ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

DESARROLLO DE UN CÓDIGO DE PROGRAMACIÓN EN LENGUAJE PYTHON PARA GENERACIÓN DE GRÁFICAS DE ASHBY APLICADAS A LA SELECCIÓN DE MATERIALES

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECÁNICO

CHIRIBOGA DEL VALLE CRISTHIAN LEONARDO
cristhian.chiriboga@epn.edu.ec

OBANDO MARTÍNEZ ESTEBAN ANDRÉS
esteban.obando@epn.edu.ec

DIRECTOR: Ing. HIDALGO DÍAZ VÍCTOR HUGO, D.Sc.
victor.hidalgo@epn.edu.ec

CO-DIRECTOR: Ing. GRANJA RAMÍREZ MARIO GERMÁN, M.Sc.
mario.granja@epn.edu.ec

Quito, julio 2018

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por los señores **CHIRIBOGA DEL VALLE CRISTHIAN LEONARDO** y **OBANDO MARTÍNEZ ESTEBAN ANDRÉS**, bajo nuestra supervisión.

_____	_____
Ing. Víctor Hugo Hidalgo, D,Sc.	Ing. Mario Germán Granja, M,Sc
DIRECTOR DE PROYECTO	**CO-DIRECTOR DE PROYECTO**

.

DECLARACIÓN

Yo, **CHIRIBOGA DEL VALLE CRISTHIAN LEONARDO**, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Chiriboga del Valle Cristhian Leonardo

DECLARACIÓN

Yo, **OBANDO MARTÍNEZ ESTEBAN ANDRÉS**, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Obando Martínez Esteban Andrés

DEDICATORIA

Dedicado a Manuel Chiriboga y a Maryannabelly del Valle.
Padres, son mi más grande inspiración.

Cristhian Ch.

DEDICATORIA

Dedicada a Gerardo Obando y Jacqueline Martínez, padres siempre han sido la base y el apoyo de todos mis proyectos.

Esteban

AGRADECIMIENTO

Agradezco a Dios por darme vida y la gracia de todas sus bendiciones, por darme a mis padres y a mis hermanos, por guiarme siempre bajo su protección y poner en mi camino a personas que me inspiran a ser un mejor ser humano.

Mi eterno agradecimiento a mis padres. Por llenar mi vida de amor y enseñanzas.

Padre, este trabajo es la cumbre de una de las montañas más importantes de mi vida que me motivaste incansablemente a subir y alcanzar.

Madre, este trabajo es el reflejo de tu dedicación y amor hacia mi desde que te conocí por primera vez.

Jonathan, Marina y Romina, gracias hermanos porque ustedes son uno de mis motivos más fuertes para ser mejor cada día y esforzarme más, por encima de mis límites.

Agradezco a mi abuelita Ofelia y a mi abuelito Ramón. Sé que uno de sus sueños es verme graduado. Gracias por su apoyo incondicional a pesar de la distancia.

Agradezco a Valeria, por su amor, su paciencia y apoyo constante. Por ser paz y felicidad.

Agradezco a Sara, Nohemí y a Guillermo por su amistad. Sin duda, ustedes forjaron una buena parte de la persona que soy. Amigos como ustedes son un regalo de Dios.

Agradezco a Esteban, mi compañero de tesis y amigo. Por su amistad, compromiso y dedicación.

Agradezco a mi director de tesis Víctor Hidalgo y co-director de tesis Mario Granja, por su predisposición y colaboración constante en este trabajo.

Agradezco a mi grupo de amigos, Los Camaradas, por la amistad desde el inicio de los estudios de mi carrera, su compañía ha sido una grata experiencia.

Finalmente, agradezco a la Facultad de Ingeniería Mecánica, por todo lo aprendido y las experiencias vividas.

Cristhian Ch.

AGRADECIMIENTO

A Dios por permitirme culminar esta etapa de mi vida.

A mis padres Gerardo y Jacqueline, por sus enseñanzas y valores, aprender de ellos me hace mejor persona cada día.

A mis hermanos, Camila y Santiago, la razón que me impulsa a ser mejor persona día a día.

A Mishell, su apoyo fue indispensable, tanto en mi vida como en la culminación de este proyecto. Me contagió su fortaleza y complementó mi visión de las cosas.

A Leonardo, mi compañero de tesis, su trabajo y dedicación permite que completemos diversos proyectos satisfactoriamente.

A Víctor Hidalgo y Mario Granja, bajo cuya dirección nunca me sentí estancado mientras realizaba este trabajo, además de compartir su experiencia conmigo me dieron un criterio investigativo.

A los Jabas Bravas, mis compañeros de la universidad que convirtieron esta etapa en una de las mejores de mi vida.

A la Facultad de Ingeniería Mecánica y la Escuela Politécnica Nacional por todos los conocimientos adquiridos.

Esteban

ÍNDICE

INDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente documento tiene por objetivo la realización de un código de programación para la elaboración de las gráficas de Ashby enfocado a la selección de materiales mediante una interfaz gráfica desarrollada en el lenguaje de programación Python que permite seleccionar las propiedades a graficar. Se utiliza una base de datos con setenta y cinco materiales categorizados en siete regiones: metales, cerámicos, polímeros, elastómeros, naturales, espumas y compuestos; y con el rango de valores de seis propiedades: densidad, módulo de Young, resistencia mecánica, conductividad térmica, coeficiente de dilatación térmica y temperatura máxima de servicio. Con dos propiedades se grafica un diagrama de burbujas que representan el rango de cada material y una envolvente que agrupa los materiales en sus respectivas regiones utilizando geometría analítica para su construcción. Una vez obtenidas las gráficas se visualiza de forma clara el conjunto de materiales que cumplen los parámetros de diseño definidos, permitiendo modificarlos de manera sencilla, optimizando así el tiempo requerido para la selección de los mejores materiales disponibles para una función específica. Finalmente se comparó tres gráficas obtenidas con la interfaz y las realizadas por Ashby, alcanzando un error menor al 15%. En conclusión, se determinó que las gráficas elaboradas son aptas para su uso en la selección de materiales y que esto es posible de desarrollar en un software no privativo, permitiendo el acceso libre para la comunidad científica, contribuyendo al desarrollo de la investigación.

