los números al vector
         de posición (pos)
      elseif Z==1
         cp{i,3}=strrep(c{i,j},'Z',''); %Elimina 'Z' para solamente
         conservar los números
         pos(i,3)=str2double(cp{i,3}); %Trasfiere los números al vector
         de posición (pos)
      elseif E==1
         cp{i,5}=strrep(c{i,j},'E',''); %Elimina 'E' para solamente
         conservar los números
         pos(i,5)=str2double(cp{i,5}); %Trasfiere los números al vector
         de posición (pos)
      elseif H==1
         cp{i,4}=strrep(c{i,j},'G',''); %Elimina 'G' para solamente
         conservar los números
         pos(i,4)=str2double(cp{i,4}); %Trasfiere el número al vector
         de posición (pos)
         ho=i;
      elseif G==1
         cp{i,4}=strrep(c{i,j},'G',''); %Elimina 'G' para solamente
         conservar los números
         pos(i,4)=str2double(cp{i,4}); %Trasfiere el número al vector
         de posición (pos)
      end
   end
end

%Inicio y fin de G-code (sin comentarios)
[k,y]=size(pos);
for i=1:1:k
   if isequal(comments(i),{'END OF THE START GCODE'})
      for j=i:-1:1
         pos(j,4)=2; %Asigna un 2 a todas las filas anteriores
      end
   elseif isequal(comments(i),{'START OF THE END GCODE'})
      for j=i:1:k
         pos(j,4)=2; %Asigna un 2 a todas las filas posteriores
      end
   end
end

fila=1;
pos_3=pos(:,1:3); %Toma parte de matriz de coordenadas XYZ
com=sum(pos_3,2); %Suma coordenadas XYZ de cada fila
```

```
    [k,y]=size(com);
    posf=zeros(1,5); %Nuevo vector posición

    %Si la sumatoria de coordenadas es superior a 0 y los valores de la
    cuarta fila son 0 ó 1, graba la línea en el nuevo vector
    for i=1:1:k
      if com(i,1)>0 && pos(i,4)<2
        posf(fila,1:5)=pos(i,1:5);
        fila=fila+1;
      end
    end

    %Sumatoria de E (valores absolutos)
    [filf,colf]=size(posf);
    for i=2:1:filf
      posf(i,5)=posf(i,5)+posf(i-1,5);
    end

    %Si en la fila ha encontrado una X, una Y o/y una Z, con lo cual
    %existe una posición, pero alguna de las coordenadas no aparece, %toma
    el valor anterior (no es cero)
    for i=2:1:filf
      if posf(i,1)==0 %Si no se especifica el valor de X
        posf(i,1)=posf(i-1,1); %Toma el valor X de la posición anterior
      end
      if posf(i,2)==0 %Si no se especifica el valor de Y
        posf(i,2)=posf(i-1,2); %Toma el valor Y de la posición anterior
      end
      if posf(i,3)==0 %Si no se especifica el valor de Z
        posf(i,3)=posf(i-1,3); %Toma el valor Z de la posición anterior
      end
    end

    %Trasladar la pieza al punto (2,2,0) y pasar a metros
    coordinates=posf(1:filf,1:5);
    [filc,colc]=size(coordinates);
    minx=coordinates(1,1);
    miny=coordinates(1,2);
    for i=1:1:filc
      if coordinates(i,1)<minx
        minx=coordinates(i,1);
      end
      if coordinates(i,2)<miny
        miny=coordinates(i,2);
      end
    end
    minx=minx-2;
    miny=miny-2;
    for i=1:1:filc
        coordinates(i,1)=(coordinates(i,1)-minx)/1000;
        coordinates(i,2)=(coordinates(i,2)-miny)/1000;
        coordinates(i,3)=coordinates(i,3)/1000;
    end

end %Fin de función
```

```
%NOMBRE FICHEROS
filename_xlsx=input('Introduzca el nombre del archivo Excel (terminado
en .xlsx): ','s');
filename_txt=input('Introduzca el nombre del archivo de Gmsh (terminado
en .txt): ','s');

%LECTURA COORDENADAS
COORD=xlsread(filename_xlsx);
[m,n]=size(COORD); %Dimensiones tabla; m = número de coordenadas
COORD_mm=COORD*1000; %Coordenadas en mm
[row,col]=find(COORD_mm(:,4) == 0);
row=[row
    m+1];
row_size=size(row,1);
fil_D=2.85;
h_layer=COORD_mm(row(1),3);
h_newlayer=h_layer;
surface=0;
n_loop=1;
n_surface=1;
first_point=0;
n_line=0;
contador=0;
cerrado=0;

[row,col]=find(COORD_mm(:,4) <= 0);
row=[row
    m+1];
row_size=size(row,1);

%Generación .TXT
FILE=fopen(filename_txt,'w');
for j=1:1:2%row_size-2
  %Definición de tramo
  START=row(j,1);
  END=row(j+1,1)-1;
  h_newlayer=COORD_mm(j,3);
  %Altura de capa
  if h_newlayer==COORD_mm(1,3)
    height=h_newlayer;
  elseif h_newlayer==h_layer
    height=height;
  else
    height=h_newlayer-h_layer;
  end
  %Determinación de ancho extrusión
  %Sumatoria distancias
  dist=0;
  for i=START:1:END-1
    p1=COORD_mm(i,1:2);
    p2=COORD_mm(i+1,1:2);
    dist_n=sqrt(sum((p1-p2).^2));
    dist=dist+dist_n;
  end
  %Diferencia E
  AE=(COORD_mm(END,5)-COORD_mm(START,5))/1000;
  %Ancho de extrusión
  width=height+(AE*pi*(fil_D/2)^2/dist-pi*height^2/4)/height;
  %Comprobación trayectorias cerradas
```

```
dist_ext=COORD_mm(row(j),1:2)-COORD_mm(row(j+1)-1,1:2);
if sqrt(sum(dist_ext.^2))<(width/2)
   cerrado=1; %La trayectoria es cerrada
else
   cerrado=0; %La trayectoria es abierta
end
for i=START:1:END-1
   if i==START
      %Perfil
      if cerrado==0 %Si la trayectoria es abierta
         P1=COORD_mm(i,1:3);
         P2=COORD_mm(i+1,1:3);
         %if START~=(END-1)
         P3=COORD_mm(i+2,1:3);
         [pc1,angle1,p_1,pp,m1,T1]=Corte_trayectoria(width,P1,P2,P3);
         %end
      else %Si la trayectoria es cerrada
         if COORD_mm(START,1:3)==COORD_mm(END,1:3)
            P4=COORD_mm(END-1,1:3);
            P1=COORD_mm(i,1:3);
            P2=COORD_mm(i+1,1:3);
            P3=COORD_mm(i+2,1:3);
            [pc0,angle0,p_01,p_02,M0,T0,sa1]=Corte_trayectoria(width,
            P4,P1,P2);
            [pc1,angle1,p_11,p_12,m1,T1]=Corte_trayectoria(width,P1,P2,
            P3);
            P1=p_02(1:2);
         else
            %Completar
         end
      end

      %Coordenadas perfil
      Pnt1=(width-height)/2;
      Pnt2=width/2;
      X=[Pnt1;-Pnt1;-Pnt1;Pnt1;Pnt1;-Pnt1;Pnt2;-Pnt2];
      Y=X;
      Z=[h_newlayer;h_newlayer;h_newlayer-height;h_newlayer-height;
      h_newlayer-height/2;h_newlayer-height/2;h_newlayer-height/2;
      h_newlayer-height/2];

      %Rotación de perfil (perpendicular a trayectoria)
      if abs(m1)==Inf
         X=X*sind(T1)+P1(1);
         Y=Y*cosd(T1)+P1(2);
      elseif m1<=0
         X=X*cosd(T1)+P1(1);
         Y=Y*sind(T1)+P1(2);
      else
         X=X*cosd(T1)+P1(1);
         Y=Y*(-sind(T1))+P1(2);
      end
      perfil=[X Y Z];

      %Definición perfil
      perfil_pyc=[[1 2 3 4 5 6 7 8]+first_point;transpose(perfil)];
      lines=transpose([n_line+1  [2  1]+first_point;n_line+2  [4  3]+
      first_point]);
      circles=transpose([n_line+3 [2 6 8]+first_point;n_line+4 [3 6 8]+
      first_point;n_line+5  [1  5  7]+first_point;n_line+6  [4  5  7]+
      first_point]);
```

```
    loop=[n_loop [1 5]+n_line -6-n_line [2 4]+n_line -3-n_line];
    n_loop=n_loop+1;
    surface=surface+1;
    fprintf(FILE,'Point(%d) = {%1.3f, %1.3f, %1.3f, 1.0};\nPoint(%d)
    = {%1.3f, %1.3f, %1.3f, 1.0};\nPoint(%d) = {%1.3f, %1.3f, %1.3f,
    1.0};\nPoint(%d) = {%1.3f, %1.3f, %1.3f, 1.0};\nPoint(%d) =
    {%1.3f, %1.3f, %1.3f, 1.0};\nPoint(%d) = {%1.3f, %1.3f, %1.3f,
    1.0};\nPoint(%d) = {%1.3f, %1.3f, %1.3f, 1.0};\nPoint(%d) =
    {%1.3f, %1.3f, %1.3f, 1.0};\nLine(%d) = {%d, %d};\nLine(%d) =
    {%d, %d};\nCircle(%d) = {%d, %d, %d};\nCircle(%d) = {%d, %d,
    %d};\nCircle(%d) = {%d, %d, %d};\nCircle(%d) = {%d, %d,
    %d};\nCurve Loop(%d) = {%d, %d, %d, %d, %d, %d};\nPlane
    Surface(%d) = {%d};\n',perfil_pyc(:),lines(:),circles(:),
    loop(:),surface,n_surface);
    n_surface=n_surface+1;

    %Extrusión
    if cerrado==0
      if START~=(END-1)
        ext_dist=p_1-P1(1:2);
      else
        ext_dist=P2(1:2)-P1(1:2);
      end
    else
      ext_dist=p_11-p_02;
    end

    fprintf(FILE,'Extrude {%1.3f, %1.3f, 0} {\nSurface{%d};\n}\n',
    ext_dist(:),surface);
    if contador==0
      surface=surface+37;
      contador=1;
    else
      surface=surface+32;
    end
    first_point=first_point+8;
    n_line=n_line+8;

  elseif i==END-1
    P1=COORD_mm(i-1,1:3);
    P2=COORD_mm(i,1:3);
    P3=COORD_mm(i+1,1:3);
    [pc1,angle1,p_1,p_2,M1,T1,sa]=Corte_trayectoria(width,P1,P2,
    P3);
    angle=sa*angle1/pi;
    if cerrado==1
      if P3==COORD_mm(START,1:3)
        P4=COORD_mm(START+1,1:3);
        [pc2,angle2,p_21,p_22,M2,T2,sa2]=Corte_trayectoria(width,P2
        ,P3,P4);
        angle_2=sa2*angle2/pi;
      else
        %Completar: programar si el último punto no es como el
        primero de la trayectoria
      end
    end

    %Extrusión rotación
    fprintf(FILE,'Extrude {{0, 0, 1}, {%1.3f, %1.3f, %1.3f},
    %1.2f*Pi} {\nSurface{%d};\n}\n',pc1(:),COORD_mm(i,3)-height/2,
    angle,surface);
```

```
        surface=surface+30;
        n_line=n_line+32;
        first_point=first_point+52;

        %Extrusión lineal
        if cerrado==0
          ext_dist=P3(1:2)-p_2;
        else
          ext_dist=p_21-p_2;
        end
        fprintf(FILE,'Extrude {%1.3f, %1.3f, 0} {\nSurface{%d};\n}\n',
        ext_dist(:),surface);
        first_point=first_point+21*2;
        n_line=n_line+30+24;
        surface=surface+32;

        %Si la trayectoria es cerrada se realiza otra rotación
        if cerrado==1
          fprintf(FILE,'Extrude {{0, 0, 1}, {%1.3f, %1.3f, %1.3f},
          %1.2f*Pi}          {\nSurface{%d};\n}\n',pc2(:),COORD_mm(i,3)-
          height/2,angle_2,surface);
          surface=surface+30;
          n_line=n_line+32;
          first_point=first_point+52;
        end

      else %Puntos intermedios
        %Puntos de rotación
        P1=COORD_mm(i-1,1:3);
        P2=COORD_mm(i,1:3);
        P3=COORD_mm(i+1,1:3);
        P4=COORD_mm(i+2,1:3);
        [pc1,angle1,p_11,p_12,M1,T1,sa1]=Corte_trayectoria(width,P1,P2,
        P3);
        [pc2,angle2,p_21,p_22]=Corte_trayectoria(width,P2,P3,P4);
        angle=sa1*angle1/pi;

        %Extrusión rotación
        fprintf(FILE,'Extrude {{0, 0, 1}, {%1.3f, %1.3f, %1.3f},
        %1.2f*Pi}  {\nSurface{%d};\n}\n',pc1(:),COORD_mm(i,3)-height/2,
        angle,surface);
        surface=surface+30;
        n_line=n_line+32;
        first_point=first_point+52;

        %Extrusión lineal
        ext_dist=p_21-p_12;
        fprintf(FILE,'Extrude {%1.3f, %1.3f, 0} {\nSurface{%d};\n}\n',
        ext_dist(:),surface);
        surface=surface+32;
        n_line=n_line+30;
        first_point=first_point+21;
      end
    end
end

fclose(FILE);
```

```
%NOMBRE FICHEROS
filename_xlsx=input('Introduzca el nombre del archivo Excel (terminado
en .xlsx): ','s');

k=1;
%LECTURA COORDENADAS
COORD=xlsread(filename_xlsx);
[m,n]=size(COORD); %Dimensiones tabla; m = número de coordenadas
COORD_mm=COORD*1000; %Coordenadas en mm

[row,col]=find(COORD_mm(:,4) == 0);
row_size=size(row,1);

for j=1:1:row_size-1
   i_start=row(j)+1;
   i_end=row(j+1)-2;
   for i=i_start:1:i_end
      %Hallar ángulos agudos
      p1=COORD_mm(i-1,1:2);
      p2=COORD_mm(i,1:2);
      p3=COORD_mm(i+1,1:2);
      v1=p1-p2;
      v2=p3-p2;
      mag_v1=sqrt(sum(v1.^2));
      mag_v2=sqrt(sum(v2.^2));
      angle=acosd(dot(v1,v2)/(mag_v1*mag_v2));

      if angle<90 || mag_v1<0.24 || mag_v2<0.24
         COORD_mm(i,4)=-1;
      end
   end
end

[row,col]=find(COORD_mm(:,4) <= 0);
row=[row
    m+1];
row_size=size(row,1);
fil_D=2.85;
h_layer=COORD_mm(row(1),3);
h_newlayer=h_layer;
n_points=0;

%Número máximo de puntos de trayectoria
for i=1:1:row_size-1
   n_pnt=row(i+1)-row(i);
   if n_pnt>n_points
      n_points=n_pnt;
   end
end

STOP=2;
n_pts_sum=0;
for j=1:1:row_size-2
   START=row(j,1);
   %END=row(j+1,1)-1;
   if COORD_mm(row(j+1,1),4)==0
      END=row(j+1,1)-1;
   else
```

```
   END=row(j+1,1);
end

if STOP>=1
   if STOP==1
      fprintf(FILE,'End Sub');
      fclose('all');
   end

   %NOMBRE FICHERO
   filename = sprintf('file%d.cls', k);

   %Generación .TXT
   FILE=fopen(filename,'w');

   %Cabecera
   fprintf(FILE,'Option Explicit\n\nPublic Sub SweepFeature()\n'' Set
   a reference to the currently active document.\n'' This assumes that
   it is a part document.\nDim oPartDoc As PartDocument\nSet oPartDoc
   = ThisApplication.ActiveDocument\n\n'' Set a reference to the
   component definition.\nDim oCompDef As PartComponentDefinition\nSet
   oCompDef = oPartDoc.ComponentDefinition\n\n'' Set a reference to
   the transient geometry object.\nDim oTG As TransientGeometry\nSet
   oTG = ThisApplication.TransientGeometry\n\n');

   %Inicializar puntos de trabajo, croquis 3D, líneas de croquis 3D,
   %trayectoria, croquis del perfil, coordenadas, líneas y arcos de
   %perfil, perfil, barrido y plano de trabajo para nueva capa
   fprintf(FILE,''' Create some workpoints that will be used for the
   3D sketch.\nDim oWPs(1 To %d) As WorkPoint\n'' Create a new 3D
   Sketch.\nDim oSketch3D As Sketch3D\n'' Create 3D lines.\nDim
   oLine(1 To %d) As SketchLine3D\n''Create Path\nDim oPath As
   Path\n''Create a sketch containing the profile\nDim oSketch As
   PlanarSketch\n''Create coordinates\nDim oCoord1 As Point2d\nDim
   oCoord2 As Point2d\n''Create profile lines and arcs\nDim oLines(1
   To 2) As SketchLine\nDim oArcs(1 To 2) As SketchArc\n''Create a
   profile.\nDim oProfile As Profile\n''Create the sweep feature.\nDim
   oSweep As SweepFeature\n'' Create a work plane at the end of the
   path sketch.\nDim oWP As WorkPlane\n\n',n_points,n_points);

   write_end=1;
   STOP=0;
   k=k+1;
end

%Contador de líneas
n_pts=END-START+1;
n_pts_sum=n_pts_sum+26+2*n_pts;
if n_pts_sum>967
   STOP=1;
   n_pts_sum=0;
end

h_newlayer=COORD_mm(j,3);

%Altura de capa
if h_newlayer==COORD_mm(1,3)
   height=h_newlayer;
elseif h_newlayer==h_layer
   height=height;
```

```
else
   height=h_newlayer-h_layer;
end

%Determinación de width
%Sumatoria distancias
dist=0;
for i=START:1:END-1
p1=COORD_mm(i,1:2);
p2=COORD_mm(i+1,1:2);
dist_n=sqrt(sum((p1-p2).^2));
dist=dist+dist_n;
end
%Diferencia E
AE=(COORD_mm(END,5)-COORD_mm(START,5))/1000;
%Ancho de extrusión
width=height+(AE*pi*(fil_D/2)^2/dist-pi*height^2/4)/height;

for i=START:1:END
   %Definición coordenadas de puntos de trabajo
   wp=i-START+1;
   fprintf(FILE,'Set    oWPs(%d)    =    oCompDef.WorkPoints.AddFixed(
   oTG.CreatePoint(%3.3f, %3.3f, %3.3f))\n',wp,COORD_mm(i,1:3));
end

%Crear croquis 3D
fprintf(FILE,'Set    oSketch3D    =    oPartDoc.ComponentDefinition.
Sketches3D.Add\n\n');
fprintf(FILE,''' Draw 3D lines. The first line is drawn between two
of the work points.\n');
for j=1:1:wp-1
   %Dibujar líneas
   if j==1
      fprintf(FILE,'Set    oLine(%d)    =    oSketch3D.SketchLines3D.
      AddByTwoPoints(oWPs(%d), oWPs(%d), False)\n',j,j,j+1);
   else
      fprintf(FILE,'Set    oLine(%d)    =    oSketch3D.SketchLines3D.
      AddByTwoPoints(oLine(%d).EndPoint,   oWPs(%d),   False)\n',j,j-
      1,j+1);
   end
end

%Crear trayectoria
fprintf(FILE,'\n'' Create a path.\nSet oPath = oCompDef.Features.
CreatePath(oLine(1))\n');

%Definir el plano de trabajo
fprintf(FILE,'\n'' Create a work plane at the end of the path
sketch.\nSet oWP = oCompDef.WorkPlanes.AddByNormalToCurve(oLine(1),
oLine(1).StartSketchPoint)\nSet    oSketch    =    oCompDef.Sketches.
Add(oWP)');

%Crear perfil
fprintf(FILE,'\n''Create    profile    sketch\nSet    oCoord1    =
oTG.CreatePoint2d(%1.2f,        %1.2f)\nSet        oCoord2        =
oTG.CreatePoint2d(%1.2f,        %1.2f)\nSet        oLines(1)        =
oSketch.SketchLines.AddByTwoPoints(oCoord1, oCoord2)\nSet oCoord1 =
oTG.CreatePoint2d(%1.2f,        %1.2f)\nSet        oCoord2        =
oTG.CreatePoint2d(%1.2f,        %1.2f)\nSet        oArcs(1)        =
oSketch.SketchArcs.AddByCenterStartEndPoint(oCoord1,
```

```
    oLines(1).EndSketchPoint,        oCoord2)\nSet        oCoord1        =
    oTG.CreatePoint2d(%1.2f,        %1.2f)\nSet        oLines(2)        =
    oSketch.SketchLines.AddByTwoPoints(oArcs(1).EndSketchPoint,
    oCoord1)\nSet oCoord1 = oTG.CreatePoint2d(%1.2f, %1.2f)\nSet oArcs(2)
    =                oSketch.SketchArcs.AddByCenterStartEndPoint(oCoord1,
    oLines(2).EndSketchPoint,   oLines(1).StartSketchPoint)\n''Create   a
    profile.\nSet  oProfile  =  oSketch.Profiles.AddForSolid\n',-(width-
    height)/2,0,(width-height)/2,0,(width-height)/2,height/2,(width-
    height)/2,height,-(width-height)/2,height,-(width-
    height)/2,height/2);

    %Crear barrido
    fprintf(FILE,'\n''Create   the   sweep   feature.\nSet   oSweep   =
    oCompDef.Features.SweepFeatures.AddUsingPath(oProfile,        oPath,
    kJoinOperation)\n\n');