Palabras clave: Ashby, código, libre, materiales, Python.

ABSTRACT

In the present document it is proposed the development of a programming code capable of elaborating the Ashby materials charts using a visual way developed in the programming language Python. All the materials are gathered in a data base holding seventy-five materials and six properties: density, Young modulus, yield strength, thermal conductivity, thermal expansion and maximum service temperature; classified into seven regions: metals, ceramics, polymers, elastomers, naturals, foams and composites. By choosing two properties the code creates a bubble diagram that represents the values range of each material and wraps them into a region envelope created using analytic geometry methods. When the graphs are done, the code displays them in a simple way allowing to modify them easily, optimizing the required time to choose the best materials that are capable of fulfilling the objective. Finally, three graphs are compared using Ashby diagrams and the ones displayed by the Python code, reaching an error less than 15% which allows them to be used in the materials selection methods, this using a free open source software granting free access to the scientific community, contributing to the research development.

Keywords: Ashby, code, free, materials, Python.

DESARROLLO DE UN CÓDIGO DE PROGRAMACIÓN EN LENGUAJE PYTHON PARA GENERACIÓN DE GRÁFICAS DE ASHBY APLICADAS A LA SELECCIÓN DE MATERIALES

INTRODUCCIÓN

Cada material es una composición única de atributos denominados propiedades que limitan el rendimiento de los mismos [1], lo cual requiere un análisis adecuado que permita escoger qué material es idóneo utilizar para cumplir con un objetivo planteado. La selección de materiales es un proceso importante dentro del diseño debido a que garantiza que las partes mecánicas diseñadas tengan un funcionamiento correcto.

Existen varios métodos para realizar la selección de materiales, de los cuales la mayoría parten de la disponibilidad de una amplia gama de materiales, donde se deben analizar y refinar, ya sea con ayuda de recomendaciones, mapas de materiales o información escrita. “En general, el refinamiento se hace de acuerdo con las propiedades exigidas por el componente a diseñar y sustentado con criterios como: disponibilidad, facilidad de obtención, vida de servicio, factores ambientales y costos, entre otros. De esta forma, se llega a la selección de un único tipo de material, el cual debe resultar en el más apropiado para el fin pretendido.” [2]

Este trabajo se enfoca en la selección de materiales mediante el método gráfico, el cual se lo realiza usando mapas de materiales. Los mapas más importantes son los diagramas de Ashby, en los que se relacionan las propiedades de los materiales por pares. Los diagramas también muestran que las propiedades de las diferentes familias de materiales pueden variar ampliamente, formando grupos o regiones que se ubican en áreas cerradas (metales, cerámicos, polímeros, elastómeros, espumas, naturales y compuestos).

Para facilitar la aplicación del método gráfico existen varios softwares que permiten realizar la selección de materiales de manera sistematizada, siendo el más conocido el CES Edupack, que es ampliamente utilizado en el mundo de diseño y la ingeniería. Una gran cantidad de empresas y universidades utilizan sus más de 3000 materiales y 200 procesos disponibles. “CES Edupack es utilizado por más de 1000 universidades en el mundo para apoyar la educación en materiales, desde cursos introductorios de materiales hasta investigaciones de doctorado” [3]. La desventaja es que el costo de la licencia puede llegar hasta 413 dólares mensuales por usuario [4], lo que implica que limita el desarrollo del conocimiento ya que crea un círculo privado en el cual las investigaciones quedan reservadas para quienes puedan pagar dichas licencias.

La contribución práctica es crear un código de programación utilizando un FOSS (software libre de código abierto, por sus siglas en inglés), que permita analizar y refinar los materiales utilizando el método gráfico de selección, permitiendo la modificación del mismo para adaptarse a la necesidad del investigador, sin la necesidad de adquirir licencias.

Pregunta de Investigación

¿Es posible crear un software para selección de materiales utilizando un código de programación libre y accesible para todos?

Objetivo general

Desarrollar un código de programación en lenguaje de programación Python para generación de gráficas de Ashby aplicadas a la selección de materiales.

Objetivos específicos

- Recopilar información en una base de datos en la que conste las propiedades principales de los materiales.
- Desarrollar un código de programación en el lenguaje libre Python.
- Comprobar el código desarrollado en Python mediante la comparación con las gráficas de Ashby.
- Optimizar el código para la selección de materiales.

Alcance

El presente proyecto tiene como finalidad desarrollar una interfaz gráfica para la creación de mapas de materiales que permitan optimizar el tiempo de diseño en cuanto a la selección de los materiales. Constará de la recopilación de una base de datos de 6 propiedades con 75 materiales.

1. MARCO TEÓRICO

1.1. Antecedentes

En el siglo XX empiezan a surgir términos como ambientes virtuales y ambientes sintéticos [5], concebidos como la primera idea de lo que hoy en día se conoce como softwares, pese a que la tecnología no era suficiente para materializar dichos ambientes. En los últimos años estos ambientes computacionales han ido creciendo exponencialmente, "las herramientas computacionales se soportan en nuevas tecnologías, en especial, Internet, que ha eliminado las distancias al momento de capacitar y educar." [6]

Como resultado del desarrollo de estas herramientas se ha beneficiado la investigación, un ejemplo es la tesis de maestría *"Exergy Analysis of Atmospheric Vortex Engine for Chimney Cooling Tower with Flue Gas Discharge (2012)"* perteneciente a Víctor Hugo Hidalgo, D.Sc., en la cual utiliza el módulo CFD para mecánica de fluidos en el software Ansys [7] o la tesis previo a la obtención del título de ingeniera mecánica "*Modelación y análisis de la cinemática directa e inversa del manipulador Stanford de seis grados de* libertad (2014)" de la Ing. Victoria Granja en la que se utiliza el software Matlab para desarrollar el modelado del robot Stanford. [8]

Los softwares utilizados en los dos trabajos mencionados son desarrollados por compañías privadas, por lo que para utilizarlos en más investigaciones es necesario adquirir la licencia. Esto limita la capacidad de investigación si no se cuenta con los recursos financieros suficientes por lo que actualmente se está promoviendo el desarrollo de códigos de programación en softwares libres, tal como Python. Una de las principales ventajas de Python es que el código puede ser optimizado por el investigador para realizar tareas específicas. Un ejemplo de la aplicabilidad de Python es la tesis de doctorado "*Numerical study on unsteady cavitating flow and erosion based on homogeneous mixture assumption (2016)"* de Víctor Hugo Hidalgo, D.Sc en donde realiza los estudios utilizando también software libre como OpenFOAM, Paraview Salome, GMSH y el lenguaje C++ [9]; otro ejemplo es la tesis de pregrado *"Procesamiento de imágenes mediante software libre Python para el análisis metalográfico en aceros de bajo contenido de carbono (2014)"* del Ing. Freddy Llulluna en donde pretende disminuir el tiempo de análisis y determinación del tamaño de grano y porcentajes de fases en aceros de bajo contenido de carbono [10] y por último se cita la tesis de pregrado *"Desarrollo del código de programación para procesamiento de imágenes aplicada en fundiciones nodulares (2017)"* del Ing. Aníbal Silva cuya investigación obtuvo un software capaz de realizar un análisis apropiado de metalografías de fundiciones nodulares [11], en las cuales se utiliza a Python para

desarrollar el código de programación que permita completar los respectivos objetivos de cada investigación.

1.2. Concepto de diseño

El diseño es el proceso de materializar una necesidad de mercado (idealizada) en algo tangible, obteniendo información detallada que pueda ser utilizada para manufacturar un producto que solvente dicha necesidad. El diseño es un proceso iterativo con muchas fases interactivas, lo que lo convierte en una técnica de innovación y la toma de decisiones es fundamental ya que las posibilidades en el diseño son prácticamente infinitas.

Figura 1.1. Diagrama de flujo del proceso de diseño.
(Fuente: [12])

En la figura 1.1. se puede apreciar el procedimiento completo de diseño, comienza con la identificación de una necesidad y la decisión de hacer algo al respecto. Después de varias iteraciones el resultado es la presentación de los planes para satisfacer la necesidad. De acuerdo con la naturaleza de la tarea de diseño, algunas fases de este pueden repetirse durante la vida del producto, desde la concepción hasta la terminación [12].

Dado que es un proceso iterativo, la finalidad es llegar al mejor resultado posible, si el diseño preliminar no es satisfactorio se realizan modificaciones que permitan llegar al mejor rendimiento, es decir se optimiza el diseño. Optimización es el proceso de maximizar una cantidad deseada o minimizar una indeseada [13]. Generalmente se

realiza un modelado matemático en función de la cantidad o parámetro a ser maximizado o minimizado, en donde se aplican restricciones del objetivo.
El proceso de diseño requiere y llega a un objetivo: obtener una base de datos y herramientas computarizadas que permitan evaluar diferentes casos de estudio y posibilidades de diseño, tales como modelado, optimización de rutinas y compatibilidad con otros programas de diseño.

1.3. Importancia de los materiales en el diseño

Cada parte del proceso de diseño exige decisiones en cuanto a la selección de materiales, ya que normalmente el diseño dicta que material se utilizará, pero de manera investigativa sucede el opuesto dado que se sugiere un nuevo material para un producto existente. Sin embargo, este proceso resulta complejo debido a que el número de materiales existentes es vasto, considerando que para el diseñador están disponibles más de 120000 materiales [1].
A lo largo de los años el proceso de diseño ha ido desarrollándose, desde escoger los materiales por experiencia y pasar el conocimiento de maestro a aprendiz, hasta crear una manera sistemática por niveles, en la cual se resuelve problemas de nivel a nivel. Dichos problemas no tienen una única solución, aunque existen soluciones más claras que otras. Una de las principales soluciones es la selección sistemática de materiales, ya que el diseño requiere que los materiales sean acordes y afines al proceso de manufactura a ser realizado.

1.3.1. La evolución de los materiales

A lo largo de la historia, los materiales han limitado el diseño. Su influencia es tan alta que el nombre de las eras se ha determinado mediante el material más utilizado de la época, como la piedra, bronce y hierro. Es decir, los materiales han acompañado a la humanidad desde el comienzo de su existencia. Dentro de los primeros materiales utilizados por el hombre se encuentran algunas clases de piedra y madera, huesos, fibras, plumas, conchas, piel animal e incluso el barro sirvió para propósitos específicos [14]. Los materiales se utilizaban en forma predominante para el desarrollo de herramientas, armas y refugio, también se utilizaban para decoraciones y joyería.
Incluso algunos nombres de metales han sido utilizados de manera lingüística, tal como la *edad de oro* de las civilizaciones, que implica que la civilización estuvo en su máximo apogeo en esa época, pese a que no se refleja descriptivamente los materiales utilizados en esa época, su significado es más metafórico, ya que el oro siempre ha sido

valioso a los ojos de la humanidad, esto solo denota más la importancia de los materiales.