end
```

ANEXO G. APLICACIÓN DE METODOLOGÍA DE MODELADO A MINIPROBETA SOMETIDA A FLEXIÓN PASO A PASO

En este anexo, se presenta el paso a paso para el modelado de una miniprobeta de ensayo, tanto con la metodología DECODE (basada en geometría) como con VOLCO (basada en vóxeles). Se describe el proceso desde la generación de coordenadas con el archivo de código G, hasta la interpretación de resultados del análisis de elementos finitos (AEF) en Abaqus/CAE 6.14-1. En este caso, se ha aplicado un ensayo a flexión.

Tabla de contenidos

i. Modelado basado en geometría

Paso 1: Ejecutar *Abaqus_model*, indicando en nombre del archivo G-code del que extraerá las coordenadas y el archivo Python que generará.

Paso 2: Abrir Abaqus/CAE 6.14-1 y ejecutar Python, mediante *Run Script*.

Paso 3: Definir mallado del modelo. El mallado se realizará en el módulo *Mesh*. En primer lugar, se definirá el tamaño de la malla en *Seed Part* (Imagen 1).

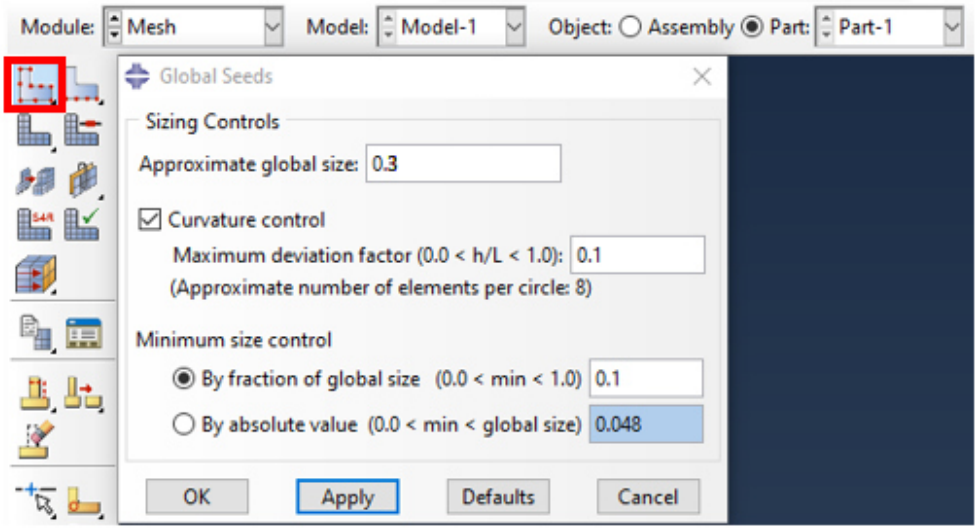


Imagen 1. Definición de tamaño de malla

A continuación, se definirá el tipo de malla, en este caso, tetraédrica, en *Assign Mesh Controls* (Imagen 2).

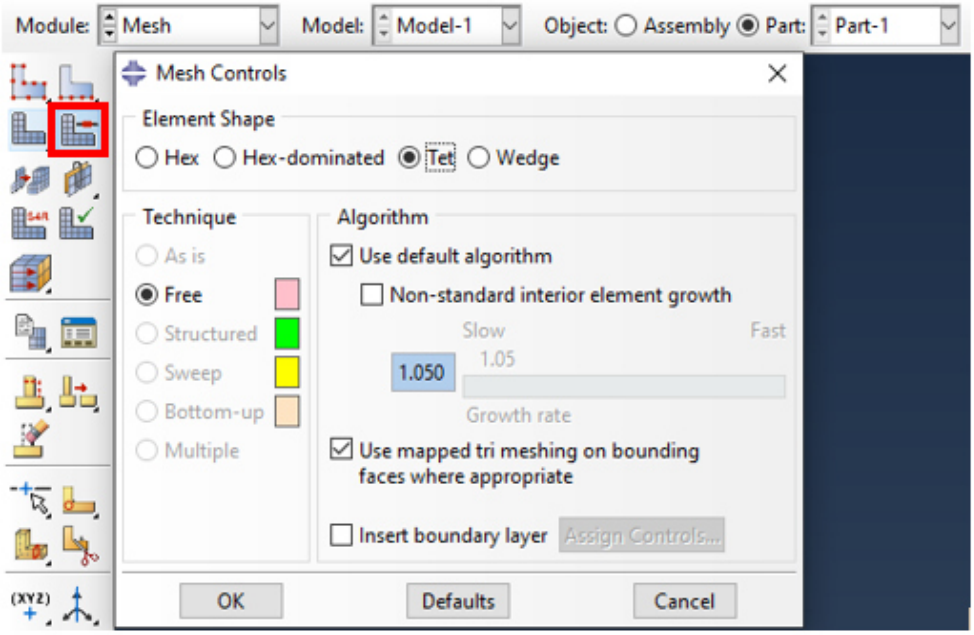


Imagen 2. Asignación forma de malla

En *Assign Element Type* (Imagen 3) se puede seleccionar el orden geométrico: linear (C3D4) o cuadrático (C3D10).

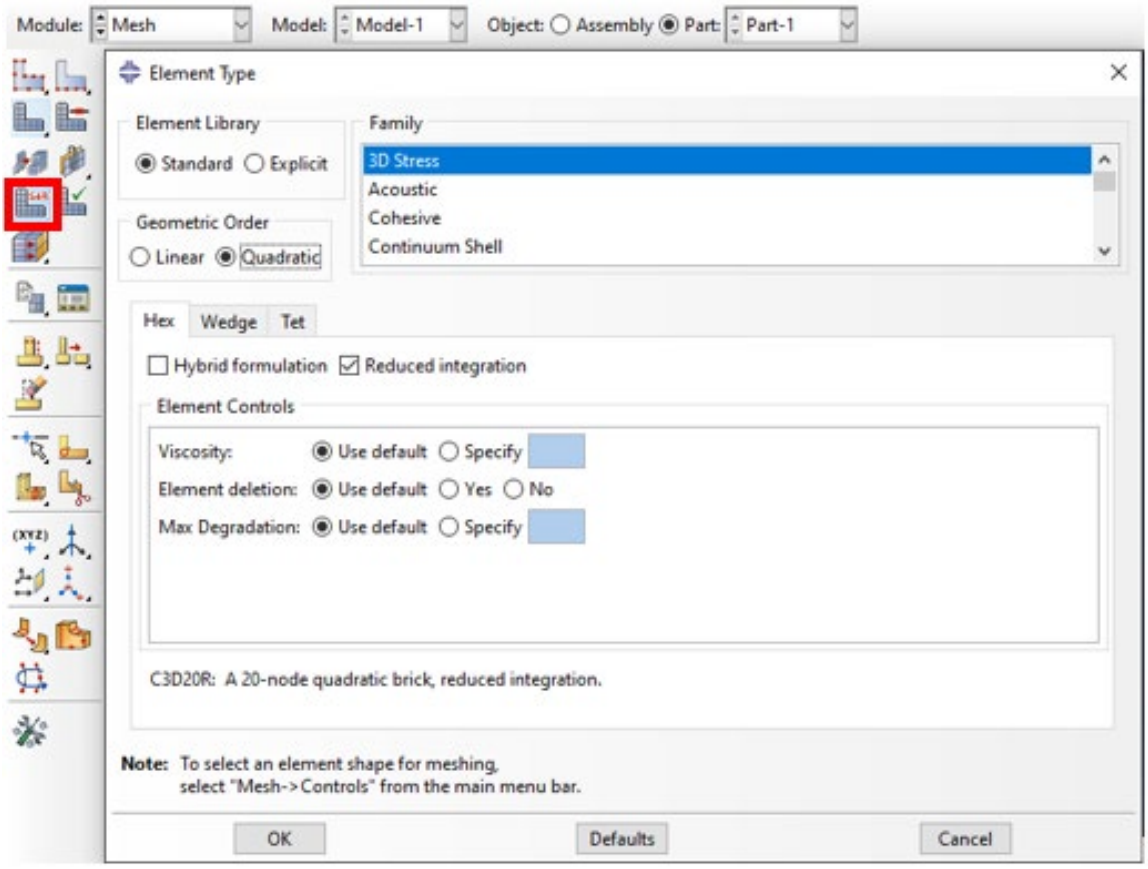


Imagen 3. Asignar tipo de malla (orden geométrico)

Paso 4: (Opcional) Si existen errores al mallar, una posible solución es cambiar el tamaño de semilla. Si sigue sin funcionar, otra opción es aplicar *Virtual Topology* (Imagen 4).

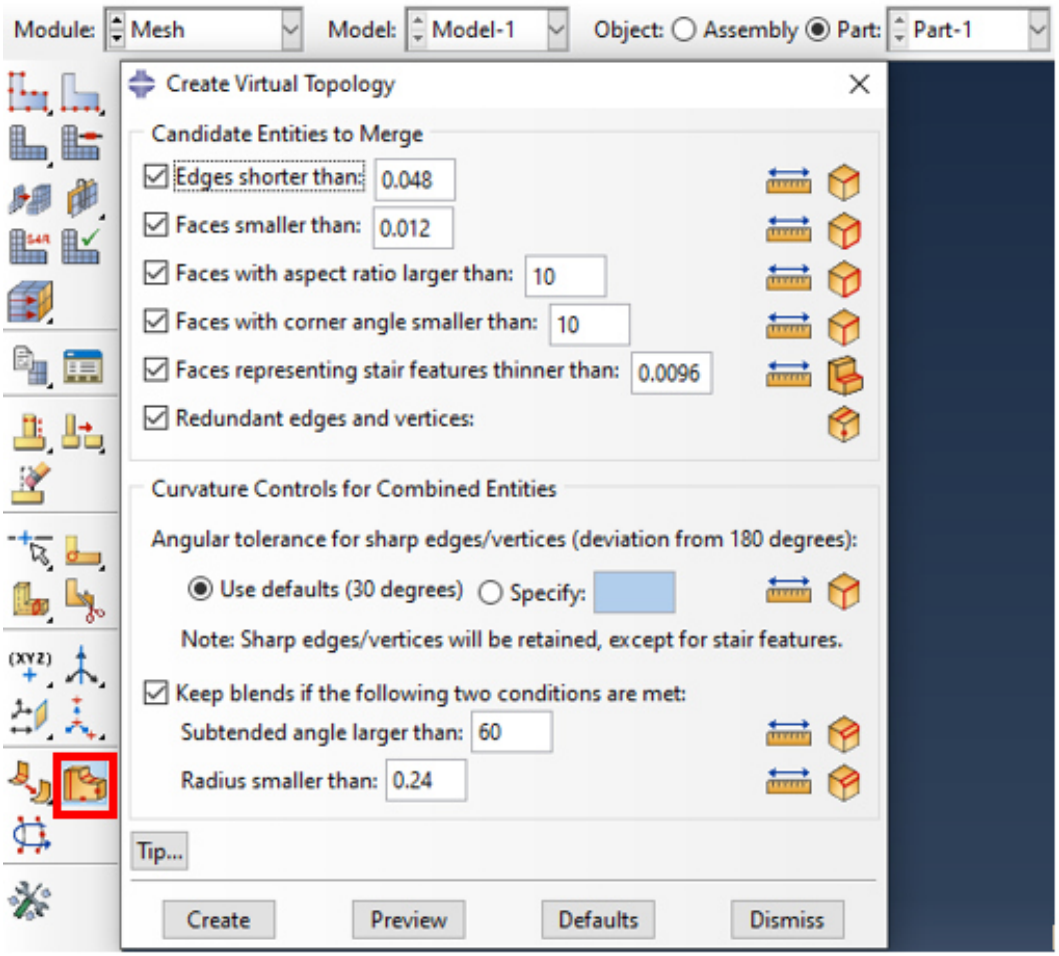


Imagen 4. Aplicación de Virtual Topology

Paso 5: Mallar modelo. En el módulo *Mesh*, clicando en el botón *Mesh Part* (Imagen 5), se malla la pieza.

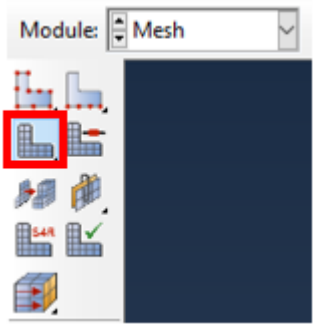


Imagen 5. Botón "Mesh Part" para el mallado

Paso 6: En el módulo *Assembly*, se introduce la pieza como parte única del ensamblaje, en *Create Instance* (Imagen 6).

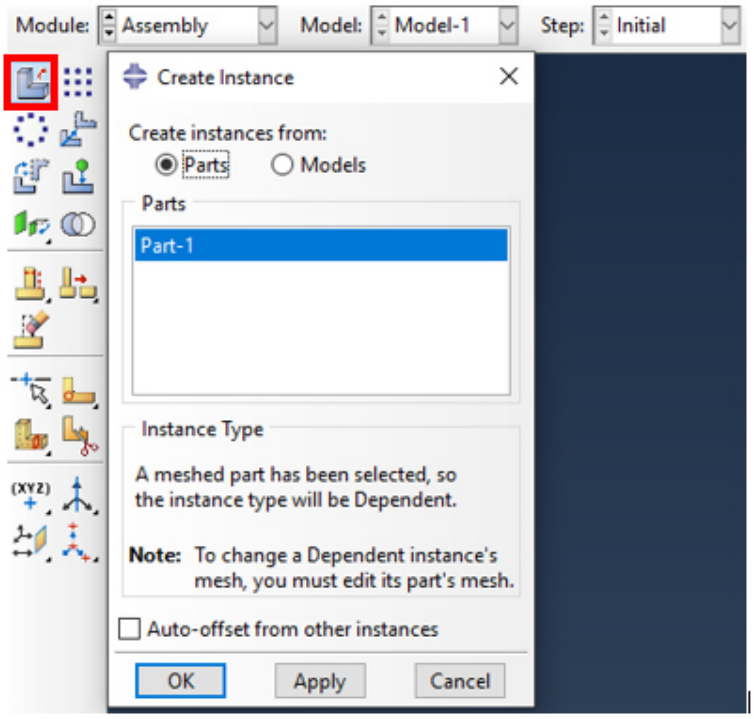


Imagen 6. Introducir pieza en el ensamblaje

El modelo se ve representado en la Imagen 7.

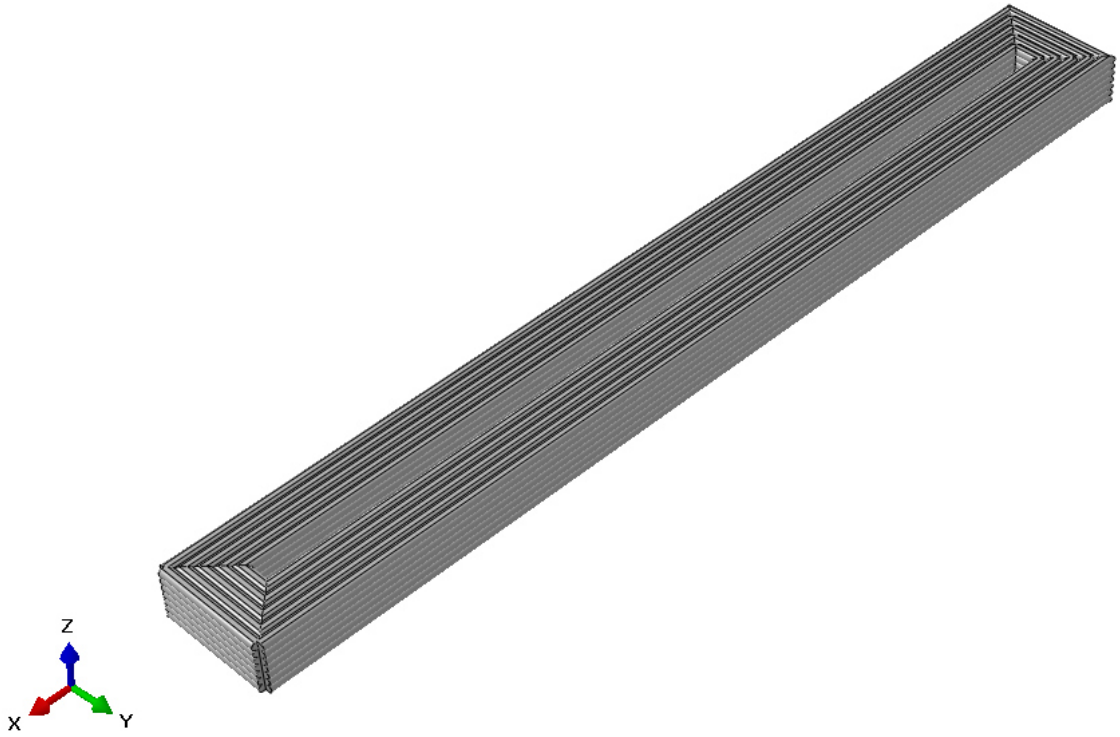


Imagen 7. Modelo basado en geometría de la miniprobeta

Paso 7: Calcular volumen y dimensiones. Esto se puede realizar desde *Tools > Query… > Point / Distance / Mass properties* (Imagen 8).

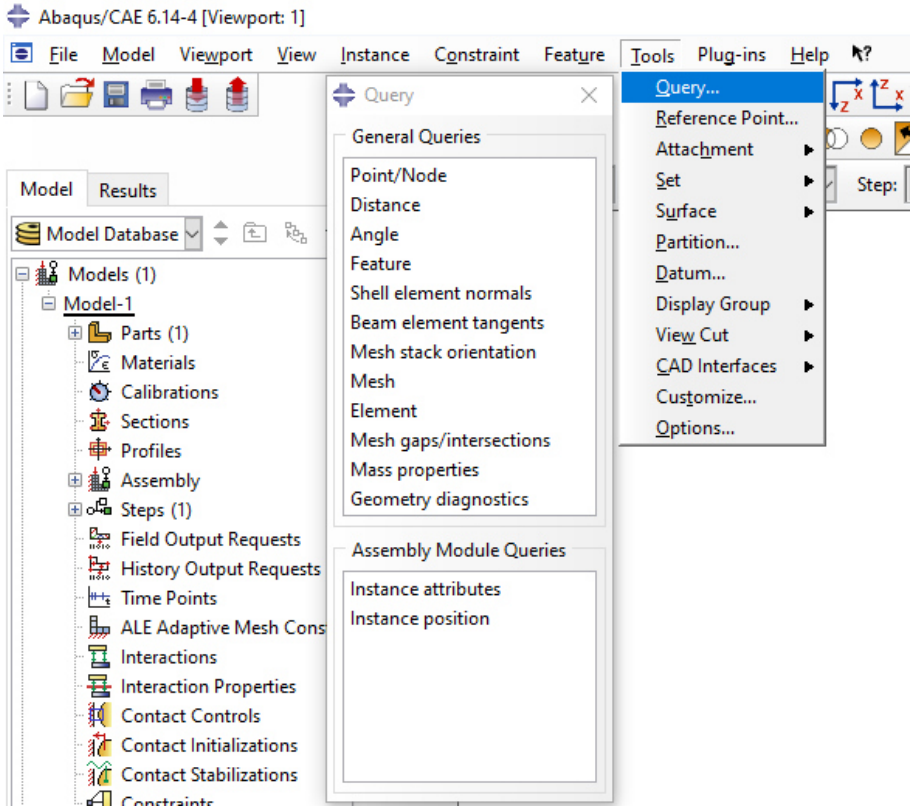


Imagen 8. Tomar datos puntos, dimensiones o masa

ii. Modelado basado en vóxeles

Paso 1: Generar coordenadas a partir del archivo G-code en la subrutina correspondiente, dependiendo del laminador utilizado (*Coord_Gen_Slic3r*, *Coord_Gen_Simplify* o *Coord_Gen_FullControl*).

Paso 2: Introducir coordenadas en la pestaña *Toolpath* del *Setup.xlsm* (VOLCO).

Paso 3: Cambiar datos en *Parameters for the voxel simulation* (Imagen 9) y *Parameters for the STL output from MATLAB and to set up the model in ABAQUS* (Imagen 10).

Parameters for the voxel simulation				
Name	Value	Units	SI value	SI units
VoxelSize	100	µm	0.0001	m
SizeX	50	mm	0.05	m
SizeY	15	mm	0.015	m
SizeZ	5	mm	0.005	m
VoxelsX	500		500	
VoxelsY	150		150	
VoxelsZ	50		50	
OriginalSphereRadius	198.923	µm	0.000199	m
SphereCentreBelowNozzle	198.923	µm	0.000199	m
GlueLayerThickness	0	µm	0	m
SphereStepRadiusIncreaseFraction	0.05		0.05	
MaxSphereIncreaseSteps	3000		3000	
nominalStep	101.4507	µm	0.000101	m
Toolpath offset X	2000	µm	0.002	m
Toolpath offset Y	2000	µm	0.002	m
Toolpath offset Z	-198.923	µm	-0.0002	m
MaxDepositionRadius	10000	µm	0.01	m

Imagen 9. Parámetros para la simulación del vóxel (miniprobeta)

Parameters for the STL output from MATLAB and to set up the model in ABAQUS				
Name	Value	Units	SI value	SI units
Xstart	0	mm	0	m
Xend	45	mm	0.045	m
Ystart	0	mm	0	m
Yend	7	mm	0.007	m
Zstart	0.2	mm	0.0002	m
Zend	3.2	mm	0.0032	m
Layer height	0.3	mm	0.0003	m
Start height	0.3	mm	0.0003	m
STL model name			VOLCO_minip7_5	
Job name			Job_VOLCO_TEMP	
Young's modulus	2.03E+09	Pa	2.03E+09	Pa
Poisson's ratio	0.36		0.36	
BOUNDARY CONDITIONS		DIRECTION		
NODES	TYPE	X	Y	Z
LOW_X	FIX	0	UNSET	UNSET
HIGH_X	PLANAR	ON	OFF	OFF
LOW_Y	FIX	UNSET	0	UNSET
HIGH_Y	PLANAR	OFF	ON	OFF
LOW_Z	FIX	UNSET	UNSET	0
HIGH_Z	DISP	UNSET	UNSET	-0.06

Imagen 10. Parámetros para el STL de salida de Matlab (miniprobeta)

Paso 4: Ejecutar *OUTPUT SETUP FILE* y *OUTPUT TOOLPATH* (Imagen 11Ilustración 11).

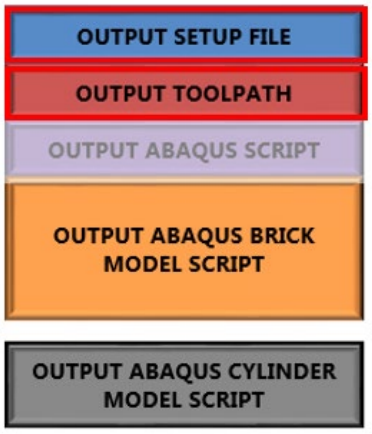


Imagen 11. Botonera para generar archivos de salida

Paso 5: Ejecutar *Voxel_Code_v8_IMPORT_GCODE.m* en Matlab.

Paso 6: Anotar datos que devuelve VOLCO al terminar de generar el STL (Imagen 12).