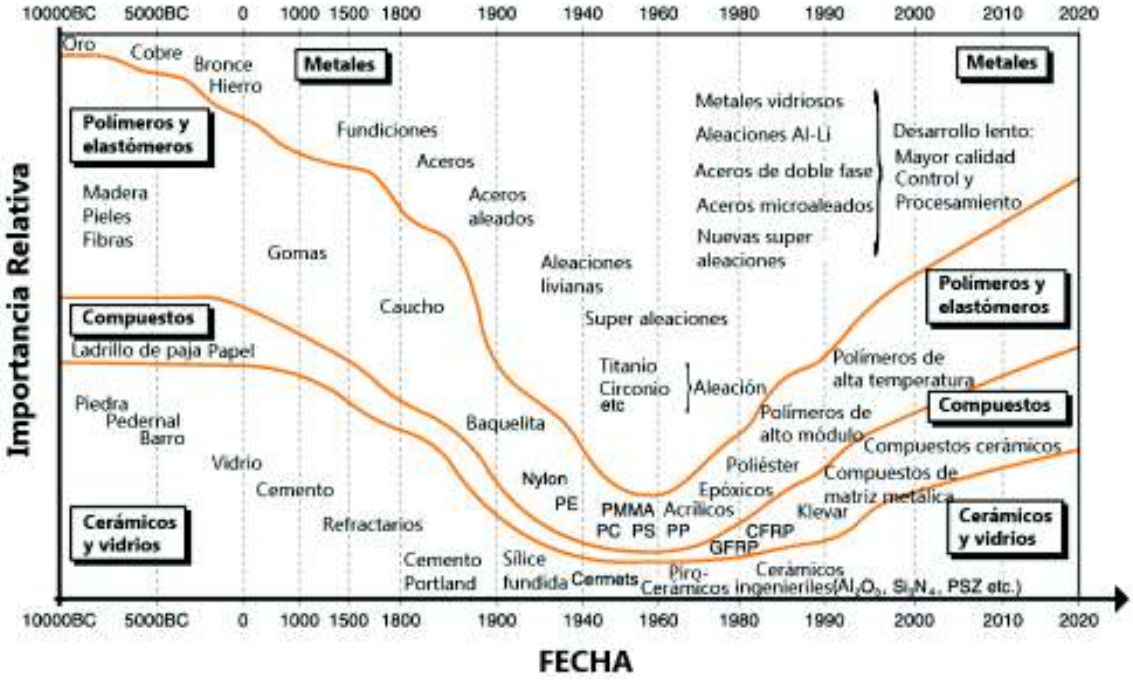

Figura 1.2. Importancia de los materiales en el tiempo.
(Fuente: [1])

En la figura 1.2. se puede apreciar el uso de los materiales desde que el hombre creó herramientas para facilitar y lograr su supervivencia, mientras que en épocas antiguas un material era el predominante, desde el siglo XIX la creación de nuevos materiales crece de manera exponencial, lo que significa que ya no es una era de un solo material, sino de varios. La ciencia de materiales se ha expandido desde la metalurgia y cerámica tradicional, a explorar nuevas áreas tales como los polímeros electrónicos, fluidos complejos, compuestos orgánicos, compuestos estructurales, materiales biomédicos, biomecánicos, cerámicos superconductores, materiales con poco impacto ambiental, nanomateriales, entre otros. Esto evidencia que los materiales están entrando en un nuevo territorio de estudio cuya tendencia se espera que continúe [14].

En la actualidad la gran cantidad de materiales puede ser un problema para el diseñador que tiene que elegir, de entre todos ellos, los mejores para completar su tarea, siendo indispensable una guía para lograr dicho cometido. Es importante realizar un análisis de todos los materiales óptimos para el desarrollo de un producto, tratando de evitar las opciones no tradicionales, ya que se limitaría la innovación. La innovación en ingeniería significa la mayoría de las veces, el uso inteligente de un nuevo material. Cabe recalcar que si el material es nuevo para una aplicación particular, no necesariamente significa

que debió ser desarrollado recientemente; por ejemplo, los polímeros tienen un mejor rendimiento que los metales en ciertas aplicaciones, tales como los clips para papel y álabes de turbina [15].

1.4. Materiales en el Ecuador

La evolución de los materiales ha sido una base fuerte para el desarrollo de nuevas tecnologías. Se puede afirmar que el avance tecnológico tiene que ir a la par y depende, hasta cierto punto, del progreso de los materiales. El Ecuador se caracteriza por ser un país dedicado a la producción de materia prima pero el programa de gobierno 2013-2017 pretende llevar al país a una transición de la fase de dependencia de productos limitados a una de productos ilimitados, tales como la ciencia, tecnología y conocimiento. [16] Según datos estadísticos del Banco Central del Ecuador, el sector manufacturero contribuyó con 14,1% al producto interno bruto (PIB) en 2010 [17], la mejora en ese porcentaje depende en gran medida del desarrollo de nuevos materiales.

En el país, la generación de materiales compuestos es un foco de investigación que está siendo explotado debido a la facilidad de obtener las materias primas requeridas, en este caso fibras. Pese a que el camino es largo y requiere un impulso en la investigación se están obteniendo buenos resultados que motivan a la continua realización de proyectos, tales como el PIC-08-493 “Desarrollo de nuevos materiales para aplicaciones estructurales e Industriales” realizado en la Escuela Politécnica Nacional, este proyecto presenta, de entre varios aspectos, estudios que permitan analizar la factibilidad de utilizar materiales conocidos en aplicaciones para las que no estaban pensados. Como ejemplo se tiene a la utilización de polímeros biodegradables, que se aplican en el área médica y la industria del empaquetamiento, para la construcción y diseño de elementos más complejos. [18]

1.5. Métodos de selección de materiales

La adecuada selección de materiales garantiza el rendimiento y seguridad de los elementos de máquina creados. Desde el punto de vista práctico, la posibilidad de usar varios métodos y poderlos confrontar, garantiza una mayor eficiencia en la selección correcta del material y un fin específico [2].

En general los métodos para la selección de materiales se basan en una serie de parámetros (físicos, mecánicos, térmicos, eléctricos) que determinan la utilidad del material y debido al alto número de factores que afectan la selección de materiales, se determinan las propiedades más relevantes para la función que se requiere. Esto se

puede realizar utilizando uno de los dos métodos generales de selección de materiales, los mismos que se describen a continuación:

El primer método es el tradicional, el cual permite al diseñador escoger el material que cree más adecuado para la tarea. Este método se basa en la experiencia, especialmente en las partes similares al elemento diseñado y que han funcionado satisfactoriamente.

El segundo es el método gráfico donde se utilizan los diagramas de Ashby, que son diagramas de burbuja que relacionan dos propiedades e incluyen índices de desempeño que generalmente hacen parte de una función objetivo, la cual maximiza o minimiza un atributo del material. El método gráfico es el enfoque de este trabajo.

1.5.1. Diagramas de Ashby

Las propiedades de los materiales no necesariamente están representadas por un solo valor, sino que abarcan un rango de valores admisibles en el cual dicha propiedad pertenece al material. A veces este rango es muy grande, por lo que es conveniente utilizar una forma gráfica de representar y visualizar las propiedades.

En la figura 1.3. se aprecia un ejemplo de diagrama para una sola propiedad (conductividad térmica), donde los materiales que tienen valores similares de las mismas propiedades se consideran dentro de una misma familia y esto se visualiza de forma sencilla en un diagrama de barras.

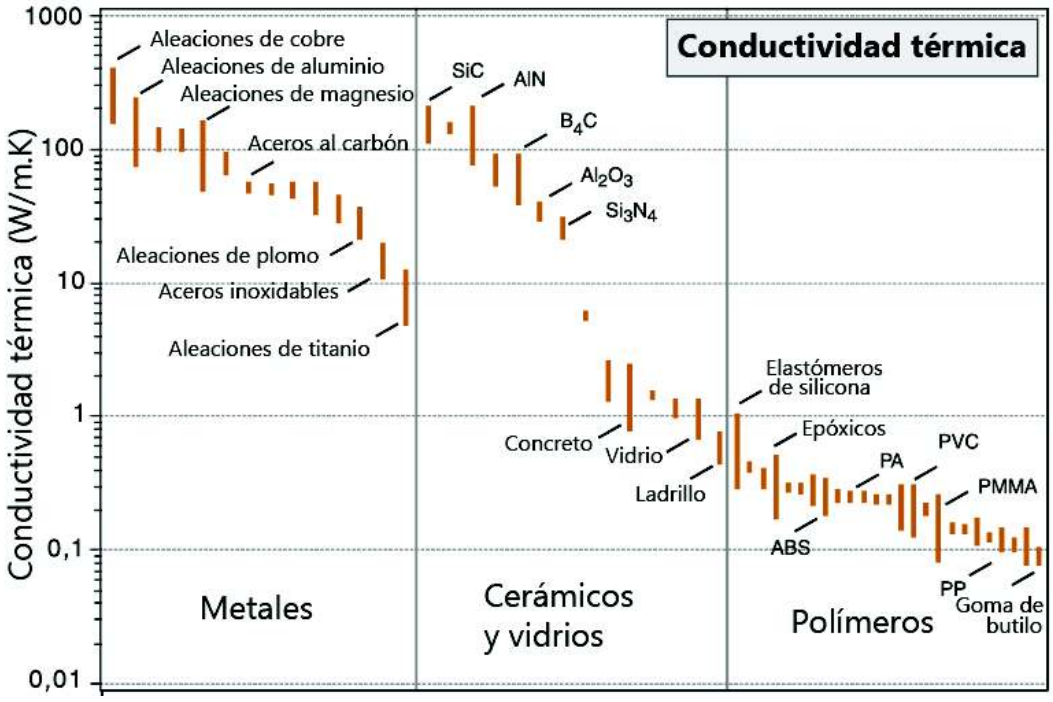

Figura 1.3. Diagrama de barras de la conductividad térmica.
(Fuente: [1])

Mucha más información es desplegada si se grafica dos propiedades de materiales en un plano, de modo que la gráfica sea un versus de propiedades. Como las propiedades de los materiales poseen un rango de valores, es adecuado utilizar un diagrama de burbujas, el cual es una mezcla entre un diagrama de dispersión y un diagrama de áreas proporcionales [16] con el fin de representar con claridad cada material. Adicional se recomienda escalar los ejes de forma logarítmica debido al amplio rango de valores logrando así una visualización completa. En la figura 1.4. se visualiza como se grafica el módulo de Young versus la densidad, de manera que abarca todo el espectro de materiales existentes, “desde la más liviana y endeble espuma, hasta el metal más pesado” [1]

Estas gráficas se denominan diagramas de Ashby, nombradas así por su creador Michael Farries Ashby, un ingeniero metalúrgico nacido en Gran Bretaña, quien desarrolló herramientas para facilitar sus estudios de mecanismos, obteniendo de esta forma los diagramas.