```
Elapsed time is 360.082179 seconds.
if sizeX / VoxelSize is not an integer then this is not ideal... the
region may not be a unit cell.
ans =
  0.044900000000000
Maxheight =
  2.100000000000000
SurfacePorosity =
  0.446857142857143
Volume =
   3.551690000000000e-07
ForCopying =
  0.446857142857143   0.000000355169000
```

Imagen 12. Datos de salida recogidos después de la generación del STL

Paso 7: Comparar el volumen del modelo con el teórico. Cambiar el radio de la esfera (*OriginalSphereRadius*) en los parámetros de simulación de vóxeles y, con los resultados, buscar el radio que genera una geometría con un volumen igual al teórico (usando el comando *spline(x,y,xq)* en Matlab). En la Tabla 1 se muestran el proceso de comparación de volúmenes, en el cual el modelo 5

(radio de esfera 198.923 µm) se ajusta más a la pieza teórica.

Tabla 1. Comparación volumen teórico y del modelo

	Teórico	Modelo				
		1	2	3	4	5
Volumen (mm^3)	355.17	197.5	359.034	354.73	355.159	355.169
Radio de esfera (µm)	-	150	200	198.8	198.92	198.923

Paso 8: Importar la pieza en Abaqus: *Plug-ins>Tools>STL import*. El modelo aparecerá representado mediante vóxeles (Imagen 13).

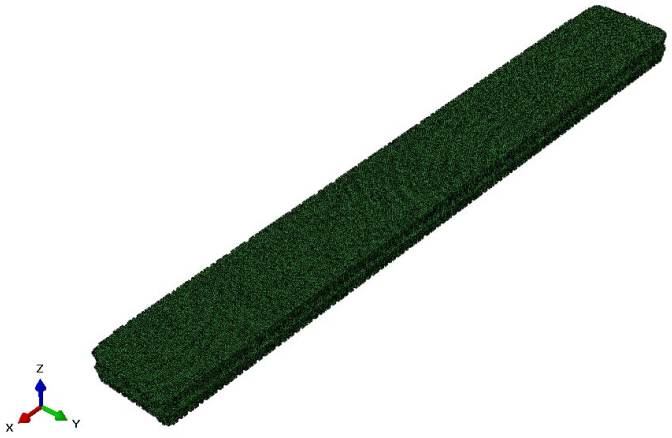


Imagen 13. Vista modelo miniprobeta en Abaqus

Paso 9: Para pasar de triángulos a tetraedros y obtener una malla C3D4 (elementos tetraédricos de cuatro nodos), se debe editar la malla en el módulo *Mesh*, *Mesh>Edit…>Mesh>Convert tri to tet* (Imagen 14).

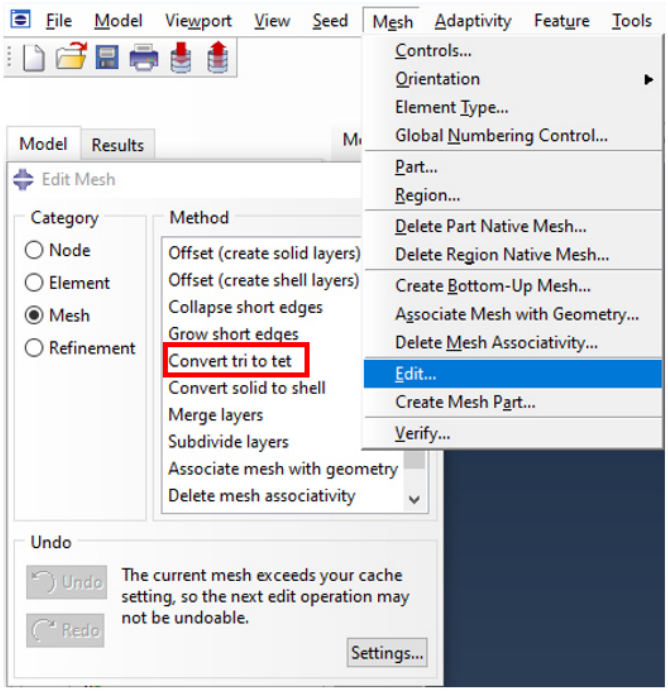
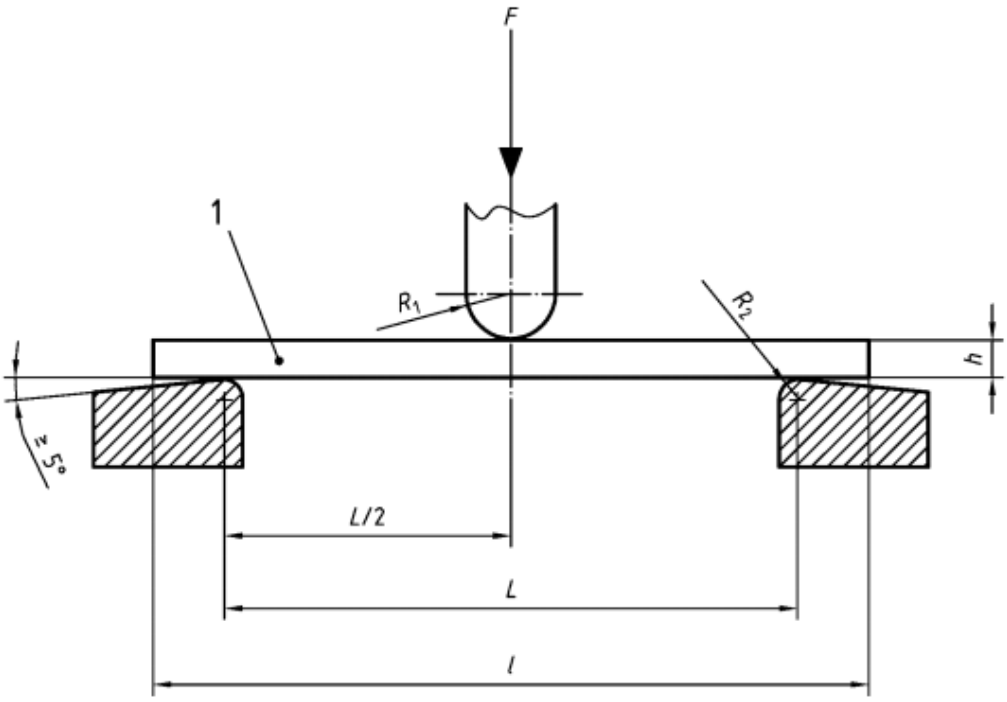


Imagen 14. Obtención de malla C3D4

iii. Análisis de elementos finitos

Paso 1: Definir los puntos donde irán ubicados los apoyos y la cruceta. Según la UNE-EN ISO 178:2001 (Plásticos – Determinación de las propiedades de flexión), la disposición debería ser la presentada en la Imagen 15.



Leyenda

1	Probeta	h	espesor de la probeta
F	Fuerza aplicada	l	longitud de la probeta
R_1	Radio del elemento de carga	L	longitud de la distancia entre soportes
R_2	Radios de los soportes		

Imagen 15. Posición de la probeta al inicio del ensayo (ISO 178)

El punto en el que irá apoyada la cruceta deberá estar en el centro de la probeta y los soportes se situarán de forma equidistante a dicho centro. En este caso, se ha utilizado una probeta reducida, en comparación con la estándar, y denominada miniprobeta. Los valores de las variables, representadas en la Imagen 15, para este caso específico se presentan en la Tabla 2.

Tabla 2. Datos para preparación del ensayo a flexión

Datos posición mini-probeta teórica			Datos modelo mini-probeta	
			VOLCO	DECODE
R_1, radio de cruceta (mm)		5	5	
R_2, radio de soportes (mm)		5	5	
l, longitud probeta (mm)		40	40.6	40
b, anchura probeta (mm)		5	5.5	5
h, espesor probeta (mm)		2	1.8	1.95
L, distancia entre soportes (mm)		32	32	
Ubicación de puntos (X,Y,Z)				
Punto central (mm)	Altura de capa 0.3 mm	20, 5, 2	20.3, 5.5, 2	20, 5, 2.1
	Altura de capa 0.15 mm	20, 5, 2	20.3, 5.5, 1.8	20, 5, 1.95
Punto apoyo 1 (mm)		4, 5, 0	4.3, 5.5, 0	4, 5, 0
Punto apoyo 2 (mm)		36, 5, 0	36.3, 5.5, 0	36, 5, 0

La longitud, anchura y espesor de los modelos se pueden medir haciendo uso de la herramienta *Query* (*Tools>Query>Point/Node*). Una vez obtenidos estos valores, se pueden definir los puntos de apoyo.

Paso 2: Crear soportes y cruceta. En el módulo *Part*, utilizando el botón *Create Part* (Imagen 16), se crea una nueva pieza sólida por extrusión (rígida discreta) de 5 mm de radio y 5 mm de espesor.

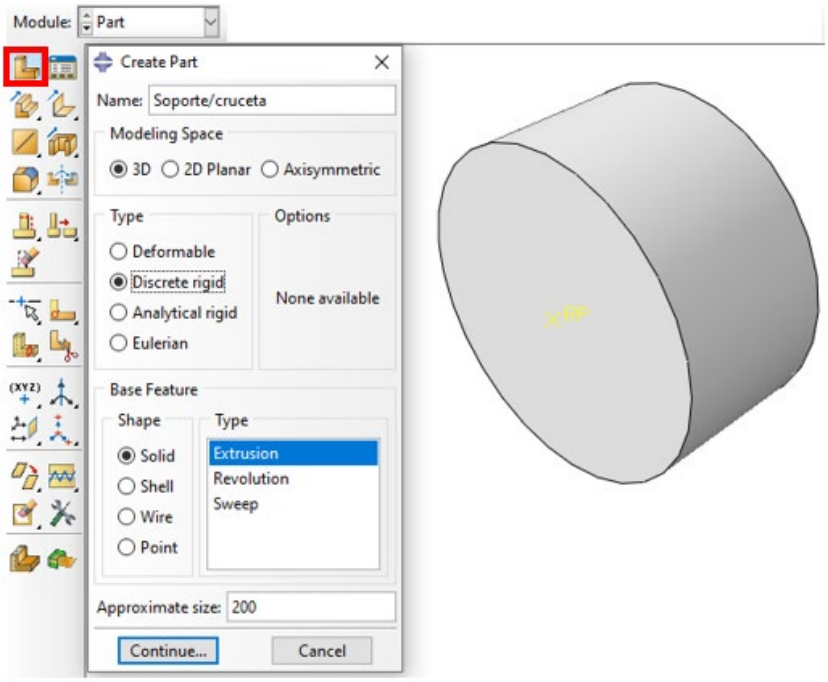


Imagen 16. Creación soportes/cruceta

Luego, a partir de este sólido, se crea una "cáscara" (*shell*) con *Create Shell:*

Form Solid (Imagen 17).

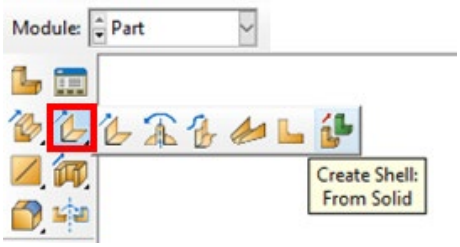


Imagen 17. Creación de malla a partir de sólido

Por último, se añade un punto de referencia en el centro del círculo de este sólido (*Tools>Reference Point…*). En la Imagen 16 se muestra el punto de referencia.

Paso 3: Atribuir las propiedades del material a la miniprobeta. En el módulo *Property*, se designarán las propiedades del material en el *Material Manager*. En este caso, se añadirán las propiedades del PLA mostradas en la Tabla 3.

Tabla 3. Propiedades del material (PLA)

Propiedades de PLA Smartfil		
General	Densidad	1.24 g/cm^3
Mecánico/Elástico	Módulo Young	3861 MPa
	Coeficiente de Poisson	0.35

A continuación, se creará una nueva sección asociada al material en el *Section Manager* y se asociará esta sección a la pieza en el *Section Assignment Manager*. Para esto último, habrá que seleccionar primero toda la pieza, asegurándose de no dejarse vóxeles deseleccionados en el caso de VOLCO.

Paso 4: Definir superficies de la base y la cara superior de la miniprobeta. Las superficies se crean en el apartado de ensamblaje (*Assembly*) dentro de la base de datos del modelo, en la parte izquierda de la ventana (Figura 56).

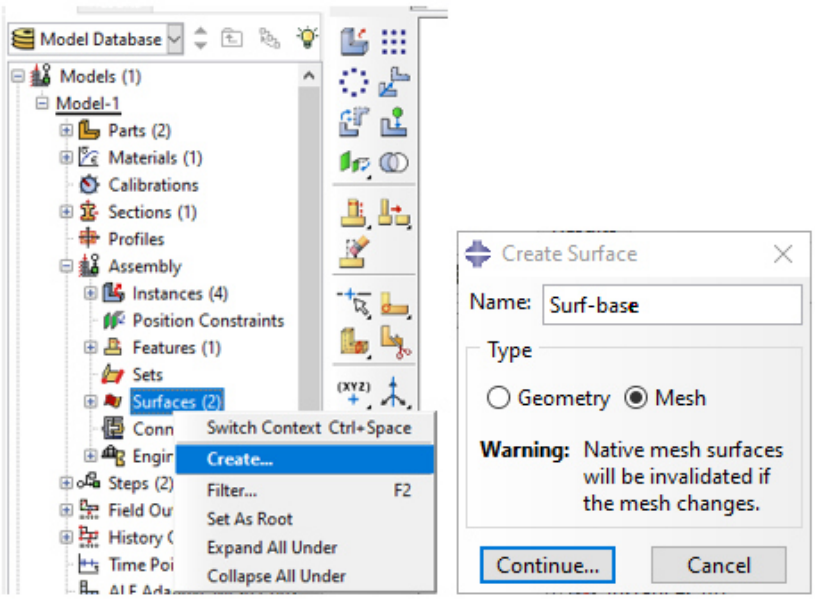


Imagen 18. Crear superficies

Al crear la nueva superficie, se selecciona que sea de tipo *Mesh* para definirla como un conjunto de nodos. En la Imagen 19 se muestran las superficies de la miniprobeta voxelizada y en la Imagen 20 de la modelada basada en geometría.

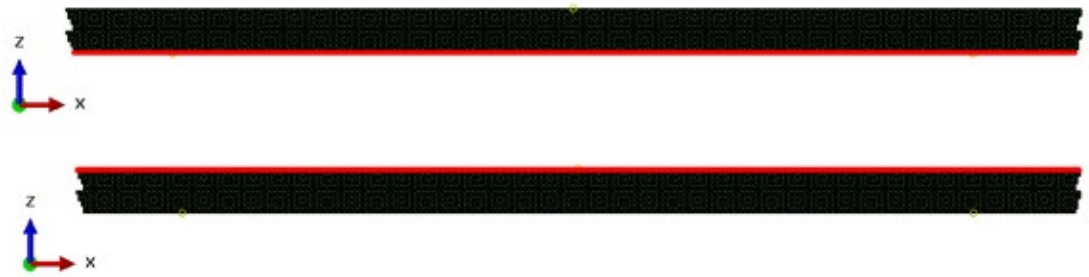


Imagen 19. Superficie de la base (arriba) y de la cara superior (abajo) de la miniprobeta VOLCO

Imagen 20. Superficie de la base (arriba) y de la cara superior (abajo) de la minoprobeta DECODE

Paso 5: Añadir soportes y cruceta al ensamblaje. Para ello, en el módulo *Assembly*, en *Create Instance*, se escoge el cilindro creado como soporte y se

sitúa en el punto de apoyo (definido previamente) mediante los botones *Translate Instance* y *Rotate Instance* (Imagen 21).

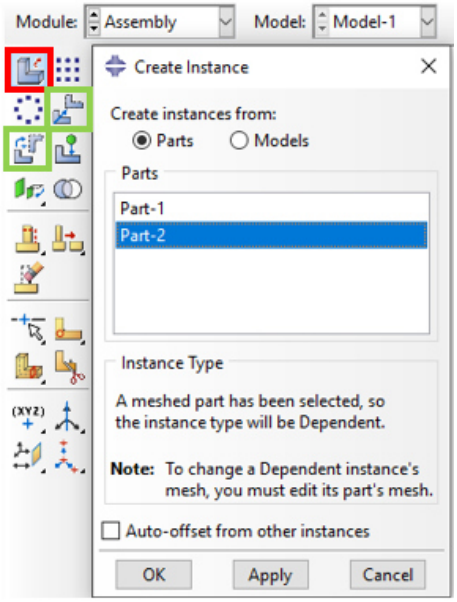


Imagen 21. Añadir y posicionar los soportes y la cruceta

Los ensamblajes quedarían como se muestra en la Imagen 22 (VOLCO) e Imagen 23 (DECODE).

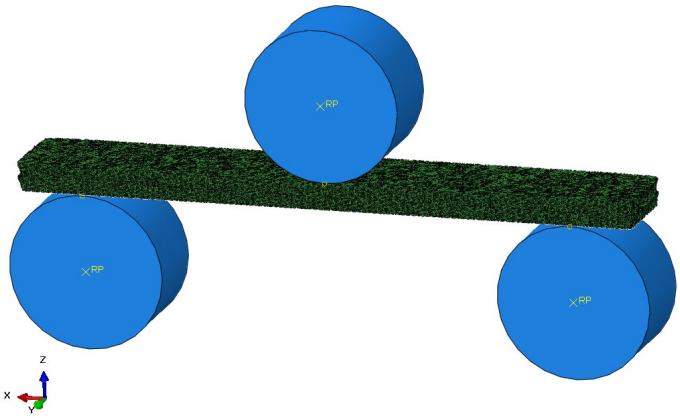


Imagen 22. Ensamblaje para ensayo miniprobeta VOLCO

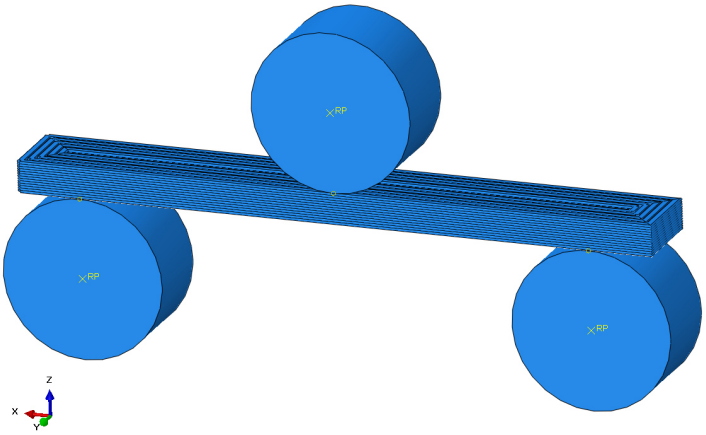


Imagen 23. Ensamblaje para ensayo miniprobeta DECODE

Paso 6: Crear *Step-1*. Desde el *Step manager*, crear un nuevo *Step* como *Static, General* y con la opción de geometría no linear activada (*NIgeom: On*). Las características de este nuevo paso (básicas, incrementos y otras), se muestran en la Imagen 25, Imagen 24 e Imagen 26, respectivamente.

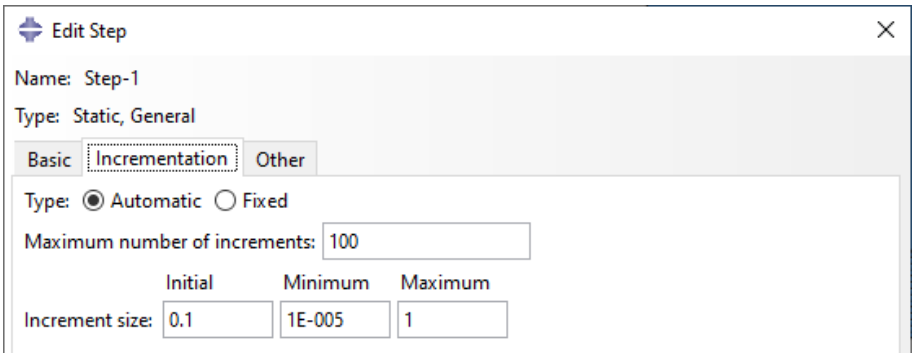


Imagen 24. Propiedades del Step-1: Incrementos

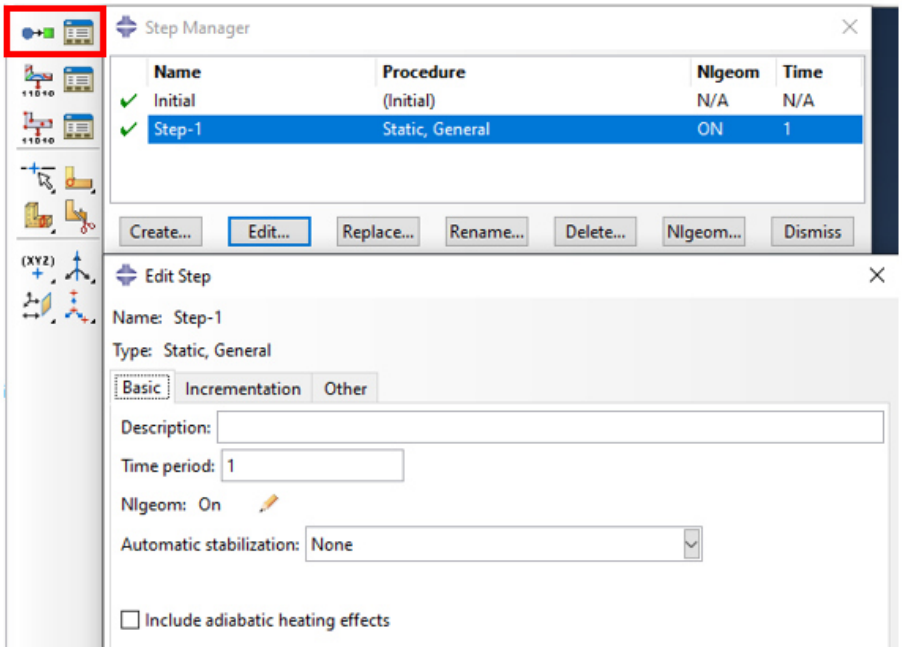


Imagen 25. Gestor de steps (Step Manager) y creación del Step-1

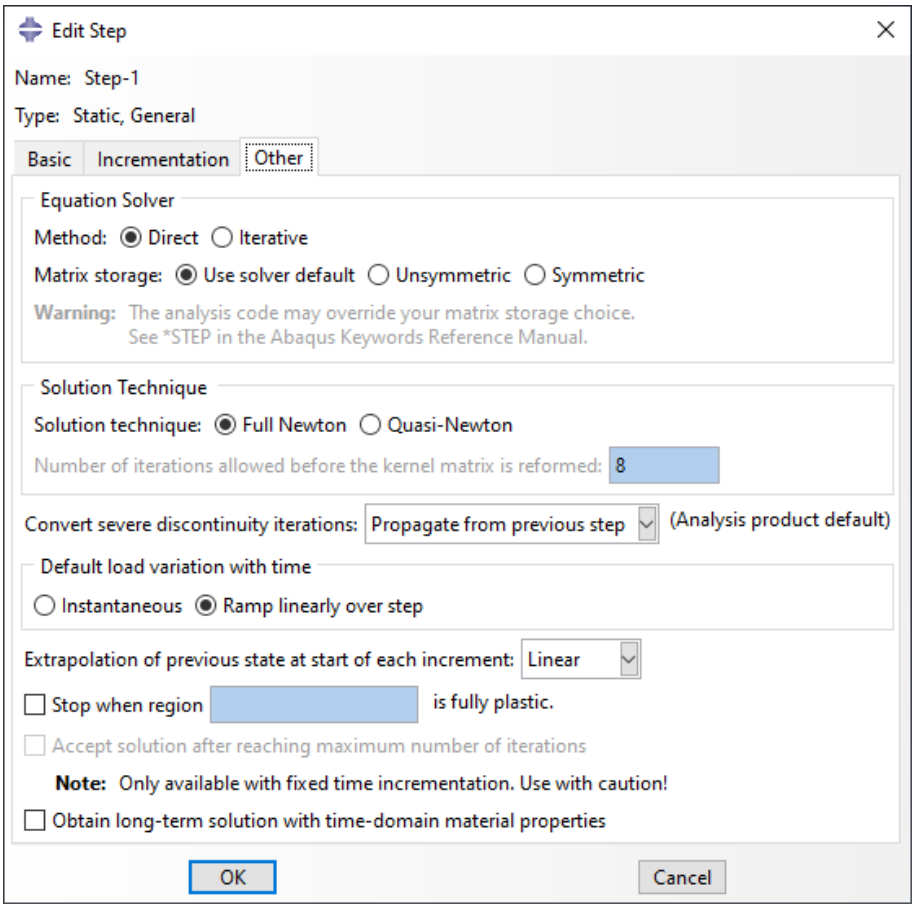


Imagen 26. Propiedades del Step-1: Otras

Paso 7: Definir las interacciones soporte con miniprobeta y cruceta con miniprobeta. En el módulo *Interaction*, en el botón *Interaction Property Manager*, se crea un nuevo contacto (Imagen 27).

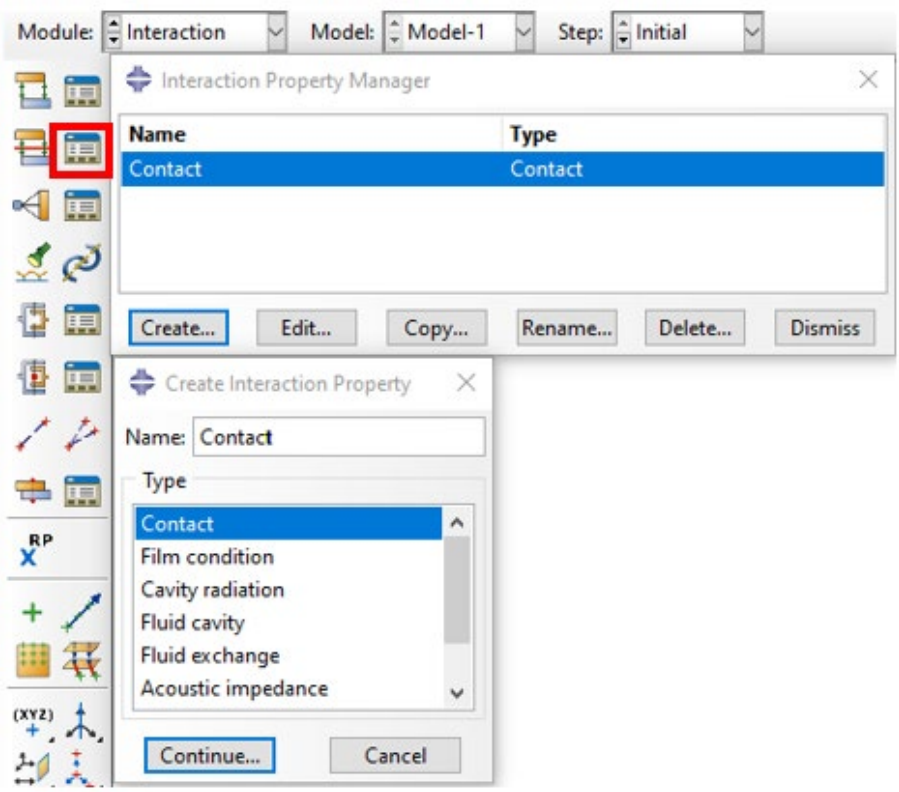


Imagen 27. Creación de nuevo contacto (interacción)

Las características del contacto se editarán para atribuirle un comportamiento normal como en la Imagen 28 y uno tangencial igual al mostrado en la Imagen 29.

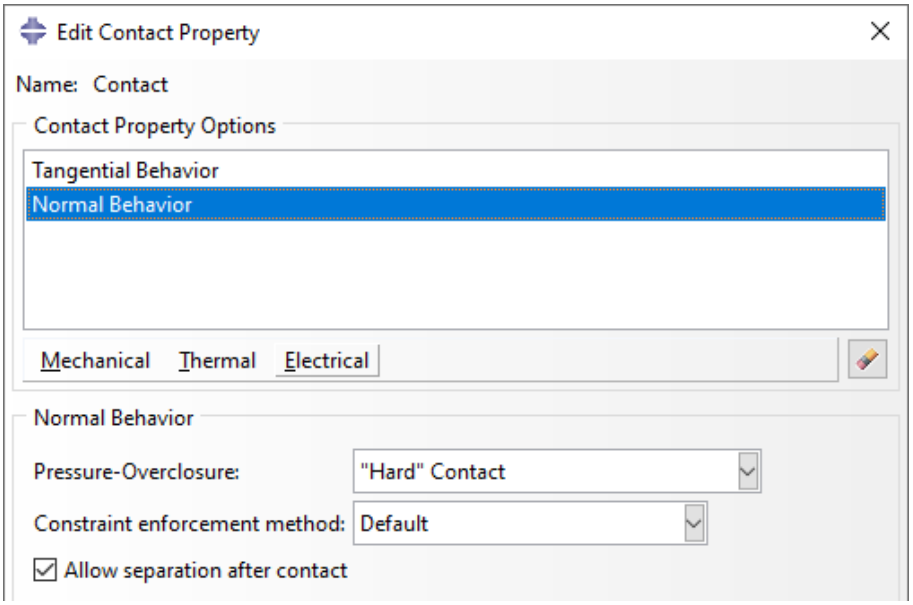


Imagen 28. Características del contacto: comportamiento normal

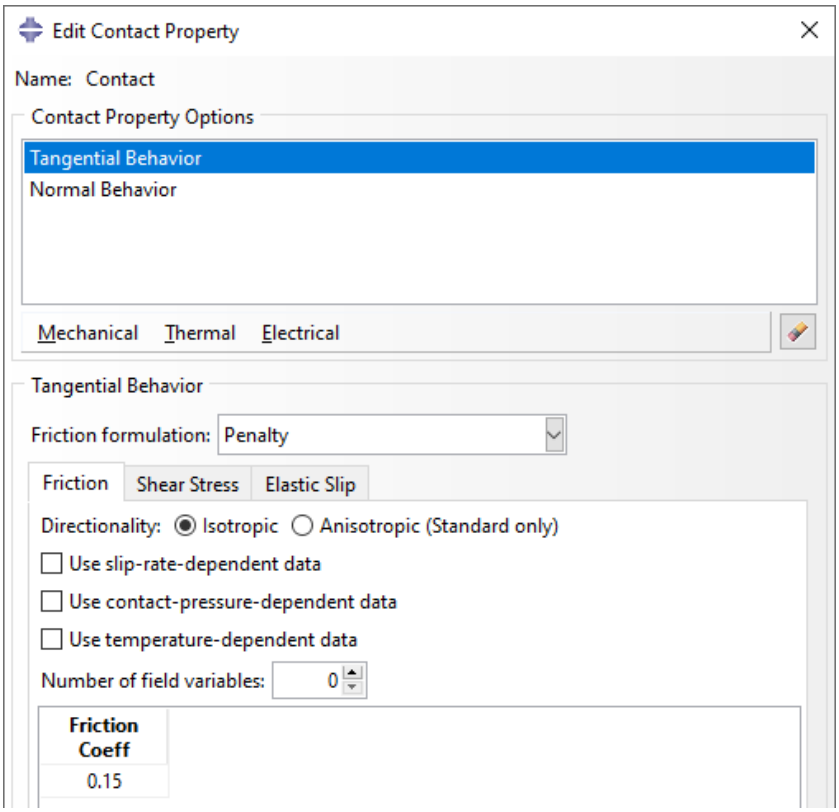


Imagen 29. Características del contacto: comportamiento tangencial

Después de definir en contacto, se atribuye sus propiedades a las diferentes interacciones entre superficies: soporte 1 – superficie base miniprobeta, soporte 2 – superficie base miniprobeta y cruceta – superficie superior de miniprobeta. Toda ellas son contacto superficie-superficie estándar (*Surface-to-surface contact*).

Paso 8: Condiciones de contorno. En el módulo *Load*, en el gestor de condiciones de contorno (*Boundary Condition Manager*), se crearán dos restricciones: una para los puntos de referencia de los soportes (encastrados en el inicio y en el *Step-1*) y otra para el punto de referencia de la cruceta (desplazamiento -1 mm en Z en el *Step-1*).

Paso 9: Mallado de soportes y cruceta. En el módulo *Mesh* se malla la pieza cilíndrica (no existen restricciones para definir estos parámetros).

Paso 10: Preparar y ejecutar la simulación. En el módulo *Job*, en el *Job Manager*, crear un nuevo *Job* y aumentar el número de procesadores para el cálculo, cuando sea posible.

Una vez se haya definido el *Job*, se ejecuta en *Submit*. Cuando el cálculo haya finalizado, el estado aparecerá como *Completed*. En caso, de haber algún fallo, aparecerá *Aborted* y se podrá conocer más información sobre la causa de este error en la tecla *Monitor.*

Paso 11: Obtener datos. En el módulo *Visualization*, en los datos de la sesión, se crea un nuevo *XY Data* del *ODB field output* (Imagen 30). De este, se selecciona la fuerza de reacción en Z dentro de *Unique nodal* (Imagen 31) y se escoge el punto de referencia de la cruceta (Imagen 32).

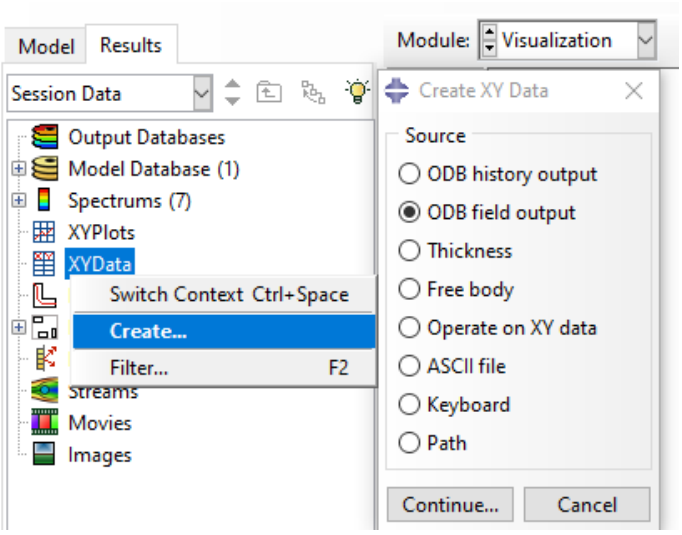


Imagen 30. Creación XY Data

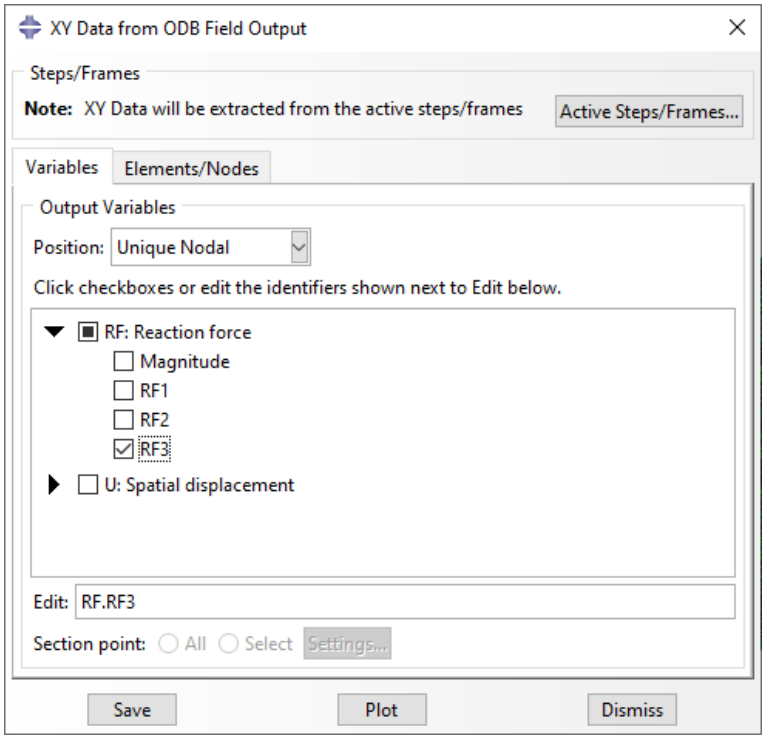


Imagen 31. XY Data: Variables

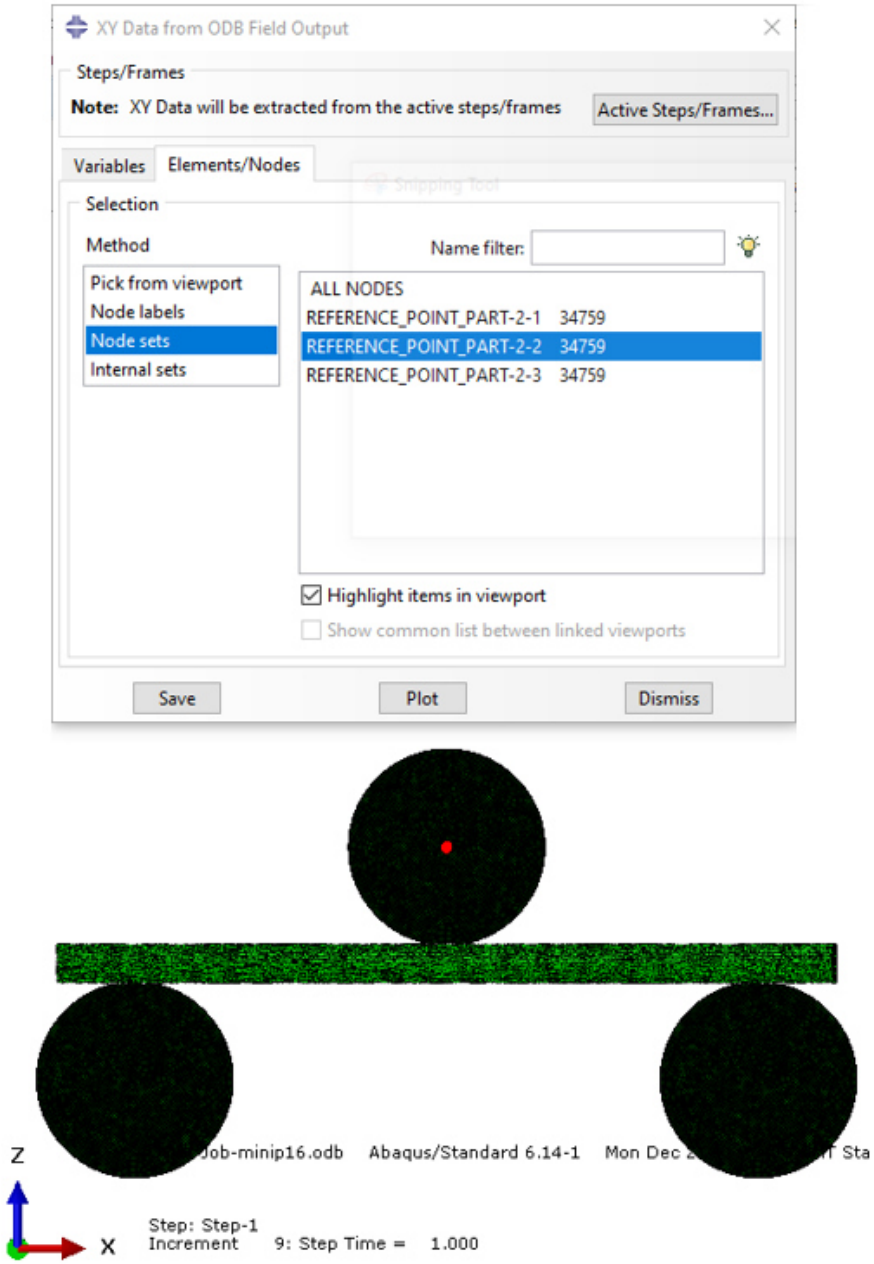


Imagen 32. Obtención fuerza de reacción mini-probeta

Paso 12: Calcular fuerza de reacción total haciendo la sumatoria de las fuerzas en los nodos. En el módulo *Visualization*, en *XY Data*, se selecciona *Operate on XY data* (Imagen 33).

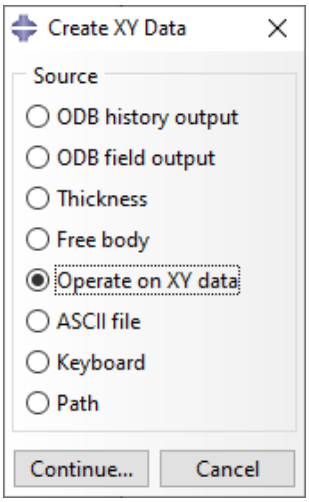


Imagen 33. Operate on XY data

Para la sumatoria se seleccionará la operación *sum* y entre paréntesis se añadirán todos los datos, haciendo uso del *Add to Expression* (Imagen 34). Por último, el resultado se guarda como fuerza de reacción.

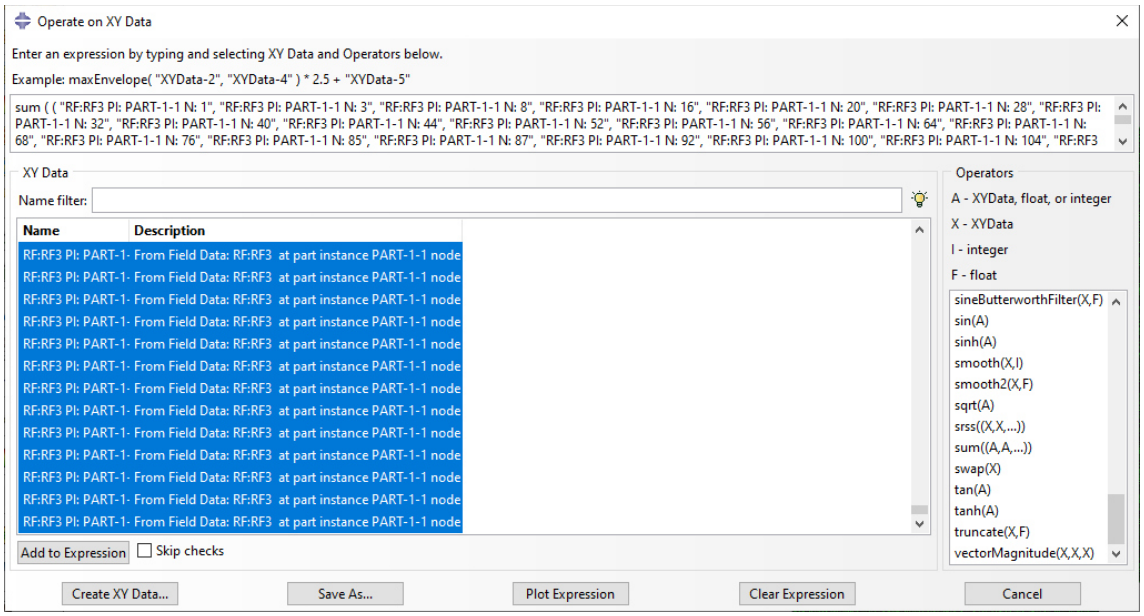


Imagen 34. Sumatoria de datos