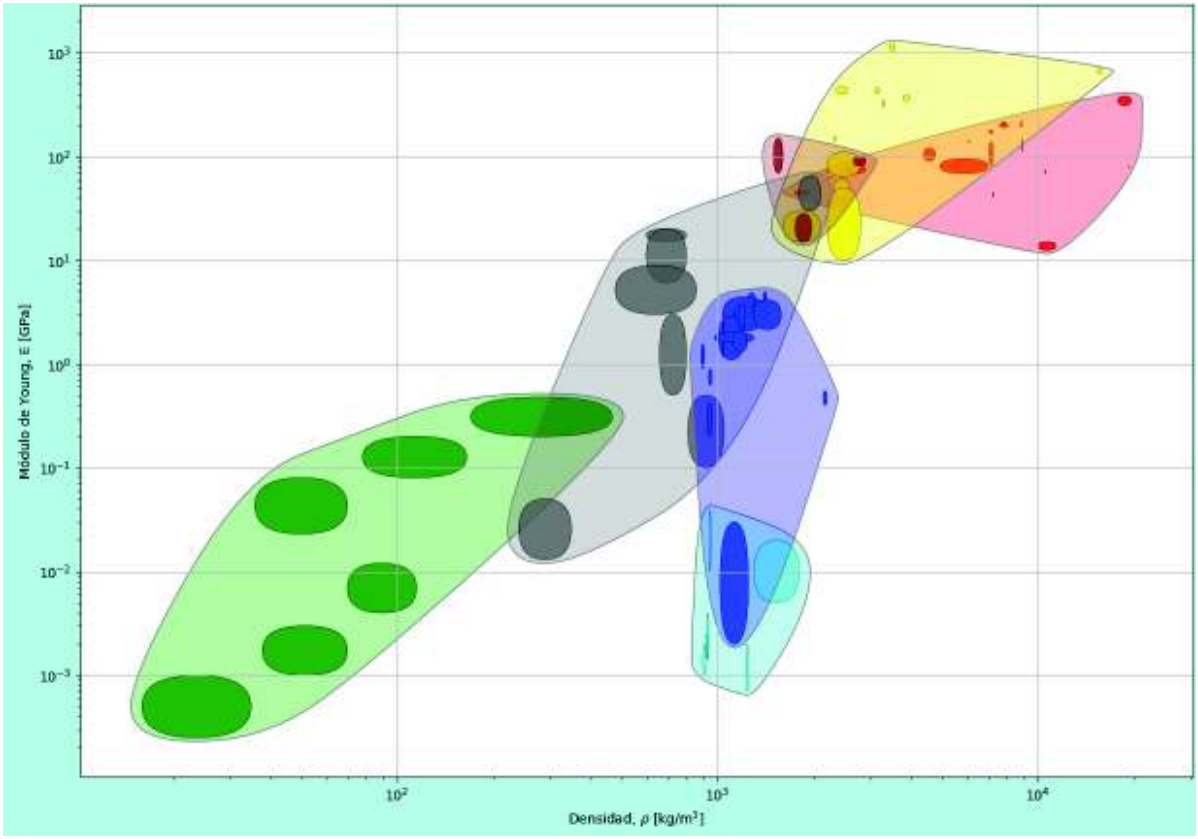

Figura 1.4. Diagrama de Ashby del módulo de Young vs. densidad utilizando Python.
(Fuente: propia)

1.5.2. Índices de materiales

Anteriormente se determinó que en el método gráfico existe la presencia de un índice de desempeño, que es parte de una función objetivo. Este índice asocia el rendimiento esperado de una función a un valor numérico, lo que permite su optimización. El rendimiento de un componente puede ser caracterizado de forma matemática mediante una ecuación, esta ecuación está compuesta por grupos de propiedades de materiales. Los índices pueden estar formados por una sola propiedad, por ejemplo, si se mide el rendimiento de acuerdo a la rigidez, el índice es el módulo de Young; o si se usa varias propiedades como la rigidez específica, donde el índice es la razón entre el módulo de Young y la densidad.

Los elementos diseñados tienen una función específica como soportar peso, soportar calor, transmitir movimiento, entre otras; y esta función viene asociada con un índice específico. La tendencia de los materiales con alto índice de rendimiento es ser más eficientes. Además, es posible comparar materiales ubicando los índices con valores similares, encontrando así varias opciones para el diseño.

Debido a que las gráficas relacionan dos propiedades y el índice representa esta relación, es posible visualizarlo dentro de la gráfica mediante una función lineal siempre que la escala sea logarítmica. Esta recta tiene como pendiente el valor del índice.

1.6. Software libre

Una herramienta computarizada es indispensable para evaluar la gran cantidad de materiales existentes, por lo que existen empresas dedicadas a desarrollar software de esa índole. Un software de los más reconocidos y utilizados es el CES Edupack, el que representa los materiales de manera gráfica y didáctica [3]. Aunque es una herramienta efectiva, se espera poder utilizar software libre, un código que pueda ser mejorado y optimizado de acuerdo a las necesidades del investigador sin limitación. Para entender claramente que es un software libre, en 1985, Richart Stallman definió el concepto de software libre en 4 libertades:

- La libertad de correr el programa, por cualquier razón.
- La libertad de estudiar las fuentes del programa y cambiarlas.
- La libertad de copiar y distribuir copias exactas cuando quiera libremente sin contratiempos.
- La libertad de hacer y distribuir copias modificadas en beneficio de la comunidad.

Las 4 libertades indican que no existe ninguna traba en el uso de software libre y puede ser un punto clave en el desarrollo de futuros proyectos de investigación. [9]

1.6.1. Python

Python es un software libre de código abierto (FOSS), el cuál presenta una serie de ventajas que lo hacen muy atractivo, tanto para su uso profesional como para su aprendizaje en la programación. Las principales ventajas se enuncian a continuación:

1. Es un lenguaje muy expresivo, es decir los programas realizados en él son muy compactos.
2. Es legible, su sintaxis es elegante y permite la escritura de programas cuya lectura es mucho más sencilla.
3. Ofrece un entorno interactivo que facilita las pruebas.
4. Su entorno de ejecución detecta muchos errores de programación que escapan al control de los compiladores.
5. Posee un gran juego de estructuras de datos que se pueden manipular de modo sencillo.
6. Su intérprete es gratuito.

Python ha sido diseñado por Guido van Rossum [19] y está en un proceso de continuo desarrollo por una gran comunidad de colaboradores. En los últimos años Python ha experimentado un aumento del número de programadores y desarrolladores y empresas que lo utilizan [20] siendo una de las razones por las cuales es conveniente su aplicación.