Paso 13: Determinar el módulo elástico a partir de la fuerza de reacción resultante en el *Step-1* o, lo que es lo mismo, cuando la cruceta se ha desplazado hasta su valor máximo (-1 mm). Para este cálculo se aplicará la ecuación del módulo de elasticidad a flexión de la norma ISO 178 (Plásticos – Determinación de las propiedades de flexión). En este caso, el área equivalente corresponderá al área de la base si fuera la pieza fuera sólida.

ANEXO H. APLICACIÓN DE METODOLOGÍA DE MODELADO A SCAFFOLD SOMETIDO A COMPRESIÓN PASO A PASO

En este anexo, se presenta el paso a paso para el modelado de un scaffold, tanto con la metodología DECODE (basada en geometría) como con VOLCO (basada en vóxeles). Se describe el proceso desde la generación de coordenadas con el archivo de código G, hasta la interpretación de resultados del análisis de elementos finitos (AEF) en Abaqus/CAE 6.14-1. En este caso, se ha aplicado un ensayo a compresión.

Tabla de contenidos

i. Modelado basado en geometría

Paso 1: Ejecutar *Abaqus_model*, indicando en nombre del archivo G-code del que extraerá las coordenadas y el archivo Python que generará.

Paso 2: Abrir Abaqus/CAE 6.14-1 y ejecutar Python, mediante *Run Script*.

Paso 3: Definir mallado del modelo. El mallado se realizará en el módulo *Mesh*. En primer lugar, se definirá el tamaño de la malla en *Seed Part* (Ilustración 1).

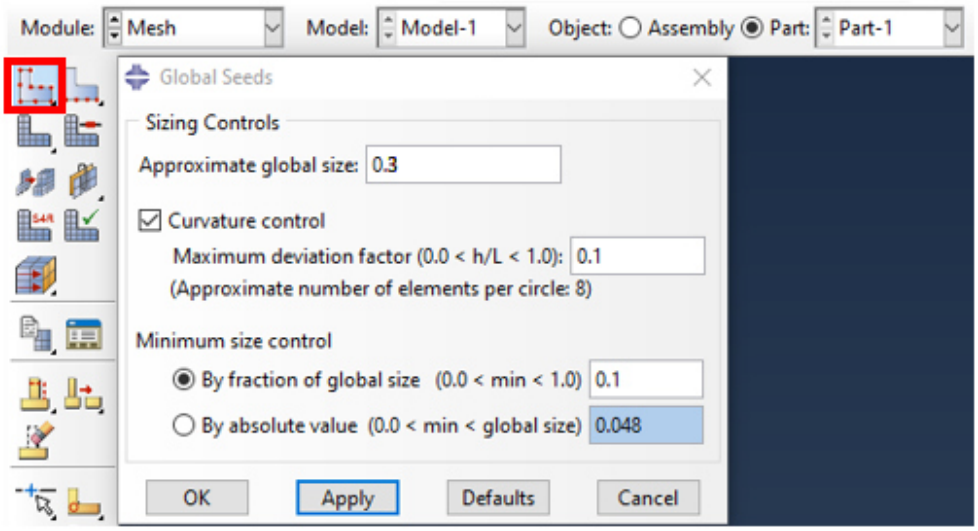


Ilustración 1. Definición de tamaño de malla

A continuación, se definirá el tipo de malla, en este caso, tetraédrica, en *Assign Mesh Controls* (Ilustración 2).

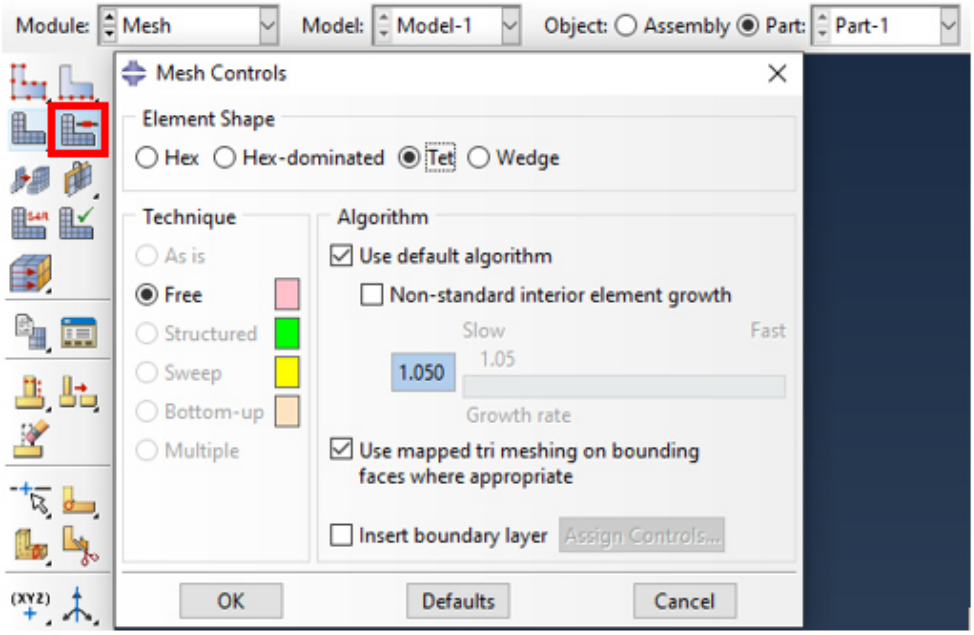


Ilustración 2. Asignación forma de malla

En *Assign Element Type* (Ilustración 3) se puede seleccionar el orden geométrico: linear (C3D4) o cuadrático (C3D10).

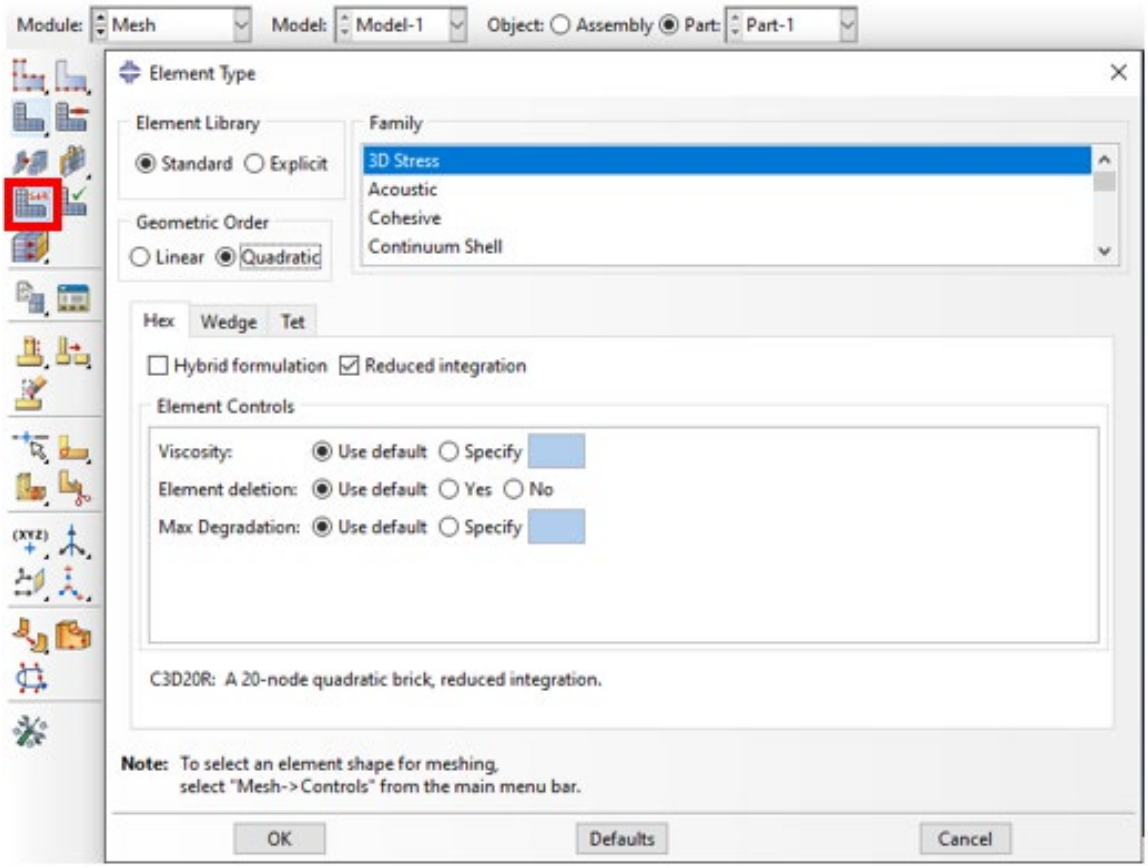


Ilustración 3. Asignar tipo de malla (orden geométrico)

Paso 4: (Opcional) Si existen errores al mallar, una posible solución es cambiar el tamaño de semilla. Si sigue sin funcionar, otra opción es aplicar *Virtual Topology* (Ilustración 4).

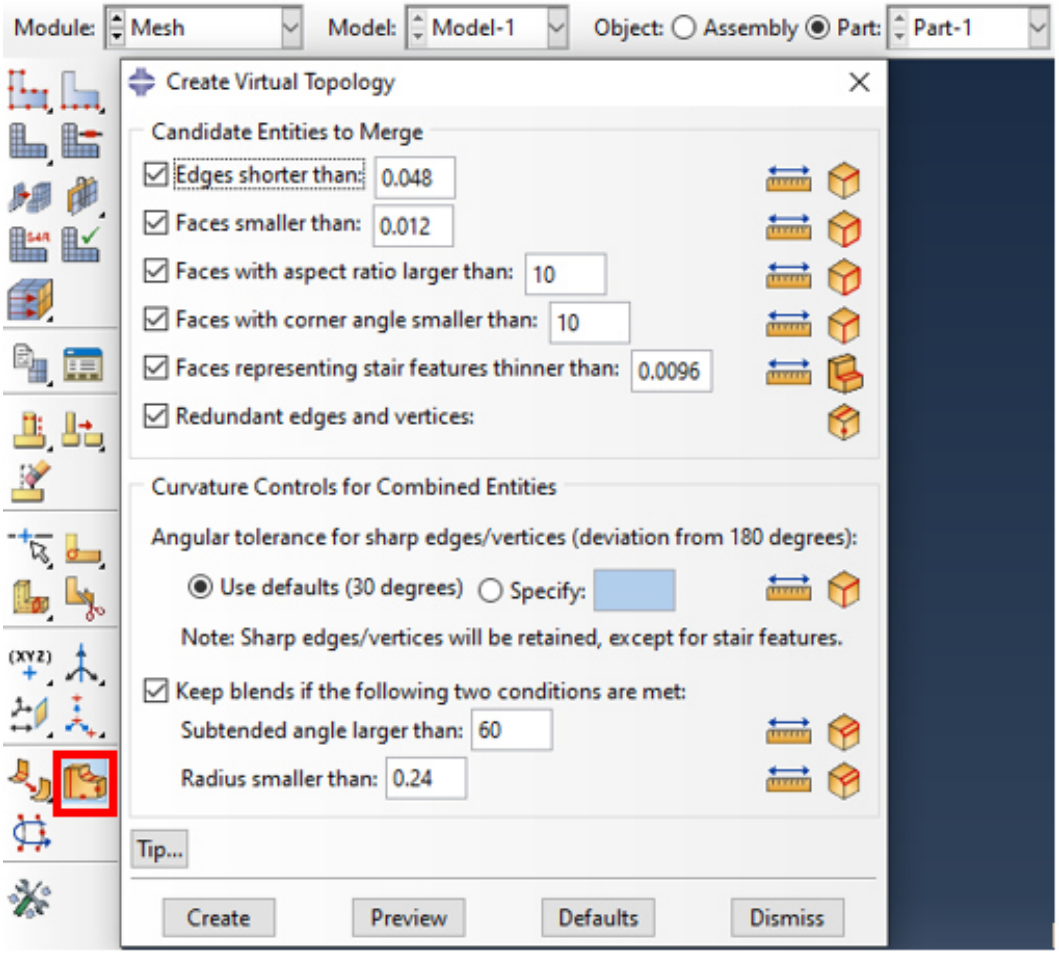


Ilustración 4. Aplicación de Virtual Topology

Paso 5: Mallar modelo. En el módulo *Mesh*, clicando en el botón *Mesh Part* (Ilustración 5), se malla la pieza.

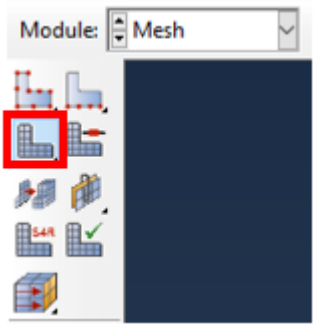


Ilustración 5. Botón "Mesh Part" para el mallado

Paso 6: En el módulo Assembly, se introduce la pieza como parte única del ensamblaje, en “Create Instance” (Ilustración 6).

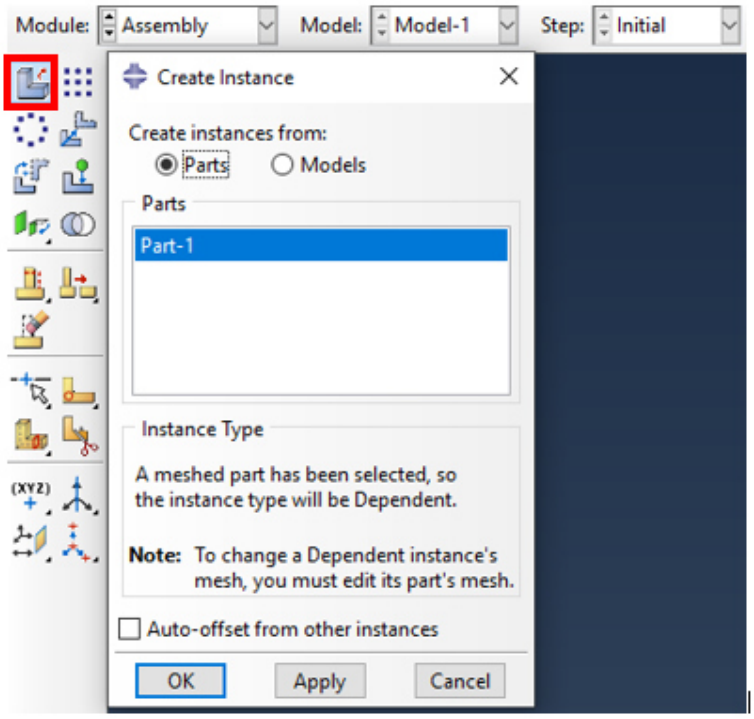


Ilustración 6. Introducir pieza en el ensamblaje

El modelo se ve representado en la Ilustración 7.

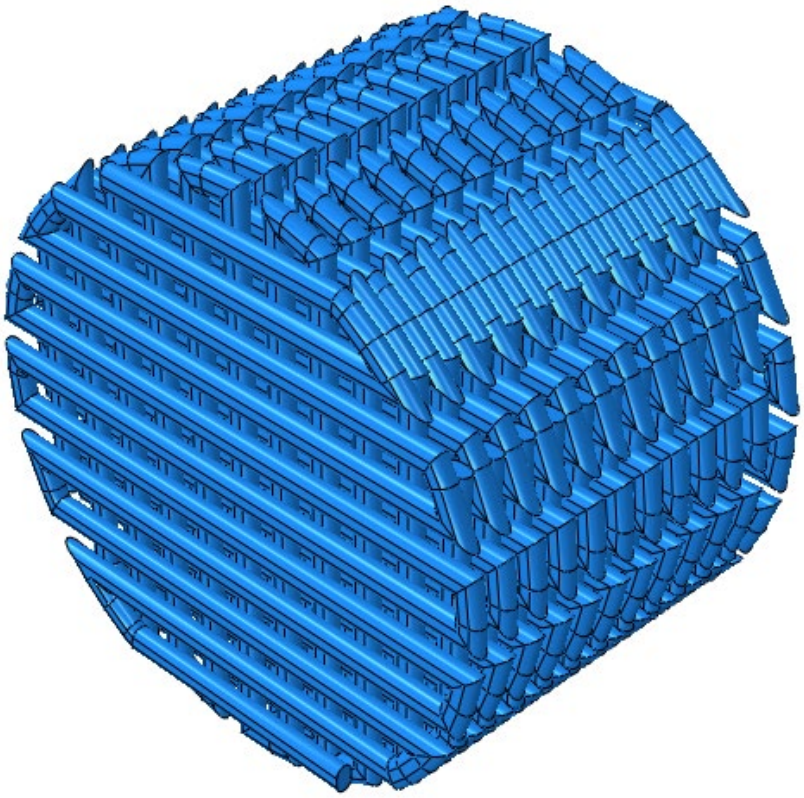


Ilustración 7. Modelo basado en geometría del scaffold

Paso 7: Calcular volumen y dimensiones. Esto se puede realizar desde *Tools* > *Query…* > *Point / Distance / Mass properties* (Ilustración 8).

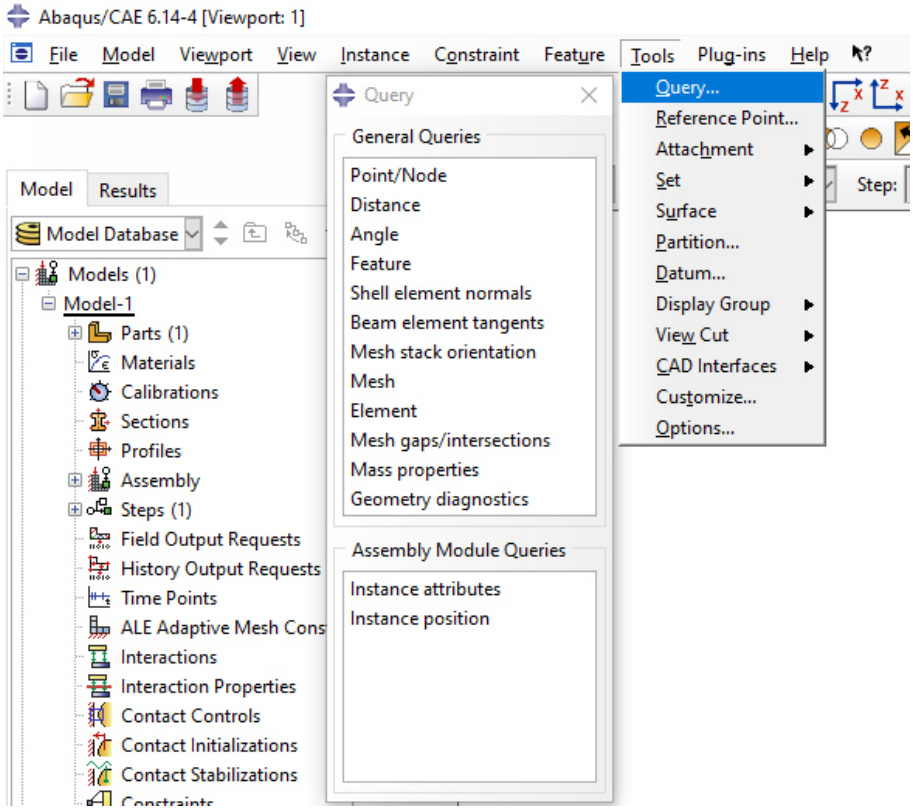


Ilustración 8. Tomar datos puntos, dimensiones o masa

ii. Modelado basado en vóxeles

Paso 1: Generar coordenadas a partir del archivo G-code en la subrutina correspondiente, dependiendo del laminador utilizado (*Coord_Gen_Slic3r*, *Coord_Gen_Simplify* o *Coord_Gen_FullControl*).

Paso 2: Introducir coordenadas en la pestaña *Toolpath* del *Setup.xlsm* (VOLCO).

Paso 3: Cambiar datos en *Parameters for the voxel simulation* (Ilustración 9) y *Parameters for the STL output from MATLAB and to set up the model in ABAQUS* (Ilustración 10).

Parameters for the voxel simulation				
Name	Value	Units	SI value	SI units
VoxelSize	50	µm	0.00005	m
SizeX	16	mm	0.016	m
SizeY	16	mm	0.016	m
SizeZ	12	mm	0.012	m
VoxelsX	320		320	
VoxelsY	320		320	
VoxelsZ	240		240	
OriginalSphereRadius	200	µm	0.0002	m
SphereCentreBelowNozzle	200	µm	0.0002	m
GlueLayerThickness	0	µm	0	m
SphereStepRadiusIncreaseFraction	0.05		0.05	
MaxSphereIncreaseSteps	3000		3000	
nominalStep	102	µm	0.0001	m
Toolpath offset X	2000	µm	0.002	m
Toolpath offset Y	2000	µm	0.002	m
Toolpath offset Z	-200	µm	-0.0002	m
MaxDepositionRadius	10000	µm	0.01	m

Ilustración 9. Parámetros para la simulación del vóxel (scaffold)

Parameters for the STL output from MATLAB and to set up the model in ABAQUS				
Name	Value	Units	SI value	SI units
Xstart	0	mm	0	m
Xend	12	mm	0.012	m
Ystart	0	mm	0	m
Yend	12	mm	0.012	m
Zstart	0.2	mm	0.0002	m
Zend	9.2	mm	0.0092	m
Layer height	0.3	mm	0.0003	m
Start height	0.35	mm	0.00035	m
STL model name			VOLCO_TEMP	
Job name			Job_VOLCO_TEMP	
Young's modulus	2.03E+09	Pa	2.03E+09	Pa
Poisson's ratio	0.36		0.36	
BOUNDARY CONDITIONS		DIRECTION		
NODES	TYPE	X	Y	Z
LOW_X	FIX	0	UNSET	UNSET
HIGH_X	PLANAR	ON	OFF	OFF
LOW_Y	FIX	UNSET	0	UNSET
HIGH_Y	PLANAR	OFF	ON	OFF
LOW_Z	FIX	UNSET	UNSET	0
HIGH_Z	DISP	UNSET	UNSET	-0.18

Ilustración 10. Parámetros para el STL de salida de Matlab (scaffold)

Paso 4: Ejecutar *OUTPUT SETUP FILE* y *OUTPUT TOOLPATH* (Ilustración 11).

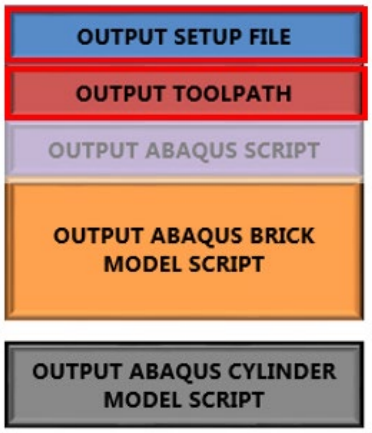


Ilustración 11. Botonera para generar archivos de salida

Paso 5: Ejecutar *Voxel_Code_v8_IMPORT_GCODE.m* en Matlab.

Paso 6: Anotar datos que devuelve VOLCO al terminar de generar el STL (Ilustración 12).

```
Elapsed time is 187.949815 seconds.
if sizeX / VoxelSize is not an integer then this is not ideal... the
region may not be a unit cell.
ans =
 0.011950000000000
Maxheight =
 6.950000000000000
SurfacePorosity =
 0.503003472222222
Volume =
   2.795713750000001e-07
ForCopying =
  0.503003472222222   0.000000279571375
```

Ilustración 12. Datos de salida recogidos después de la generación del STL

Paso 7: Comparar el volumen y porosidad obtenidos con el teórico (Tabla 1). Cambiar el radio de la esfera (*OriginalSphereRadius*) en los parámetros de simulación de vóxeles y, con los resultados, buscar el radio que genera una geometría con un volumen igual al teórico (usando el comando *spline(x,y,xq)* en Matlab).

Tabla 1. Comparación volumen y porosidad teóricos y del modelo

	Teórico	Primer modelo	Modelo tras aplicar spline
Volumen (mm^3)	278.8	279.571	278.797
Porosidad (%)	50	50.3	50.34
Radio de esfera (µm)	-	200	199.7232

Paso 8: Importar la pieza en Abaqus: *Plug-ins>Tools>STL import*. El modelo aparecerá representado mediante vóxeles (Ilustración 13).

Ilustración 13. Vista modelo scaffold en Abaqus

Paso 9: Para pasar de triángulos a tetraedros y obtener una malla C3D4 (elementos tetraédricos de cuatro nodos), se debe editar la malla en el módulo *Mesh*, *Mesh>Edit…>Mesh>Convert tri to tet* (Ilustración 14).

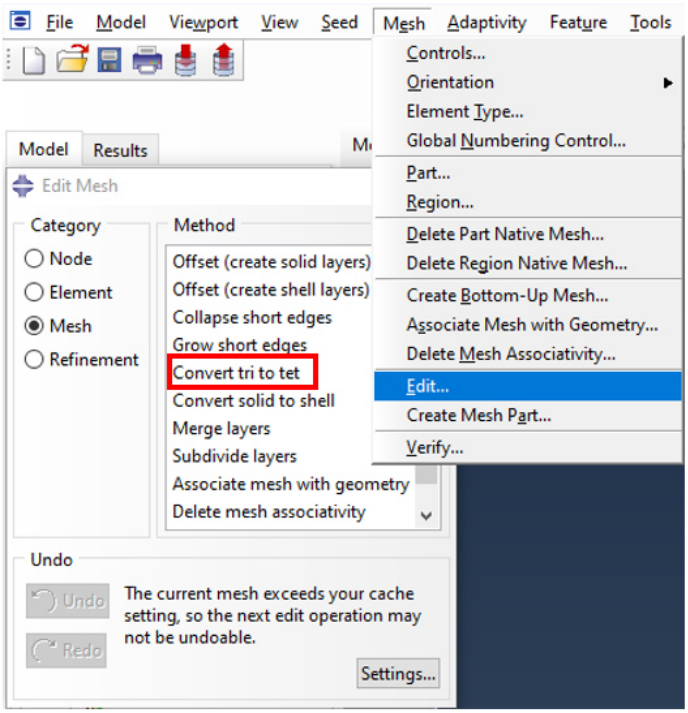


Ilustración 14. Obtención de malla C3D4

iii. Análisis de elementos finitos

Paso 1: Definir propiedades. En el módulo *Property*, se designarán las propiedades del material en el *Material Manager* (Ilustración 15).

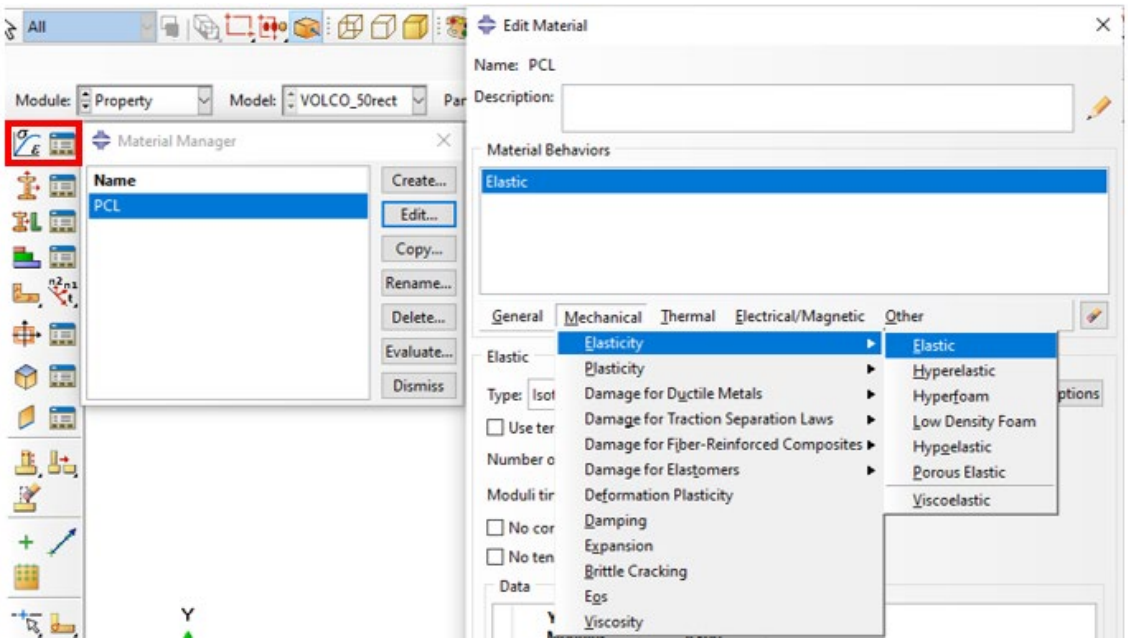


Ilustración 15. Gestor de materiales (Material manager)

Las propiedades del material utilizado en este caso, PCL, se muestran en la Ilustración 16.

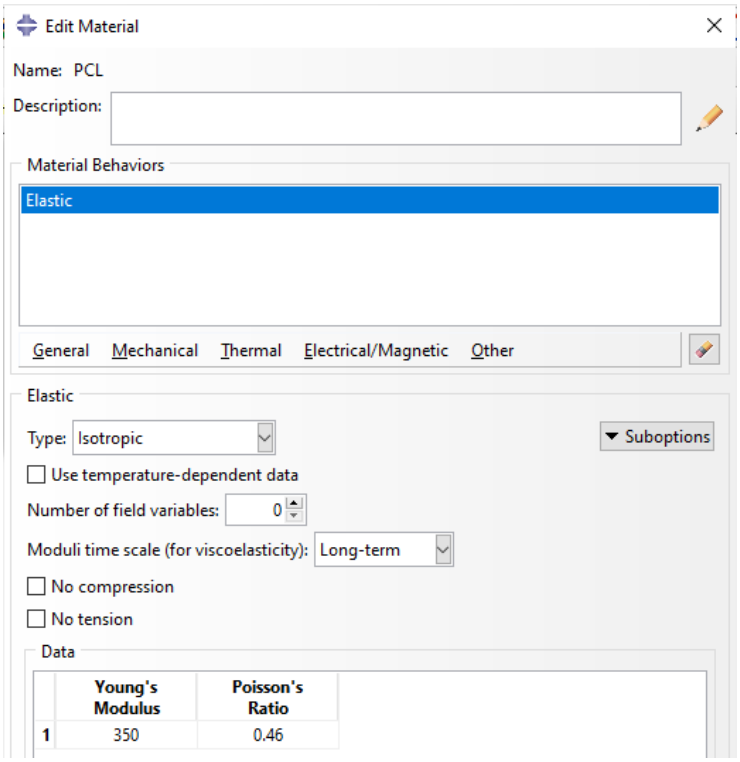


Ilustración 16. Propiedades del PCL

A continuación, se creará una nueva sección asociada al material en el *Section Manager* (Ilustración 17) y se asociará esta sección a la pieza en el *Section Assignment Manager* (Ilustración 18). Para esto último, habrá que seleccionar primero toda la pieza, asegurándose de no dejarse vóxeles deseleccionados, en el caso de trabajar con el modelo VOLCO.

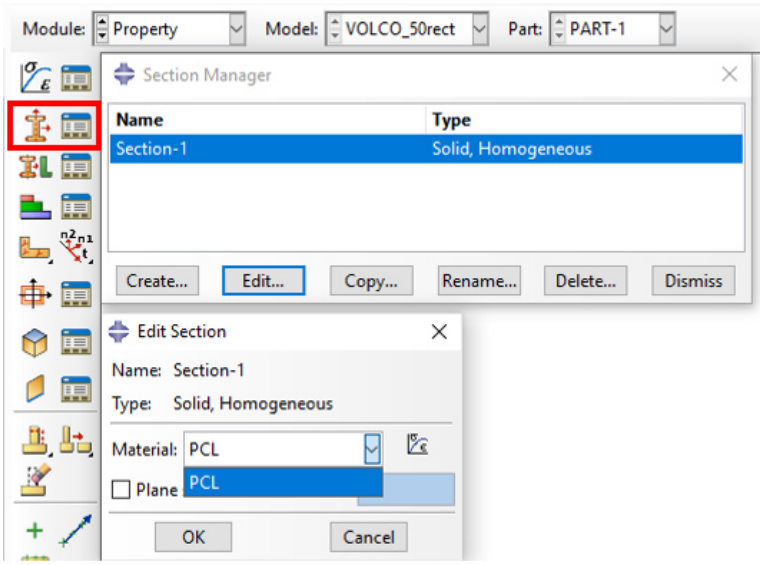


Ilustración 17. Gestor de secciones (Section manager)

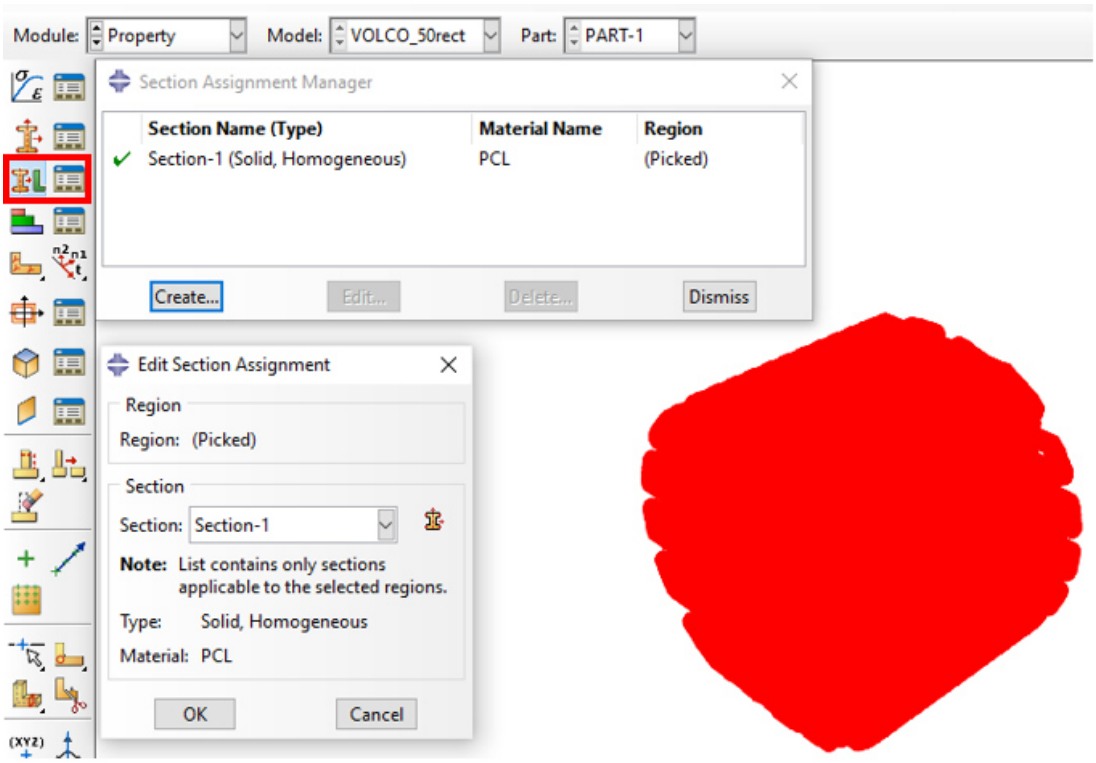


Ilustración 18. Gestor de asignación de sección (Section assignment manager)

Paso 2: Crear *Sets*: base, cara superior, punto 1 y punto 2. Los sets se crean en el apartado de ensamblaje (*Assembly*) dentro de la base de datos del modelo, en la parte izquierda de la ventana. Los nuevos sets, se definirán por nodos (Ilustración 19).

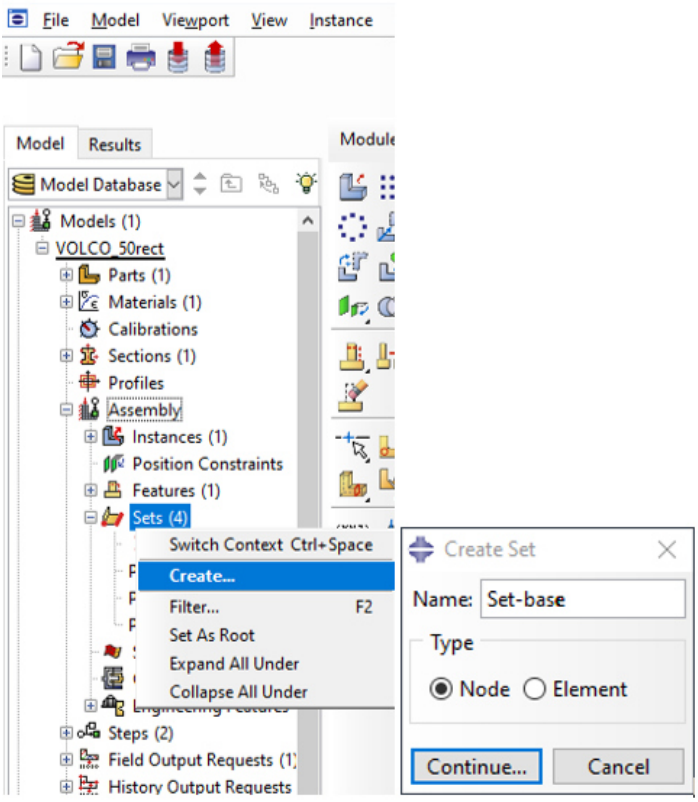


Ilustración 19. Creación sets

Se crearán tres sets: uno estará formado por todos los nodos de la cara base, otro por todos los nodos de la cara superior y otro por dos nodos centrales de la cara base.

La representación de los sets del modelo voxelizado se encuentra en la Ilustración 20. Mientras que, en la Ilustración 21, se encuentran las del modelo basado en geometría.

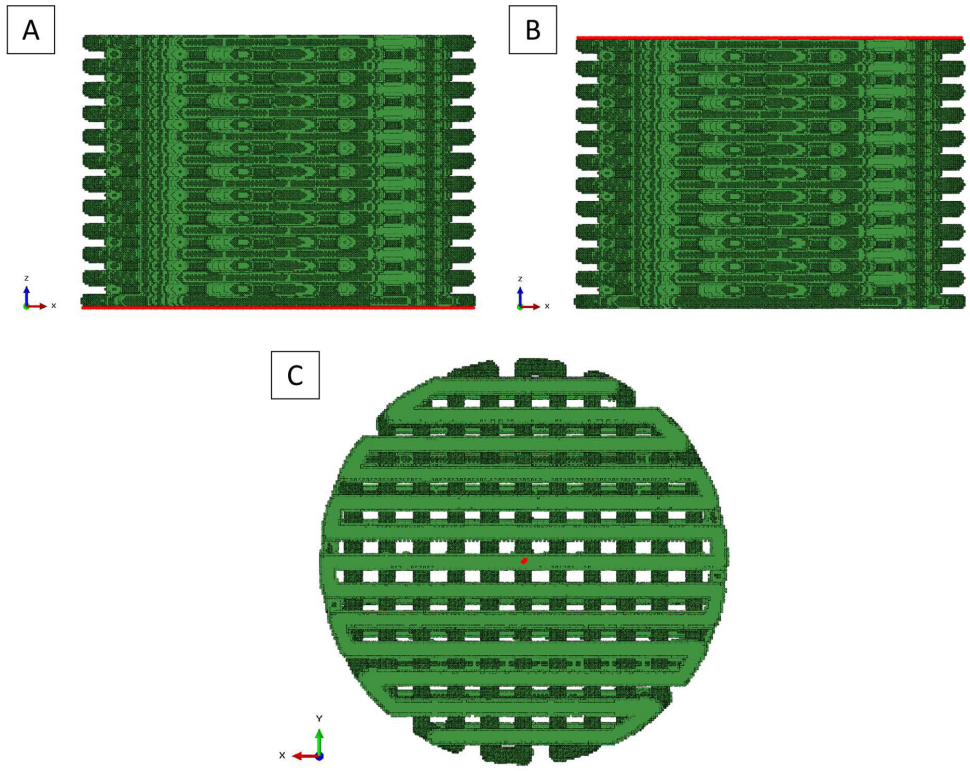


Ilustración 20. Sets modelo voxelizado: (A) set de la base, (B) set de lacara superior (top) y (C) set de dos puntos en la base

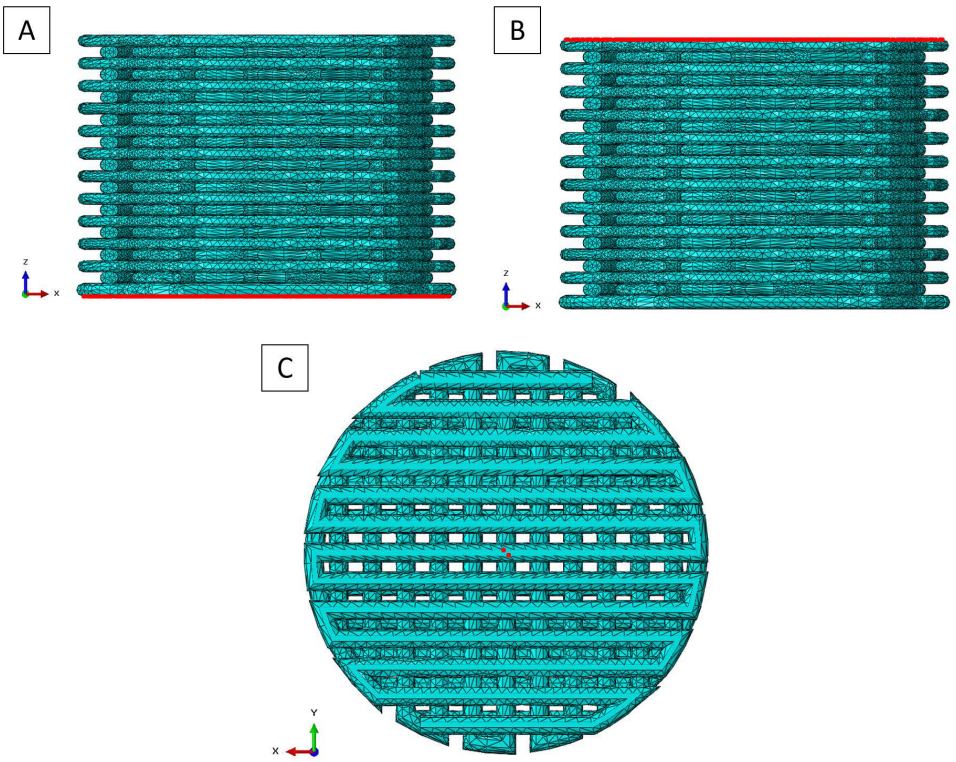


Ilustración 21. Sets modelo basado en geometría: (A) set de la base, (B) set de lacara superior (top) y (C) set de dos puntos en la base

Paso 3: Crear Step-1. Desde el *Step manager*, crear un nuevo *Step* como *Static, General* y con la opción de geometría no linear activada (*NIgeom: On*). Las características de este nuevo paso (básicas, incrementos y otras), se muestran en la Ilustración 22, Ilustración 23 e Ilustración 24, respectivamente.

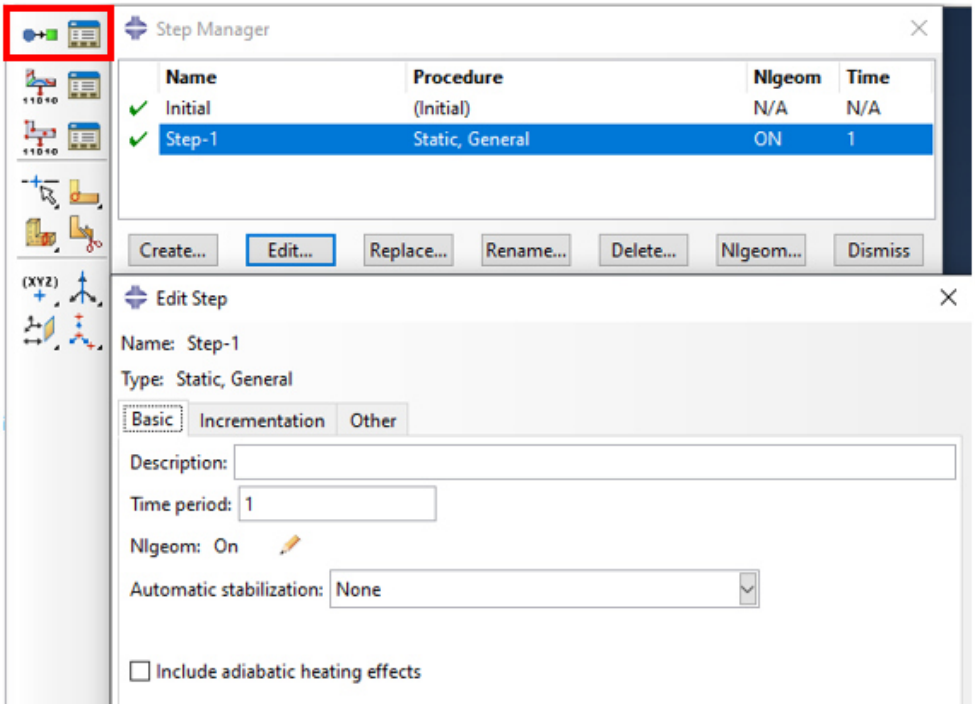


Ilustración 22. Gestor de steps (Step Manager) y creación del Step-1

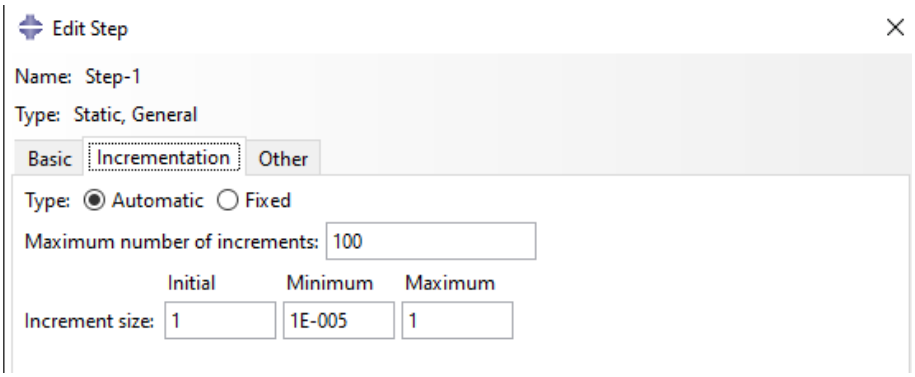


Ilustración 23. Propiedades del Step-1: Incrementos

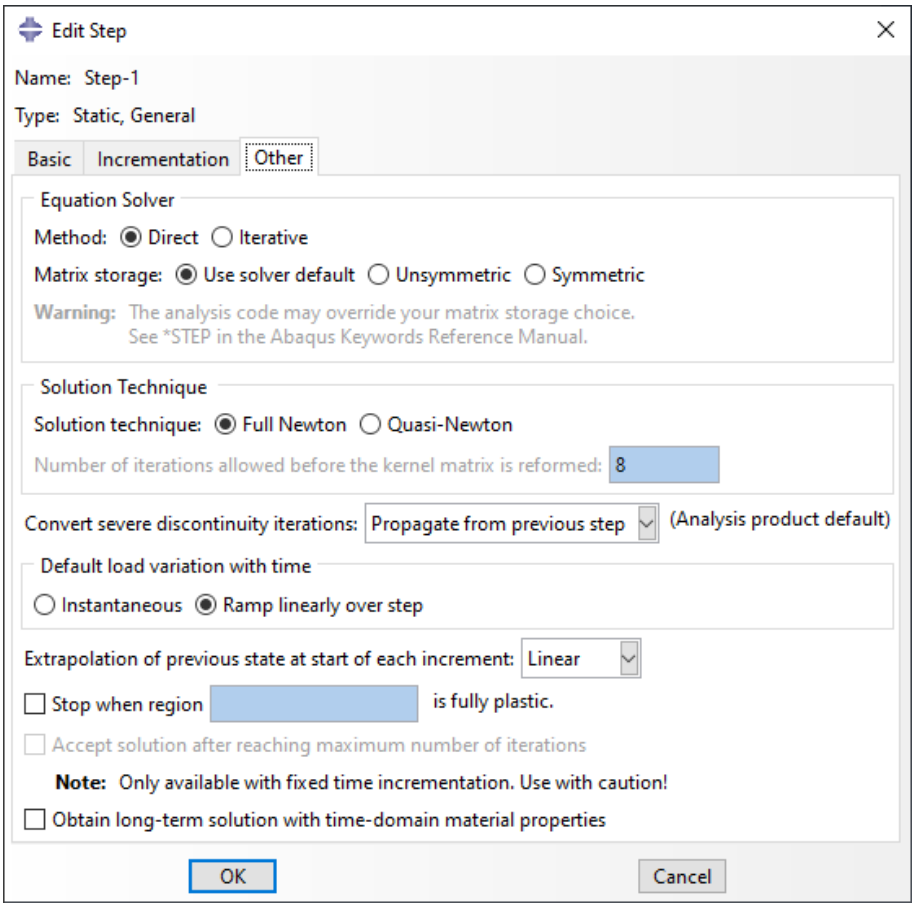


Ilustración 24. Propiedades del Step-1: Otras

Paso 4: (Opcional) En caso de necesitar reducir el tiempo de simulación, se podría simplificar el cálculo en el módulo *Step*, en *Field Output*, seleccionando sólo en el set de la base (*Domain*), el último incremento (*Frequency*) y las variables RF (*Forces/Reactions*) y U (*Displacement/Velocity/Acceleration*). Este paso se muestra en la Ilustración 25.

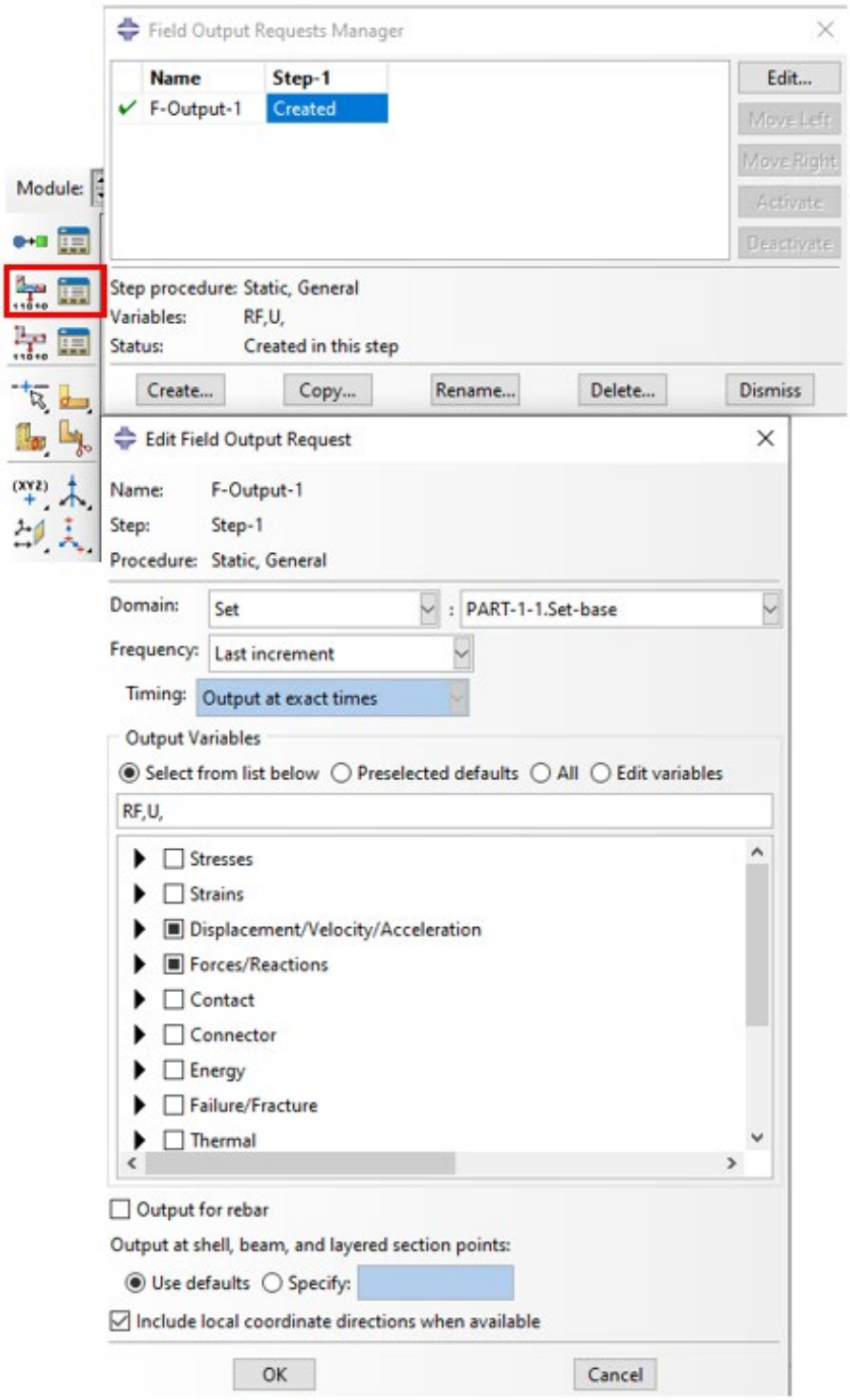


Ilustración 25. Field Output Manager

Paso 5: Condiciones de contorno. En el módulo *Load*, en el gestor de condiciones de contorno (*Boundary Condition Manager*) se crearán tres restricciones: una para los dos puntos de la base (encastrados en el inicio y en el *Step-1*), otra para la base (desplazamiento 0 mm en Z en el inicio y en el *Step-1*) y otro para la cara superior (desplazamiento -0.2 mm en Z en el *Step-1*). El procedimiento de creación de condiciones de contorno y las características de

cada una de ellas se encuentran en la Ilustración 26 y la Ilustración 27.

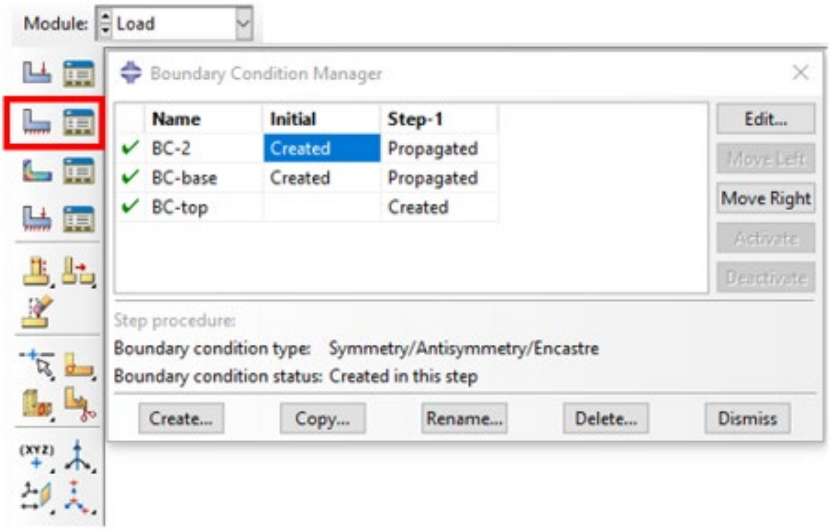


Ilustración 26. Boundary condition manager

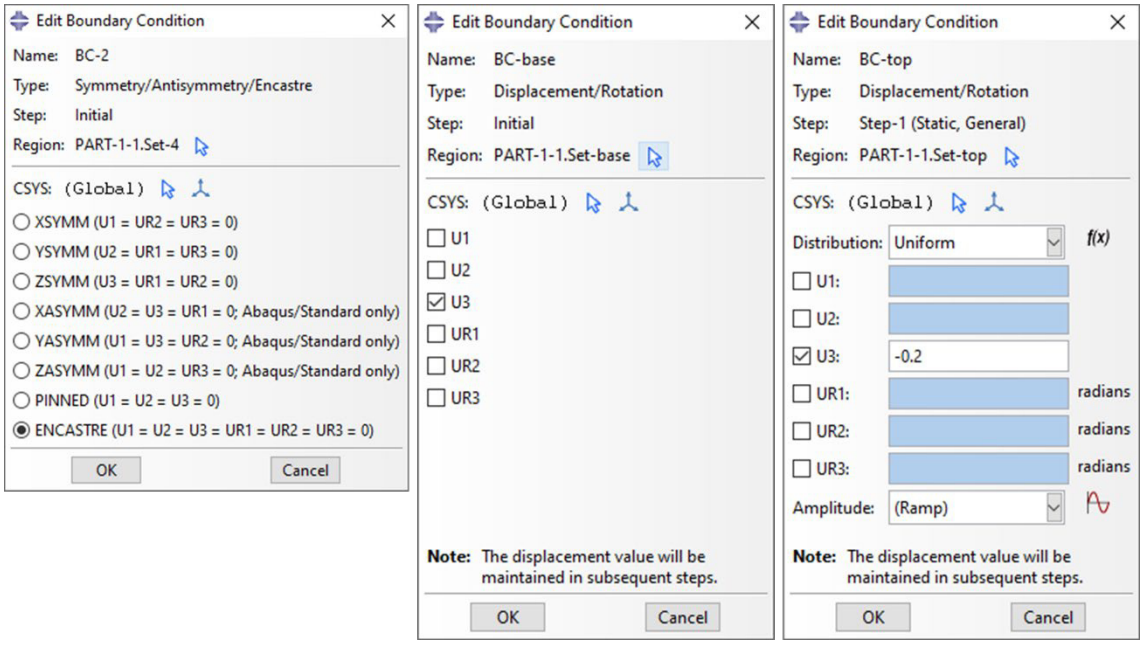


Ilustración 27. Encastre en puntos de la base (izquierda), condiciones de contorno de la base (centro) y condiciones de contorno de la cara superior (derecha)

Paso 6: Preparar y ejecutar el *Job*. En el módulo *Job*, en el *Job Manager*, crear un nuevo *Job* (Ilustración 28) y aumentar el número de procesadores para el cálculo, cuando sea posible (Ilustración 29).

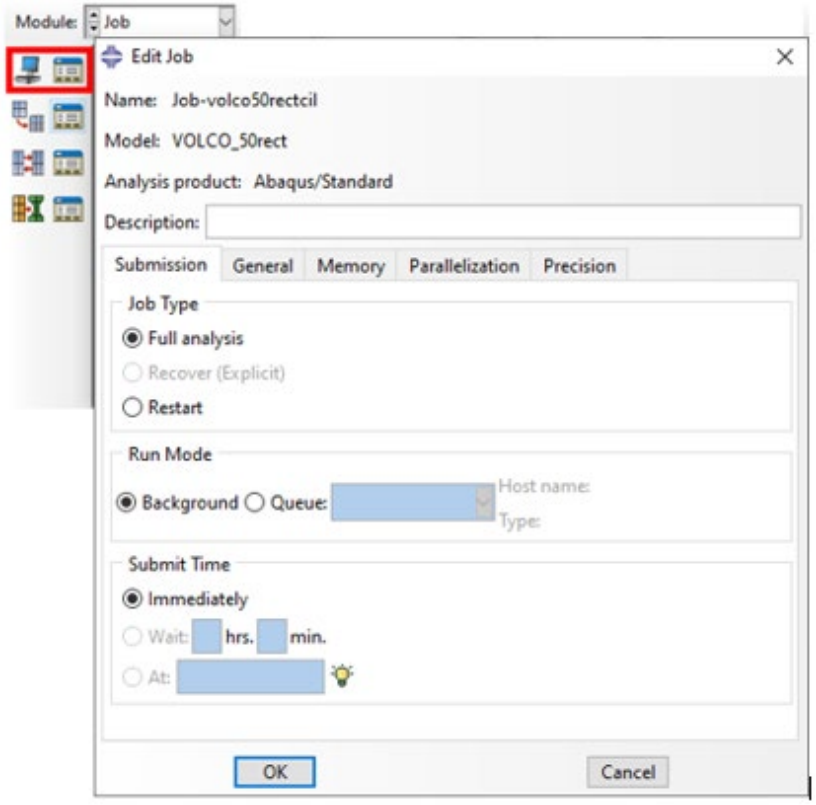


Ilustración 28. Creación Job

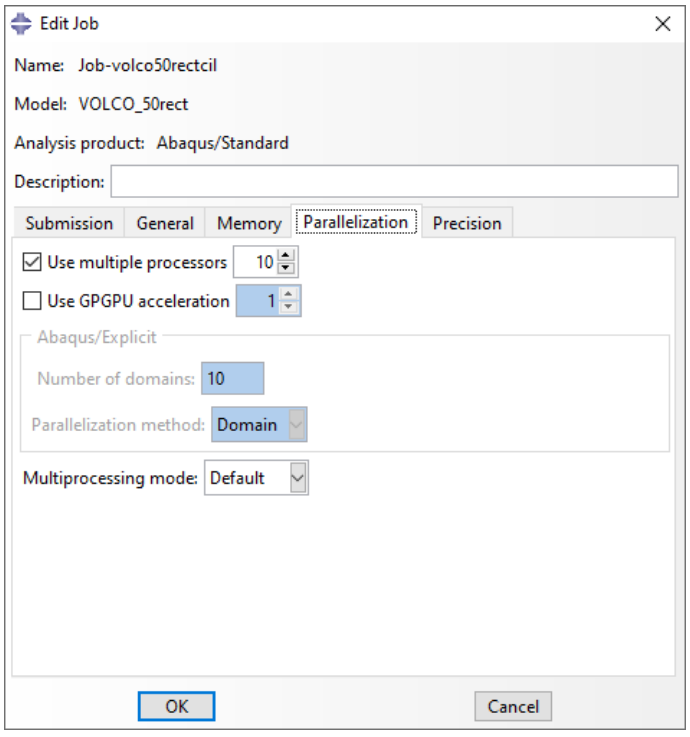


Ilustración 29. Uso de múltiples procesadores para la simulación

Una vez definido el *Job*, se ejecuta en *Submit*. Cuando el cálculo haya finalizado, el estado aparecerá como *Completed* (Ilustración 30). En caso de haber algún fallo, aparecerá *Aborted* y se podrá consultar más información

sobre este error en la tecla *Monitor*.

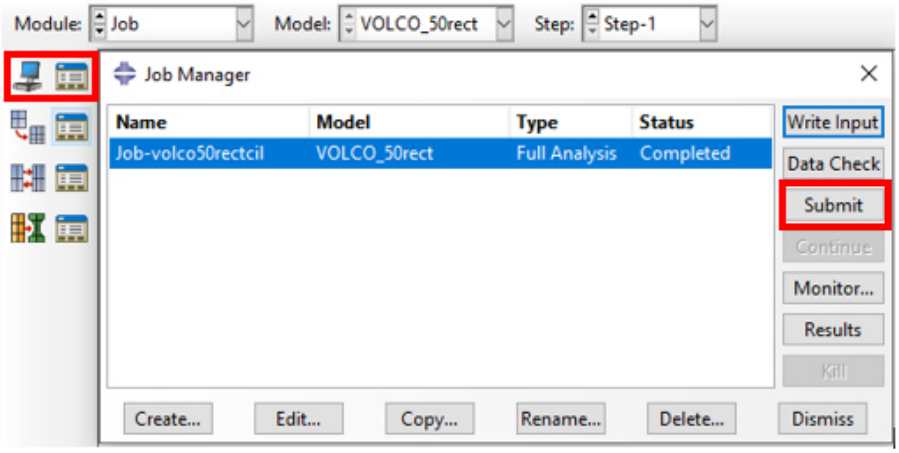


Ilustración 30. Ejecución Job

Paso 7: Consultar el resultado de la simulación en el módulo *Visualization*. Los resultados del scaffold modelado con VOLCO y con DECODE se muestran en la Ilustración 31.

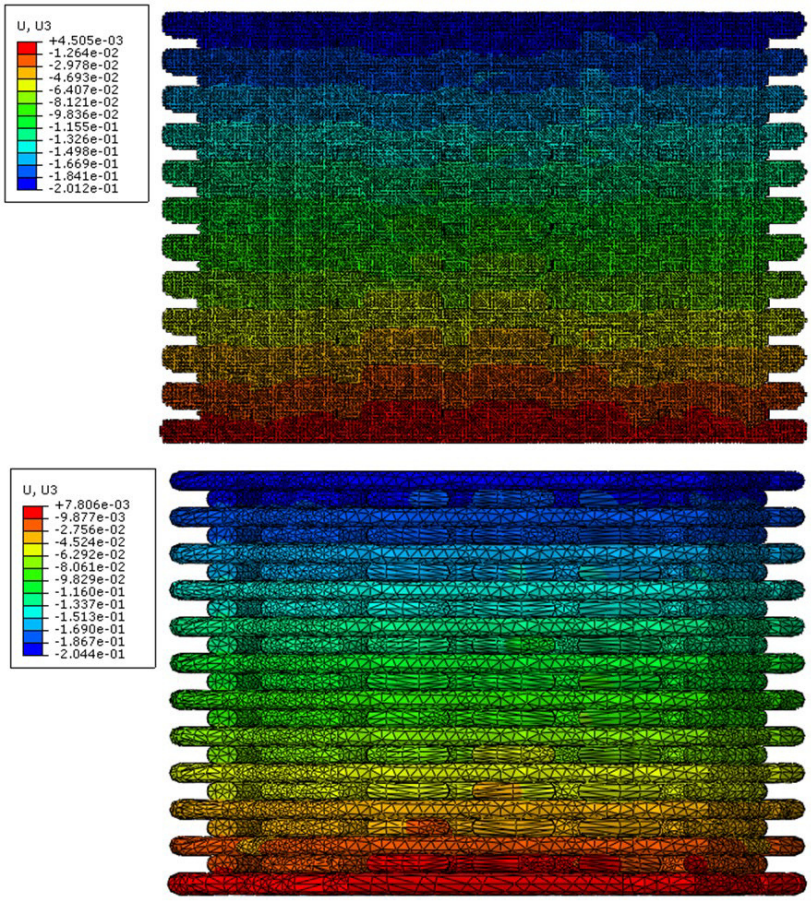


Ilustración 31. Resultado desplazamiento del scaffold a compresión: modelo voxelizado (arriba) y modelo DECODE (abajo)

Paso 8: Obtener datos. En el módulo *Visualization*, en los datos de la sesión, se crea un nuevo *XY Data* del *ODB field output* (Ilustración 32). De este, se selecciona la fuerza de reacción en Z dentro de *Unique nodal* (Ilustración 33) y se escoge el *set* de la base (Ilustración 34).

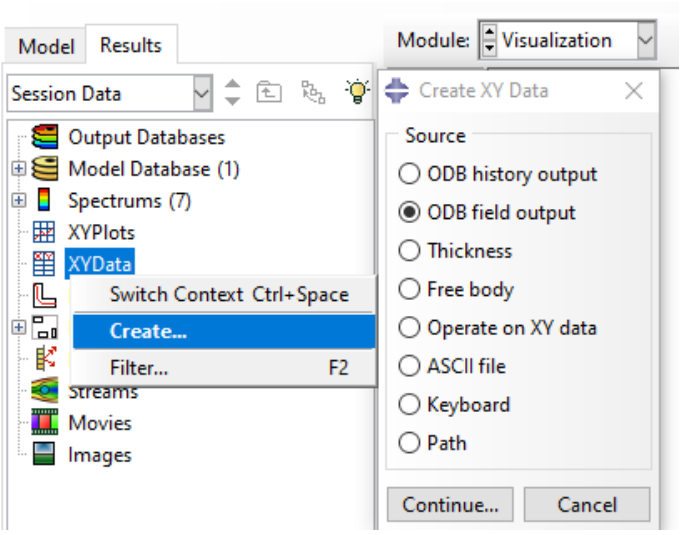


Ilustración 32. Creación XY Data

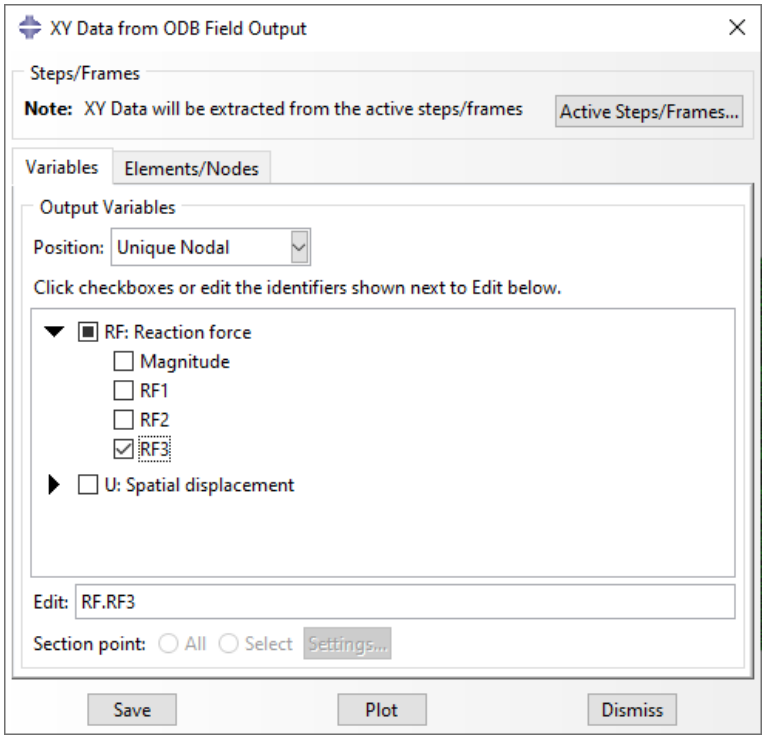


Ilustración 33. XY Data: Variables

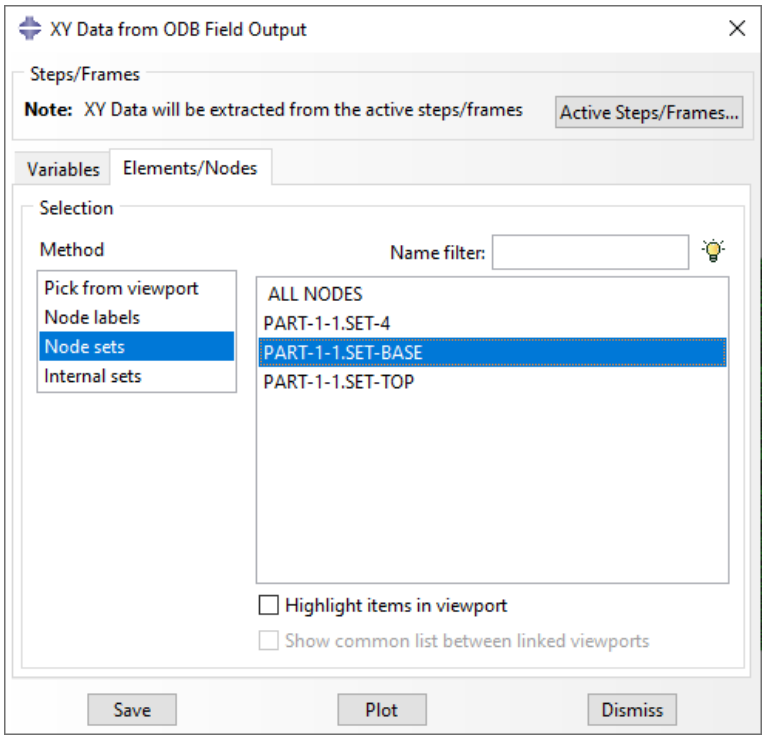


Ilustración 34. XY Data: Elementos/Nodos

Paso 9: Calcular fuerza de reacción total haciendo la sumatoria de las fuerzas en los nodos. En el módulo *Visualization*, en *XY Data*, se selecciona *Operate on XY data* (Ilustración 35).

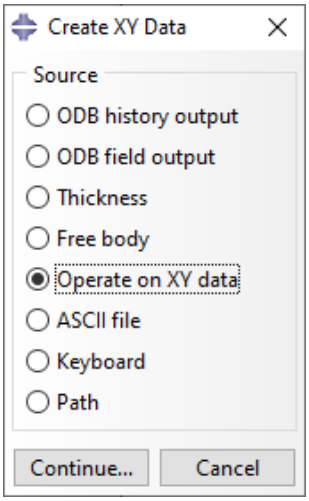


Ilustración 35. Operate on XY data

Para la sumatoria se seleccionará la operación *sum* y entre paréntesis se añadirán todos los datos, haciendo uso del *Add to Expression* (Ilustración 36). Por último, el resultado se guarda como fuerza de reacción.

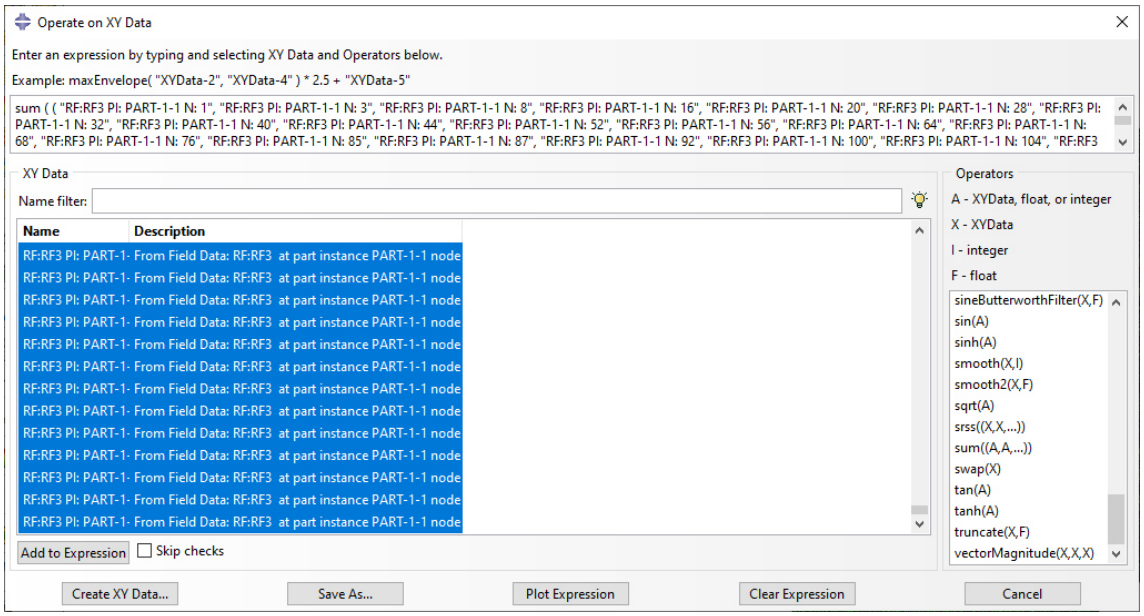


Ilustración 36. Sumatoria de datos

Paso 10: Determinar módulo elástico a partir de la fuerza de reacción obtenida en el paso anterior, el resultado será el mostrado en el *Step-1* o, lo que es lo mismo, cuando la cara se ha desplazado hasta su valor máximo (-0.2 mm). Para este cálculo se la ecuación del módulo de elasticidad a compresión extraída de la norma ISO 604 (Plásticos – Determinación de las propiedades de compresión). En este caso, el área equivalente sería el área de la base si el modelo fuera sólido.

ANEXO I. TABLAS DE DATOS DE MINIPROBETAS