1.6.2. Inkscape

Inkscape es un editor de gráficos vectoriales de código abierto. Este software tiene varios creadores [21], y permite la creación y manipulación de objetos, operaciones de trazado, soporte de texto y renderización. Sus herramientas de dibujo son sofisticadas y comparables con softwares como Adobe Illustrator y CorelDRAW. El software se basa en SVG (gráficos vectoriales escalares, por sus siglas en inglés) y es una de sus principales ventajas, ya que la extensión SVG apareció como formato universal.

La calidad de la interfaz de Inkscape lo han hecho evolucionar muy rápido y cada versión se enriquece con una gran cantidad de nuevas características. Es ligero, agradable y fácil de utilizar, cabe recalcar que el software no es perfecto y muchas veces es posible que no logre realizar las tareas con la perfección que el usuario desea; sin embargo, la pronta corrección de estos errores es lo que permite seguir evolucionando al software.

El programa tiene una serie de ventajas y desventajas específicas y muy diferentes a otros softwares de edición de fotos; dado que permite trabajar con vectores, un dibujo realizado en Inkscape se puede ampliar a la perfección sin perder la calidad de su representación, lo cual sería imposible en una fotografía, esto ocasiona un fuerte impacto sobre cómo usar el programa ya que el diseñador debe trabajar siempre con la máxima precisión y prohibir cualquier aproximación. Inkscape proporciona estas

herramientas, aunque tiene complicaciones al momento de producir resultados realistas y llevaría bastante tiempo implementarlo. [22]

1.6.3. Libre Office

Libre Office es una poderosa suite de oficina, es un conglomerado de aplicaciones libres y de código abierto. Entre las aplicaciones se encuentra el software Calc, una hoja de cálculo sencilla de utilizar y con gran variedad de opciones. La gran cantidad de funciones integradas permite agilizar el uso de la misma. Entre sus ventajas tiene una gran cantidad de estilos y formatos para las celdas, las mimas son bastante flexibles y permiten rotar sus contenidos sin mayor problema. También posee plantillas preestablecidas con funciones preprogramadas para que sean reutilizables sin la necesidad de volver a programar además permite exportar hojas de cálculo con extensión XLS. [23]

2. METODOLOGÍA

2.1. Recopilación de datos de propiedades de materiales

Se definió que los diagramas de Ashby son una representación gráfica de dos propiedades de los materiales ubicadas en un plano, es por ello que su influencia en la forma de las gráficas es muy alta siendo la recopilación de datos de materiales el primer paso fundamental para la creación del código.

Se propone recopilar setenta y cinco materiales distribuidos entre las siete regiones establecidas, este reducido grupo de materiales son los más generales y disponibles en la bibliografía que además cimentará una sólida base para futuras expansiones del programa.

Las propiedades recopiladas son una parte importante dentro de este trabajo y hay que tener en cuenta algunas variables, tales como la disponibilidad, su importancia dentro de la ingeniería mecánica y la factibilidad de encontrar un diagrama de Ashby de dicha propiedad. Es por ello que el grupo de propiedades principales se restringió a seis: módulo de Young, densidad, resistencia mecánica, conductividad térmica, coeficiente de dilatación térmica y temperatura máxima de servicio.

Las propiedades escogidas se categorizan en dos grupos, propiedades mecánicas y propiedades térmicas; las primeras determinan el comportamiento de los materiales al verse sometidos a fuerzas o cargas externas mientras que las segundas controlan el comportamiento de los materiales a diferentes temperaturas. Estas propiedades son parámetros muy utilizados y representan una gran parte del diseño mecánico, con lo que es posible obtener materiales aptos con los mejores procesos de manufactura para trabajarlos.

La recopilación de los datos de materiales se basó en distintas fuentes, todas englobadas en una única base de datos ubicada en el anexo II. La bibliografía es la siguiente: "*Materials Selection in Mechanical Design"* de M. F. Ashby, es la fuente principal ya que contiene la mayoría de materiales y los ubica dentro de rangos [1]; *"Fundamentals of Material Science and Engineering"* de W.D. Callister de donde se obtiene las propiedades mecánicas de materiales adicionales. En este libro el autor solo presenta un valor [24] y no un rango de valores por lo que es necesaria bibliografía complementaria; "Engineering materials 1" de D. R. Jones y M. F. Ashby de donde se complementa las propiedades mecánicas de los materiales [15]; "Materials selection deskbook" de N. P. Cheremisinoff de donde se extraen varias propiedades, especialmente de los elastómeros y polímeros [25]; "Fundamentos de transferencia de calor y masa" de F. Incropera *et al*. De donde se obtiene las propiedades térmicas de los materiales. [26]

2.2. Estructura de datos en el registro

Una base de datos es un conjunto de datos estructurados apropiadamente y relacionados entre sí, por lo que una adecuada configuración de la misma es un aspecto primordial en el desarrollo del código.

A continuación, se detalla la estructura del registro, la cual comprende 18 campos:

Campo 1: Número de material

Campo 2: Región

Campo 3: Familia

Campo 4: Subfamilia

Campo 5: Nombre del material

Campo 6: Densidad, valor mínimo $[kg/m^3]$

Campo 7: Densidad, valor máximo $[kg/m^3]$

Campo 8: Módulo de Young, valor mínimo $[GPa]$

Campo 9: Módulo de Young, valor máximo $[GPa]$

Campo 10: Resistencia mecánica, valor mínimo $[Mpa]$

Campo 11: Resistencia mecánica, valor máximo $[MPa]$

Campo 12: Conductividad térmica, valor mínimo $[W/mK]$

Campo 13: Conductividad térmica, valor máximo $[W/mK]$

Campo 14: Coeficiente de dilatación térmica, valor mínimo $[10^{-6}/C]$

Campo 15: Coeficiente de dilatación térmica, valor máximo $[10^{-6}/C]$

Campo 16: Temperatura máxima de servicio, valor mínimo $[°C]$

Campo 17: Temperatura máxima de servicio, valor máximo $[°C]$

Campo 18: Notas

En los campos del 2 al 5 se especifica la clasificación de los materiales. Desde el campo 6 al 17 se especifican las 6 propiedades con su valor mínimo y máximo que alcanzan cada una. El campo 18 es un campo libre para que el usuario escriba notas relevantes que ayuden a su investigación.