En este anexo se adjuntan todas tablas de datos y resultados de las miniprobetas optimizadas en una de las aplicaciones prácticas de esta tesis.

Tabla de contenidos

Tabla 1. Configuración miniprobetas (parámetros geométricos)

Muestra	Patrón de relleno	Orientación	Altura de capa (mm)	Porcentaje de relleno (%)	Air Gap	N° perímetros
1	Concéntrico	0°	0.3	20	0	1
2	Concéntrico	0°	0.3	20	0	3
3	Concéntrico	0°	0.3	50	0	1
4	Concéntrico	0°	0.3	50	0	3
5	Concéntrico	0°	0.3	70	0	1
6	Concéntrico	0°	0.3	70	0	3
7	Concéntrico	0°	0.3	100	0	1
8	Concéntrico	0°	0.3	100	0	3
9	Concéntrico	0°	0.15	20	0	1
10	Concéntrico	0°	0.15	20	0	3
11	Concéntrico	0°	0.15	50	0	1
12	Concéntrico	0°	0.15	50	0	3
13	Concéntrico	0°	0.15	70	0	1
14	Concéntrico	0°	0.15	70	0	3
15	Concéntrico	0°	0.15	100	0	1
16	Concéntrico	0°	0.15	100	0	3
17	Concéntrico	0°	0.3	68.73	0	1
18	Concéntrico	0°	0.3	69.39	0	1
19	Concéntrico	0°	0.3	68.9	0	1
20	Concéntrico	1°	0.3	69.19	0	1

Tabla 2. Comparación tiempos de impresión teóricos y reales

Miniprobeta	Tiempo impresión teórico (s)	Tiempo impresión real de cada réplica (s)					Tiempo de impresión real medio (s)
		1	2	3	4	5	
1	95	206	194	205	212	233	210
2	154	266	258	244	247	251	253.2
3	109	206	206	233	199	210	210.8
4	154	252	244	238	244	250	245.6
5	140	196	225	218	218	228	217
6	164	261	262	260	254	261	259.6
7	136	240	234	236	240	231	236.2
8	164	256	277	279	260	282	270.8
9	153	243	259	263	275	266	261.2
10	250	358	334	342	347	347	345.6
11	200	280	286	296	318	283	292.6
12	250	356	367	348	350	344	353
13	211	403	331	305	304	303	329.2
14	267	354	349	409	365	349	365.2
15	250	347	342	355	340	340	344.8
16	268	354	353	350	354	366	355.4
17	122	234	206	216	216	213	217
18	122	234	213	210	214	208	215.8
19	122	234	200	209	182	225	210
20	122	235	228	201	199	194	211.4

Tabla 3. Comparación peso de modelos y piezas impresas

Miniprobeta	Peso real de cada réplica (g)					Peso real medio (g)	Peso modelo (g)	Error modelo-pieza impresa (%)
	1	2	3	4	5			
1	0.243	0.246	0.247	0.245	0.247	0.245	0.248	1.07
2	0.453	0.456	0.452	0.455	0.455	0.454	0.437	3.72
3	0.301	0.311	0.301	0.299	0.306	0.304	0.308	1.53
4	0.450	0.457	0.459	0.448	0.451	0.453	0.437	3.45
5	0.370	0.360	0.358	0.359	0.355	0.360	0.368	2.09
6	0.466	0.465	0.467	0.461	0.465	0.465	0.450	3.31
7	0.409	0.409	0.406	0.409	0.406	0.408	0.417	2.21
8	0.461	0.462	0.468	0.468	0.480	0.468	0.449	4.00
9	0.218	0.221	0.223	0.221	0.221	0.221	0.221	0.08
10	0.400	0.398	0.401	0.404	0.404	0.401	0.397	1.11
11	0.288	0.290	0.288	0.287	0.288	0.288	0.292	1.14
12	0.400	0.402	0.404	0.401	0.408	0.403	0.397	1.48
13	0.353	0.350	0.354	0.358	0.373	0.357	0.361	1.06
14	0.440	0.439	0.440	0.439	0.441	0.440	0.439	0.21
15	0.424	0.426	0.421	0.425	0.424	0.424	0.427	0.65

Miniprobeta	Peso real de cada réplica (g)					Peso real medio (g)	Peso modelo (g)	Error modelo-pieza impresa (%)
	1	2	3	4	5			
16	0.442	0.441	0.446	0.443	0.439	0.442	0.439	0.79
17	0.367	0.367	0.367	0.366	0.369	0.367	0.366	0.28
18	0.367	0.373	0.367	0.368	0.368	0.369	0.367	0.43
19	0.365	0.363	0.363	0.365	0.366	0.365	0.367	0.64
20	0.360	0.366	0.364	0.367	0.366	0.364	0.367	0.71

Tabla 4. Comparación longitud de modelos y piezas impresas

Miniprobeta	Longitud de cada réplica de piezas impresas (mm)					Longitud media piezas impresas (mm)	Longitud modelo (mm)
	1	2	3	4	5		
1	40.05	40.04	40.03	40.00	40.08	40.04	40
2	40.14	40.14	40.13	40.13	40.17	40.14	40
3	40.04	40.05	40.04	39.97	40.10	40.04	40
4	40.17	40.04	40.18	40.16	40.12	40.13	40
5	40.09	40.07	40.07	40.07	40.06	40.07	40
6	40.14	40.20	40.17	40.18	40.15	40.17	40
7	40.06	40.15	40.08	40.09	40.06	40.09	40
8	40.15	40.19	40.23	40.27	40.18	40.21	40
9	40.01	40.02	40.05	40.06	40.02	40.03	40
10	40.09	40.06	40.12	40.13	40.06	40.09	40
11	40.02	40.03	40.01	40.01	40.03	40.02	40
12	40.06	40.08	40.10	40.08	40.06	40.07	40
13	40.12	40.16	40.09	40.11	40.16	40.13	40
14	40.20	40.20	40.12	40.10	40.20	40.16	40
15	40.01	40.02	40.04	40.00	40.00	40.02	40
16	40.15	40.09	40.10	40.10	40.13	40.12	40
17	40.19	40.28	40.25	40.21	40.24	40.23	40
18	40.24	40.18	40.20	40.25	40.22	40.22	40
19	40.31	40.25	40.30	40.22	40.27	40.27	40
20	40.20	40.19	40.24	40.19	40.18	40.20	40

Tabla 5. Comparación ancho de modelos y piezas impresas

Miniprobeta	Ancho de cada réplica de piezas impresas (mm)					Ancho medio piezas impresas (mm)	Ancho modelo (mm)
	1	2	3	4	5		
1	4.94	5.00	5.05	5.02	4.91	4.98	5
2	5.21	5.20	5.19	5.19	5.20	5.20	5
3	4.96	5.04	4.98	4.99	5.02	5.00	5
4	5.22	5.17	5.25	5.16	5.15	5.19	5
5	5.15	5.09	5.08	5.12	5.04	5.10	5
6	5.26	5.28	5.29	5.26	5.27	5.27	5

Miniprobeta	Ancho de cada réplica de piezas impresas (mm)					Ancho medio piezas impresas (mm)	Ancho modelo (mm)
	1	2	3	4	5		
7	5.02	5.04	5.09	5.07	5.00	5.04	5
8	5.25	5.25	5.22	5.27	5.26	5.25	5
9	4.84	5.01	4.94	4.98	4.89	4.93	5
10	5.22	5.21	5.22	5.24	5.17	5.21	5
11	5.05	5.04	5.04	5.05	4.92	5.02	5
12	5.19	5.19	5.15	5.23	5.18	5.19	5
13	5.22	5.15	5.20	5.20	5.26	5.21	5
14	5.29	5.28	5.27	5.24	5.25	5.27	5
15	5.08	5.16	5.13	5.10	5.07	5.11	5
16	5.24	5.26	5.24	5.22	5.25	5.24	5
17	5.21	5.29	5.16	5.22	5.22	5.22	5
18	5.20	5.23	5.12	5.13	5.15	5.17	5
19	5.13	5.37	5.28	5.30	5.32	5.28	5
20	5.09	5.07	5.18	5.12	5.14	5.12	5

Tabla 6. Comparación espesor de modelos y piezas impresas

Miniprobeta	Espesor de cada réplica de piezas impresas (mm)					Espesor medio piezas impresas (mm)	Espesor modelo (mm)
	1	2	3	4	5		
1	2.12	2.10	2.08	2.08	2.09	2.10	2.1
2	2.11	2.11	2.11	2.11	2.10	2.11	2.1
3	2.09	2.15	2.15	2.12	2.14	2.13	2.1
4	2.09	2.16	2.11	2.11	2.15	2.12	2.1
5	2.09	2.09	2.08	2.09	2.09	2.09	2.1
6	2.08	2.07	2.06	2.06	2.05	2.07	2.1
7	2.10	2.11	2.06	2.09	2.10	2.09	2.1
8	2.07	2.07	2.10	2.06	2.10	2.08	2.1
9	1.97	2.00	1.96	1.96	1.95	1.97	1.95
10	1.97	1.97	1.97	1.97	1.99	1.97	1.95
11	1.97	2.05	1.98	2.00	1.98	2.00	1.95
12	1.99	1.99	1.99	1.98	1.99	1.99	1.95
13	1.92	1.91	1.96	1.97	1.92	1.94	1.95
14	1.93	1.93	1.94	1.93	1.93	1.93	1.95
15	2.00	2.01	2.00	2.01	2.01	2.01	1.95
16	1.94	1.96	1.96	1.96	1.95	1.95	1.95
17	1.99	1.93	1.97	1.97	1.99	1.97	2.1
18	2.01	1.99	2.02	1.98	1.96	1.99	2.1
19	1.99	1.94	1.99	1.99	2.00	1.98	2.1
20	2.00	1.98	1.95	2.00	1.99	1.98	2.1

Tabla 7. Datos de simulación de modelos de miniprobeta

Miniprobeta	Tamaño de semilla (mm)	Fuerza de reacción (N)	Módulo de elasticidad (MPa)
1	1	13.49	2386.53
2	1	18.64	3297.68
3	1	15.33	2712.50
4	1	18.64	3297.59
5	1	16.76	2965.79
6	1	18.81	3328.12
7	1	17.96	3176.99
8	1	18.80	3326.51
9	1	9.95	2197.47
10	1	16.30	3602.55
11	1	12.15	2685.10
12	1	16.31	3602.97
13	0.3	14.31	3162.46
14	0.146	17.22	3804.89
15	1	16.10	3558.09
16	0.1	17.01	3759.21
17	1	15.33	2711.35
18	1	15.30	2707.65
19	1	15.31	2707.70
20	1	15.31	2708.36

Tabla 8. Comparación módulo de Young de modelos y piezas impresas

Miniprobeta	Módulo de Young real de cada réplica (MPa)					Módulo de Young medio real (MPa)	Módulo de Young modelo (MPa)	Error (%)
	1	2	3	4	5			
1	1402.1	1419.5	1474.9	1483.6	1457.0	1447.4	2386.5	64.9
2	2466.0	2311.6	2354.8	2471.0	2403.9	2401.4	3297.7	37.3
3	1842.3	1599.6	1834.7	1831.5	1686.4	1758.9	2712.5	54.2
4	2612.6	2498.6	2589.7	2487.7	2500.5	2537.8	3297.6	29.9
5	2278.7	2189.6	2069.1	2262.5	2355.4	2231.1	2965.8	32.9
6	2645.7	2534.2	2678.6	2643.8	2664.5	2633.3	3328.1	26.4
7	2424.8	2405.8	2483.6	2299.9	2502.0	2423.2	3177.0	31.1
8	2668.2	2614.6	2571.3	2548.0	2479.3	2576.3	3326.5	29.1
9	1345.6	1362.0	1321.9	1407.5	1484.1	1384.2	2197.5	58.7
10	2395.5	2479.9	2484.6	2579.5	2546.6	2497.2	3602.6	44.3
11	1954.1	1757.7	2044.6	1924.6	2001.6	1936.5	2685.1	38.7
12	2497.4	2529.5	2631.0	2555.8	2600.6	2562.9	3603.0	40.6
13	2158.5	2385.8	2263.9	2140.7	2243.7	2238.5	3162.5	41.3
14	2757.6	2613.5	2514.9	2856.9	2775.7	2703.7	3804.9	40.7
15	2585.2	2514.1	2518.2	2636.9	2619.0	2574.7	3558.1	38.2
16	2686.2	2725.2	2607.0	2707.9	2785.8	2702.4	3759.2	39.1
17	2019.0	2109.7	2239.5	2265.3	2199.6	2166.6	2711.3	25.1

Miniprobeta	Módulo de Young real de cada réplica (MPa)					Módulo de Young medio real (MPa)	Módulo de Young modelo (MPa)	Error (%)
	1	2	3	4	5			
18	2222.6	2334.5	2302.1	2280.7	2353.6	2298.7	2707.7	17.8
19	2035.6	2145.5	2095.3	2014.3	1984.6	2055.1	2707.7	31.8
20	2176.2	2125.5	2162.4	2060.9	2178.0	2140.6	2708.4	26.5

ANEXO J. SUBRUTINA PARA EL CÁLCULO DE LA INTERCONECTIVIDAD

```
file_name=input('Introduzca el nombre de la matriz (terminado en .mat):
','s');
load(file_name);
%Definir área de la matriz en la que se encuentra el scaffold
matriz_0_1=VoxelMatrix(71:261,71:262,1:180);

pore3d=~logical(matriz_0_1); %Se da el valor 1 a los huecos (aire)

porosidad=mean(pore3d(:))*100 %Porosidad en porcentaje
[N1,N2,N3]=size(pore3d);
output1=zeros(N1,N2,N3);
output2=zeros(N1,N2,N3);

% Para medir el EPCI en cada dirección, se hace un & con la matriz de
%la siguiente capa y la matriz anterior desplazada en todas las
%direcciones una unidad. Al hacer el &, si coinciden, quiere decir que
%todos los poros resultantes del & y que sean 1, están conectados en
%términos de aire.

% Comparación en Z
for k=1:N3
  if k==1
    output1(:,:,k)=pore3d(:,:,k); %El output1 va leyendo desde el
    principio, y lo anota al ppio de output1
    output2(:,:,N3)=pore3d(:,:,N3); %El output2 va leyendo desde el
    final, y lo anota al final de output2
  else
    %Toma esa capa y hace un and con la anterior desplazada 1 unidad en
    las 8 celdas contiguas
    output1(:,:,k)=pore3d(:,:,k)&(imtranslate(output1(:,:,k-1),[0,0])
    ...
      |imtranslate(output1(:,:,k-1),[1,0])...
      |imtranslate(output1(:,:,k-1),[1,1])...
      |imtranslate(output1(:,:,k-1),[0,1])...
      |imtranslate(output1(:,:,k-1),[-1,1])...
      |imtranslate(output1(:,:,k-1),[-1,0])...
      |imtranslate(output1(:,:,k-1),[-1,-1])...
      |imtranslate(output1(:,:,k-1),[0,-1])...
      |imtranslate(output1(:,:,k-1),[1,-1]));

    output2(:,:,N3-k+1)=pore3d(:,:,N3-k+1)&(imtranslate(output2(:,:,
    N3-k+2),[0,0])...
      |imtranslate(output2(:,:,N3-k+2),[1,0])...
      |imtranslate(output2(:,:,N3-k+2),[1,1])...
      |imtranslate(output2(:,:,N3-k+2),[0,1])...
      |imtranslate(output2(:,:,N3-k+2),[-1,1])...
      |imtranslate(output2(:,:,N3-k+2),[-1,0])...
      |imtranslate(output2(:,:,N3-k+2),[-1,-1])...
      |imtranslate(output2(:,:,N3-k+2),[0,-1])...
      |imtranslate(output2(:,:,N3-k+2),[1,-1]));
  end
end

EPCIZ1 = sum(sum(output1(:,:,N3)))/(N1*N2); % En el sentido 1, se coge
el output1 final, porque el flujo va del principio al final
EPCIZ2 = sum(sum(output2(:,:,1)))/(N1*N2); % En el sentido 2, se coge
el output2 inicial, porqie el flujo va del final al principio
```

```
% Ahora calculamos la intersección de output1 y output2. Esta matriz
%mostrará los canales que realmente van de un extremo a otro en la
%dirección correspondiente
output = output1 & output2;

% Calculamos número de pixels de cada capa en Z (vector que contiene
%número de pixels vacíos conectados con los extremos, para cada capa)
PixelsZ=sum(sum(output,1),2);
% Calculamos el número de pixels mínimo del vector anterior (equivalente
%al área mínima)
Azmin=min(PixelsZ);
% Calculamos el Minimum Effective Cross Section Index
MECSIz=Azmin/(N1*N2);

%Comparación en Y
for j=1:N2
  if j==1
    output1(:,j,:)=pore3d(:,j,:);  %El output1 va leyendo desde el
    principio, y lo anota al ppio de output1
    output2(:,N2,:)=pore3d(:,N2,:); %El output2 va leyendo desde el
    final, y lo anota al final de output2
  else
    output1(:,j,:)=pore3d(:,j,:)&(imtranslate(output1(:,j-1,:),
    [0,0,0])...
      |imtranslate(output1(:,j-1,:),[0,1,0])...
      |imtranslate(output1(:,j-1,:),[0,1,1])...
      |imtranslate(output1(:,j-1,:),[0,0,1])...
      |imtranslate(output1(:,j-1,:),[0,-1,1])...
      |imtranslate(output1(:,j-1,:),[0,-1,0])...
      |imtranslate(output1(:,j-1,:),[0,-1,-1])...
      |imtranslate(output1(:,j-1,:),[0,0,-1])...
      |imtranslate(output1(:,j-1,:),[0,1,-1]));

    output2(:,N2-j+1,:)=pore3d(:,N2-j+1,:)&(imtranslate(output2(:,N2-
    j+2,:),[0,0,0])...
      |imtranslate(output2(:,N2-j+2,:),[0,1,0])...
      |imtranslate(output2(:,N2-j+2,:),[0,1,1])...
      |imtranslate(output2(:,N2-j+2,:),[0,0,1])...
      |imtranslate(output2(:,N2-j+2,:),[0,-1,1])...
      |imtranslate(output2(:,N2-j+2,:),[0,-1,0])...
      |imtranslate(output2(:,N2-j+2,:),[0,-1,-1])...
      |imtranslate(output2(:,N2-j+2,:),[0,0,-1])...
      |imtranslate(output2(:,N2-j+2,:),[0,1,-1]));
  end
end

EPCIY1 = sum(sum(output1(:,N2,:)))/(N1*N3); %El output1 va leyendo desde
el principio, y lo anota al ppio de output1
EPCIY2 = sum(sum(output2(:,1,:)))/(N1*N3); %El output2 va leyendo desde
el final, y lo anota al final de output2

% Ahora calculamos la intersección de output1 y output2. Esta matriz
%mostrará los canales que realmente van de un extremo a otro en la
%dirección correspondiente
output = output1 & output2;

% Calculamos número de pixels de cada capa en Y (vector que contiene
%número de pixels vacíos conectados con los extremos, para cada capa)
PixelsY=sum(sum(output,1),3);
```

```
% Calculamos el número de pixels mínimo del vector anterior (equivalente
%al área mínima)
Aymin=min(PixelsY);
% Calculamos el Minimum Effective Cross Section Index
MECSIy=Aymin/(N1*N3);

% Comparación en X
for i=1:N1
  if i==1
    output1(i,:,:)=pore3d(i,:,:);  %El output1 va leyendo desde el
    principio, y lo anota al ppio de output1
    output2(N1,:,:)=pore3d(N1,:,:); %El output2 va leyendo desde el
    final, y lo anota al final de output2
  else
    output1(i,:,:)=pore3d(i,:,:)&(imtranslate(output1(i-1,:,:),
    [0,0,0])...
      |imtranslate(output1(i-1,:,:),[1,0,0])...
      |imtranslate(output1(i-1,:,:),[1,0,1])...
      |imtranslate(output1(i-1,:,:),[0,0,1])...
      |imtranslate(output1(i-1,:,:),[-1,0,1])...
      |imtranslate(output1(i-1,:,:),[-1,0,0])...
      |imtranslate(output1(i-1,:,:),[-1,0,-1])...
      |imtranslate(output1(i-1,:,:),[0,0,-1])...
      |imtranslate(output1(i-1,:,:),[1,0,-1]));

    output2(N1-i+1,:,:)=pore3d(N1-i+1,:,:)&(imtranslate(output2(N1-
    i+2,:,:),[0,0,0])...
      |imtranslate(output2(N1-i+2,:,:),[1,0,0])...
      |imtranslate(output2(N1-i+2,:,:),[1,0,1])...
      |imtranslate(output2(N1-i+2,:,:),[0,0,1])...
      |imtranslate(output2(N1-i+2,:,:),[-1,0,1])...
      |imtranslate(output2(N1-i+2,:,:),[-1,0,0])...
      |imtranslate(output2(N1-i+2,:,:),[-1,0,-1])...
      |imtranslate(output2(N1-i+2,:,:),[0,0,-1])...
      |imtranslate(output2(N1-i+2,:,:),[1,0,-1]));
  end
end

EPCIX1 = sum(sum(output1(N1,:,:)))/(N2*N3); %El output1 va leyendo desde
el principio, y lo anota al ppio de output1
EPCIX2 = sum(sum(output2(1,:,:)))/(N2*N3); %El output2 va leyendo desde
el final, y lo anota al final de output2

% Ahora calculamos la intersección de output1 y output2. Esta matriz
%mostrará los canales que realmente van de un extremo a otro en la
%dirección correspondiente
output = output1 & output2;

% Calculamos número de pixels de cada capa en X (vector que contiene
%número de pixels vacíos conectados con los extremos, para cada capa)
PixelsX=sum(sum(output,2),3);
% Calculamos el número de pixels mínimo del vector anterior (equivalente
%al área mínima)
Axmin=min(PixelsX);
% Calculamos el Minimum Effective Cross Section Index
MECSIx=Axmin/(N2*N3);

EPCI_final  =  (EPCIZ1 + EPCIZ2 + EPCIY1 + EPCIY2 + EPCIX1 +
EPCIX2)/6*100  % EPCI == effictive pore connectivity index
```

```
MECSI_final = (MECSIx + MECSIy + MECSIz)/3*100

%DEFINIENDO PRIMERO GRUPOS INDEPENDIENTES DE HUECOS
estructura=bwconncomp(pore3d,6); %Usamos conectividad 6, es decir, se
considera vecino si toca la cara entera, no solo vértice o arista.
matriz_etiquetas=labelmatrix(estructura); %Se crea una matriz 3d con
etiquetas. Se ponen 0 donde no hay poros, es decir, donde hay material,
y un valor de etiqueta a los grupos de poros conectados.
num_grupos=estructura.NumObjects; %Número de grupos o poros (grupos de
pixels huecos conectados entre sí por caras)

%Grupos conectados entre primera y última capa en eje Z.
contador_grupos_Z=0; % Contador para el número de grupos que conectan
primera y última capa en Z
for ng=1:num_grupos
  %Si en la primera y última capa hay número ng, entonces ese grupo
  conecta primera y última capa
  if(~isempty(find(matriz_etiquetas(:,:,1)==ng))==true)&(~isempty(find
  (matriz_etiquetas(:,:,N3)==ng))==true)
    contador_grupos_Z=contador_grupos_Z+1;
    grupos_conectan_Z(contador_grupos_Z)=ng;
  end
end

% Matriz formada por elementos de los grupos seleccionados, pero con 1.
Para ello, se va añadiendo a una matriz con la operación OR los de la
matriz de etiquetas que coinciden con la selección de grupos anterior.
matriz_Z=zeros(N1,N2,N3);
for c=1:contador_grupos_Z
  matriz_grupo_con1s=matriz_etiquetas;
  matriz_grupo_con1s(matriz_grupo_con1s~=grupos_conectan_Z(c))=0;
  matriz_grupo_con1s(matriz_grupo_con1s==grupos_conectan_Z(c))=1;
  matriz_Z=matriz_Z | matriz_grupo_con1s;
end

%Calculamos EPCI otra vez para comparar ambos métodos
EPCIZ1_h = sum(sum(matriz_Z(:,:,N3)))/(N1*N2); % En el sentido 1, se
coge el output1 final, porque el flujo va del principio al final
EPCIZ2_h = sum(sum(matriz_Z(:,:,1)))/(N1*N2); % En el sentido 2, se coge
el output2 inicial, porqie el flujo va del final al principio

% Se suman los pixels huecos en cada capa:
PixelsZ_h=sum(sum(matriz_Z,1),2);
% Calculamos el número de pixels mínimo del vector anterior (equivalente
al área mínima)
Azmin_h=min(PixelsZ_h);
% Calculamos el Minimum Effective Cross Section Index
MECSIz_h=Azmin_h/(N1*N2);

%Grupos conectados entre primera y última capa en eje Y.
contador_grupos_Y=0; % Contador para el número de grupos que conectan
primera y última capa en Z
for ng=1:num_grupos
  %Si en la primera y última capa hay número ng, entonces ese grupo
  conecta primera y última capa
  if(~isempty(find(matriz_etiquetas(:,1,:)==ng))==true)&(~isempty(find
  (matriz_etiquetas(:,N2,:)==ng))==true)
    contador_grupos_Y=contador_grupos_Y+1;
    grupos_conectan_Y(contador_grupos_Y)=ng;
  end
end
```

```
% Matriz formada por elementos de los grupos seleccionados, pero con 1.
Para ello, se va añadiendo a una matriz con la operación OR los de la
matriz de etiquetas que coinciden con la selección de grupos anterior.
matriz_Y=zeros(N1,N2,N3);
for c=1:contador_grupos_Y
  matriz_grupo_con1s=matriz_etiquetas;
  matriz_grupo_con1s(matriz_grupo_con1s~=grupos_conectan_Y(c))=0;
  matriz_grupo_con1s(matriz_grupo_con1s==grupos_conectan_Y(c))=1;
  matriz_Y=matriz_Y | matriz_grupo_con1s;
end

%Calculamos EPCI otra vez para comparar ambos métodos
EPCIY1_h = sum(sum(matriz_Y(:,N2,:)))/(N1*N3); % En el sentido 1, se
coge el output1 final, porque el flujo va del principio al final
EPCIY2_h = sum(sum(matriz_Y(:,1,:)))/(N1*N3); % En el sentido 2, se coge
el output2 inicial, porqie el flujo va del final al principio
% Se suman los pixels huecos en cada capa:
PixelsY_h=sum(sum(matriz_Y,1),3);
% Calculamos el número de pixels mínimo del vector anterior (equivalente
al % área mínima)
Aymin_h=min(PixelsY_h);
% Calculamos el Minimum Effective Cross Section Index
MECSIy_h=Aymin_h/(N1*N3);

%Grupos conectados entre primera y última capa en eje X.
contador_grupos_X=0; % Contador para el número de grupos que conectan
primera y última capa en Z
for ng=1:num_grupos
  %Si en la primera y última capa hay número ng, entonces ese grupo
  conecta primera y última capa
  if(~isempty(find(matriz_etiquetas(1,:,:)==ng))==true)&(~isempty(find
  (matriz_etiquetas(N1,:,:)==ng))==true)
    contador_grupos_X=contador_grupos_X+1;
    grupos_conectan_X(contador_grupos_X)=ng;
  end
end

% Matriz formada por elementos de los grupos seleccionados, pero con 1.
Para ello, se va añadiendo a una matriz con la operación OR los de la
matriz de etiquetas que coinciden con la selección de grupos anterior.
matriz_X=zeros(N1,N2,N3);
for c=1:contador_grupos_X
  matriz_grupo_con1s=matriz_etiquetas;
  matriz_grupo_con1s(matriz_grupo_con1s~=grupos_conectan_X(c))=0;
  matriz_grupo_con1s(matriz_grupo_con1s==grupos_conectan_X(c))=1;
  matriz_X=matriz_X | matriz_grupo_con1s;
end

%Calculamos EPCI otra vez para comparar ambos métodos
EPCIX1_h = sum(sum(matriz_X(N1,:,:)))/(N2*N3); % En el sentido 1, se
coge el output1 final, porque el flujo va del principio al final
EPCIX2_h = sum(sum(matriz_X(1,:,:)))/(N2*N3); % En el sentido 2, se coge
el output2 inicial, porqie el flujo va del final al principio
% Se suman los pixels huecos en cada capa:
PixelsX_h=sum(sum(matriz_X,2),3);
% Calculamos el número de pixels mínimo del vector anterior (equivalente
al área mínima)
Axmin_h=min(PixelsX_h);
% Calculamos el Minimum Effective Cross Section Index
MECSIx_h=Axmin_h/(N2*N3);
```

```
EPCI_final_h=(EPCIZ1_h+EPCIZ2_h+EPCIY1_h+EPCIY2_h+EPCIX1_h+EPCIX2_h)/6
*100
MECSI_final_h=(MECSIx_h+MECSIy_h+MECSIz_h)/3*100

%Ahora calculamos el volumen de grupos conectados con exterior
Matriz_pixels_conectados_exterior = matriz_X | matriz_Y | matriz_Z;
Volumen_conectado_exterior=sum(sum(sum(
Matriz_pixels_conectados_exterior,1),2),3);

%Calculamos el CVI (Connected Volume Index)
CVI=Volumen_conectado_exterior/(N1*N2*N3)*100

%Calculamos el diámetro equivalente medio de todos los grupos
Suma_D_eq_todos=0;
for ng=1:num_grupos
   Tabla_con_d_eq_ng=regionprops3(matriz_etiquetas==ng,
   'EquivDiameter');
   Suma_D_eq_todos = Suma_D_eq_todos + Tabla_con_d_eq_ng{1,1};
end
%Calculamos el diámetro equivalente medio:
D_eq_todos=Suma_D_eq_todos/num_grupos

%Calculamos el diámetro equivalente medio de todos los grupos que
conectan el exterior (solo se consideran los grupos que conectan una
cara superior e inferior en una dirección, no la mezcla, es decir, una
esquina no vale)
Suma_D_eq_conectan_exterior=0;
contador_grupos_conectan_exterior=0;
for ng=1:num_grupos
   if(~isempty(find(matriz_etiquetas(1,:,:)==ng))==true)&(~isempty(find
   (matriz_etiquetas(N1,:,:)==ng)) == true)...
      |(~isempty(find(matriz_etiquetas(:,1,:)==ng))==true)&(~isempty(
      find(matriz_etiquetas(:,N2,:)==ng)) == true)...
      |(~isempty(find(matriz_etiquetas(:,:,1)==ng))==true)&(~isempty(
      find(matriz_etiquetas(:,:,N3)==ng)) == true)...

      %Si en la primera y última capa hay número ng, entonces ese grupo
      conecta primera y última capa
      contador_grupos_conectan_exterior=
      contador_grupos_conectan_exterior+1;
      Tabla_con_d_eq_conectan_ext_ng=regionprops3(matriz_etiquetas==ng,
      'EquivDiameter');
      Suma_D_eq_conectan_exterior=Suma_D_eq_conectan_exterior+
      Tabla_con_d_eq_conectan_ext_ng{1,1};

      Tabla_ejes_elipsoide_conectan_ext=regionprops3(matriz_etiquetas==
      ng,'PrincipalAxisLength');
      Matriz_ejes_elipsoide_conectan_ext(
      contador_grupos_conectan_exterior,:)=
      Tabla_ejes_elipsoide_conectan_ext{1,1};
   end
end

%Calculamos el diámetro equivalente medio:
D_eq_conectan_exterior =
Suma_D_eq_conectan_exterior/contador_grupos_conectan_exterior

%Calculamos ejes medios y desviaciones típicas poblacional (w=1) de
los elipsoides con segundo momento central igual a cada grupo
```

```
Eje1_medio_elipsoide_conectan_exterior=mean(
Matriz_ejes_elipsoide_conectan_ext(:,1))

Desv_tipica_Eje1_medio_elipsoide_conectan_exterior=std(
Matriz_ejes_elipsoide_conectan_ext(:,1),1)

Eje2_medio_elipsoide_conectan_exterior=mean(
Matriz_ejes_elipsoide_conectan_ext(:,2))

Desv_tipica_Eje2_medio_elipsoide_conectan_exterior=std(
Matriz_ejes_elipsoide_conectan_ext(:,2),1)

Eje3_medio_elipsoide_conectan_exterior=mean(
Matriz_ejes_elipsoide_conectan_ext(:,3))

Desv_tipica_Eje3_medio_elipsoide_conectan_exterior=std(
Matriz_ejes_elipsoide_conectan_ext(:,3),1)

%Calculamos el diámetro equivalente medio de todos los grupos que
conectan el exterior (se consideran todos los grupos que conectan dos
caras del contorno)
%Aprovechamos también para calcular el CVI_2, es decir, el volumen de
grupos que conectan dos caras frente al volumen total
Suma_V_conectan_2caras=0;
Suma_D_eq_conectan_2caras=0;
contador_grupos_conectan_2caras=0;
for ng=1:num_grupos
  if (~isempty(find(matriz_etiquetas(1,:,:)==ng)) == true) &
  (~isempty(find(matriz_etiquetas(N1,:,:)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,1,:)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,N2,:)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,1)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,N3)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(N1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,1,:)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(N1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,N2,:)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(N1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,1)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(N1,:,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,N3)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(:,1,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,N2,:)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(:,1,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,1)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(:,1,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,N3)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(:,N2,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,1)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(:,N2,:)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,N3)==ng)) == true)...
    |(~isempty(find(matriz_etiquetas(:,:,1)==ng)) == true) &
    (~isempty(find(matriz_etiquetas(:,:,N3)==ng)) == true)...

    %Si en la primera y última capa en cualquier de las direcciones hay
    número ng, entonces ese grupo conecta 2 caras
```

```
        contador_grupos_conectan_2caras = contador_grupos_conectan_2caras
        + 1;
        Tabla_con_d_eq_conectan_2caras_ng=regionprops3(matriz_etiquetas==
        ng,'EquivDiameter');
        Suma_D_eq_conectan_2caras = Suma_D_eq_conectan_2caras +
        Tabla_con_d_eq_conectan_2caras_ng{1,1};

        Tabla_con_V=regionprops3(matriz_etiquetas==ng,'Volume');
        Suma_V_conectan_2caras=Suma_V_conectan_2caras+Tabla_con_V{1,1};

        Tabla_ejes_elipsoide_conectan_2caras=regionprops3(matriz_etiquetas
        ==ng,'PrincipalAxisLength');
        Matriz_ejes_elipsoide_conectan_2caras(
        contador_grupos_conectan_2caras,:)=
        Tabla_ejes_elipsoide_conectan_2caras{1,1};
    end
end

%Calculamos el CVI_2 (Connected Volume Index, es decir, el mismo que el
anterior pero esta vez teniendo en cuenta todos los grupos que conectan
dos caras cualesquiera)
CVI_2=Suma_V_conectan_2caras/(N1*N2*N3)*100

%Calculamos el diámetro equivalente medio:
D_eq_conectan_2caras=Suma_D_eq_conectan_2caras/
contador_grupos_conectan_2caras
%El diámetro sale en unidades de pixel. Es decir, si se multiplica el
valorobtenido por el tamaño de pixel, se obtiene el diámetro real.

%Calculamos ejes medios y desviaciones típicas poblacional (w=1) de los
elipsoides con segundo momento central igual a cada grupo
Eje1_medio_elipsoide_conectan_2caras=mean(
Matriz_ejes_elipsoide_conectan_2caras(:,1))

Desv_tipica_Eje1_medio_elipsoide_conectan_2caras=std(
Matriz_ejes_elipsoide_conectan_2caras(:,1),1)

Eje2_medio_elipsoide_conectan_2caras=mean(
Matriz_ejes_elipsoide_conectan_2caras(:,2))

Desv_tipica_Eje2_medio_elipsoide_conectan_2caras=std(
Matriz_ejes_elipsoide_conectan_2caras(:,2),1)

Eje3_medio_elipsoide_conectan_2caras=mean(
Matriz_ejes_elipsoide_conectan_2caras(:,3))

Desv_tipica_Eje3_medio_elipsoide_conectan_2caras=std(
Matriz_ejes_elipsoide_conectan_2caras(:,3),1)

disp('Fin')
```

ANEXO K. SUBRUTINA PARA EL CÁLCULO DEL TAMAÑO DE PORO

```
file_name=input('Introduzca el nombre de la matriz (terminado en .mat):
','s');
load(file_name);
%Definir área de la matriz en la que se encuentra el scaffold
matriz_0_1=VoxelMatrix(78:264,78:264,2:180);
%Definir el tamaño de vóxel utilizado
Voxel_size=0.05;

pore3d=~logical(matriz_0_1); %Se pone a 1 donde hay aire
[N1,N2,N3]=size(pore3d);
pore2d(:,:,1)=pore3d(:,:,1);
for i=2:1:N3
  pore2d(:,:,1)=pore2d(:,:,1) & pore3d(:,:,i);
end

%DEFINIENDO PRIMERO GRUPOS INDEPENDIENTES DE HUECOS
estructura=bwconncomp(pore2d,4); %Usamos conectividad 6, es decir, se
considera vecino si toca la cara entera, no solo vértice o arista.
matriz_etiquetas=labelmatrix(estructura); %Se crea una matriz 3d con
etiquetas. Se ponen 0 donde no hay poros, es decir, donde hay material,
y un valor de etiqueta a los grupos de poros conectados.
num_grupos=estructura.NumObjects; %Número de grupos o poros (grupos de
pixels huecos conectados entre sí por caras)

%Seleccionamos solamente los poros interiores
%Creación de la matriz borde
Matriz_borde=ones(N1,N2);
Matriz_borde(2:N1-1,2:N2-1)=0;
%Creación matriz de huecos
k=0;
for ng=1:1:num_grupos
  Matriz_hueco=zeros(N1,N2);
  for i=1:1:(N1*N2)
    if matriz_etiquetas(i)==ng
      Matriz_hueco(i)=1;
    end
  end

  if isempty(find(Matriz_borde & Matriz_hueco,1))==1
    k=k+1;
    hueco_int(k)=ng;
  end
end

if k>0
  %Calculamos el diámetro equivalente medio de todos los grupos
  Suma_D_eq_todos=0;
  [fil,num_huecos_int]=size(hueco_int);
  for ng=1:1:num_huecos_int
    Tabla_con_d_eq_ng=struct2cell(regionprops(matriz_etiquetas==hueco_
    int(ng),'EquivDiameter'));
    Suma_D_eq_todos = Suma_D_eq_todos + Tabla_con_d_eq_ng{1};
  end
  %Calculamos el diámetro equivalente medio (um):
  D_eq_todos=(Suma_D_eq_todos/num_huecos_int)*0.05*1000
else
  disp('No existen huecos interiores')
end
```

```
%Vista proyección
K=mat2gray(pore2d);
imshow(K)
```