2.3. Desarrollo del código

Para poder desarrollar el código de programación es necesario definir las funciones que debe realizar la interfaz gráfica tales como la especificación de los datos de entrada, el procesamiento de esos datos, y los datos de salida. En la figura 2.1. se muestra el diagrama de flujo con las funciones generales que debe realizar la interfaz.

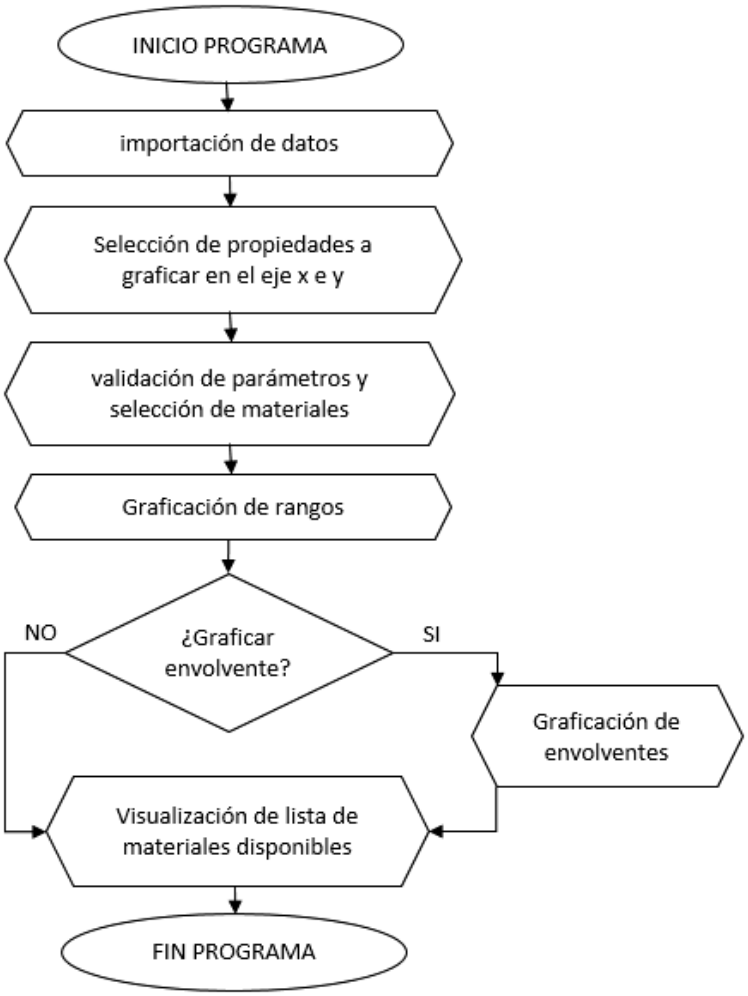

Figura 2.1. Diagrama de flujo con las funciones generales de la interfaz.
(Fuente: propia)

El código se desarrolló con 15 funciones principales, las cuales se integran y relacionan entre sí para una ejecución eficiente. En la figura 2.2. se muestra la lista de las funciones implementadas y la Clase <<*GUI ASHBY*>> la cual agrupa todas las funciones de la interfaz. A continuación, se describe brevemente lo que realizan estas funciones.

La función <<*_init_*>> especifica las características iniciales de la ventada de la interfaz, como tamaño, color de fondo, ícono de la interfaz, y además contiene el comando *self.Show* , el cual permite que se muestre la interfaz en la pantalla.

La función <<*InitUI*>> crea todos los objetos y los ubica en la ventana de la interfaz como los paneles, botones, cuadros de textos, etiquetas, entre otros. Adicionalmente, también especifica las funciones que serán llamadas desde los diferentes objetos.

La función <<*importancion_datos*>> importa los datos desde la base de datos a Python.

La función <<*validacion_parametros*>> extrae los materiales que cumplen con los parámetros definidos por el usuario.

La función <<*check_all*>> se utiliza para seleccionar o deseleccionar todos los cuadros de opciones para graficar todas las familias de materiales.

La función <<*some_material*>> permite graficar una o varias familias de materiales.

La función <<*seleccion_propiedad*>> es usada para que el código reconozca rápidamente que propiedades de los materiales serán graficadas.

La función <<*Graficar*>> permite graficar lo rangos y regiones de los materiales. Esta función es la que dibuja los diagramas de Ashby y hace uso de cuatro funciones más: *rango_materiales, grafico_regiones, contorno* y *ptos_paralelos*. En la sección 2.3.4 se describe con más detalle los procedimientos que se realizan dentro de estas funciones.

La función <<*Graficar_rectaIndice*>> dibuja una recta en el área gráfica la cual representa el índice de rendimiento, el cual se lo revisó en la sección 1.5.2.

La función <<*nombre_material*>> muestra el nombre de un material al hacer doble clic sobre el árbol de materiales.

La función <<*Mostrar_nombres*>> muestra los nombres de todos los materiales graficados.

En el anexo I se encuentra el código de programación completo, donde se detalla las líneas de programación, se especifica la utilidad de las variables principales, se añade una descripción de como se usa cada función y se explica los bucles anidados más complejos para favorecer la comprensión del código con el fin de facilitar la personalización del mismo de acuerdo a las necesidades del usuario.

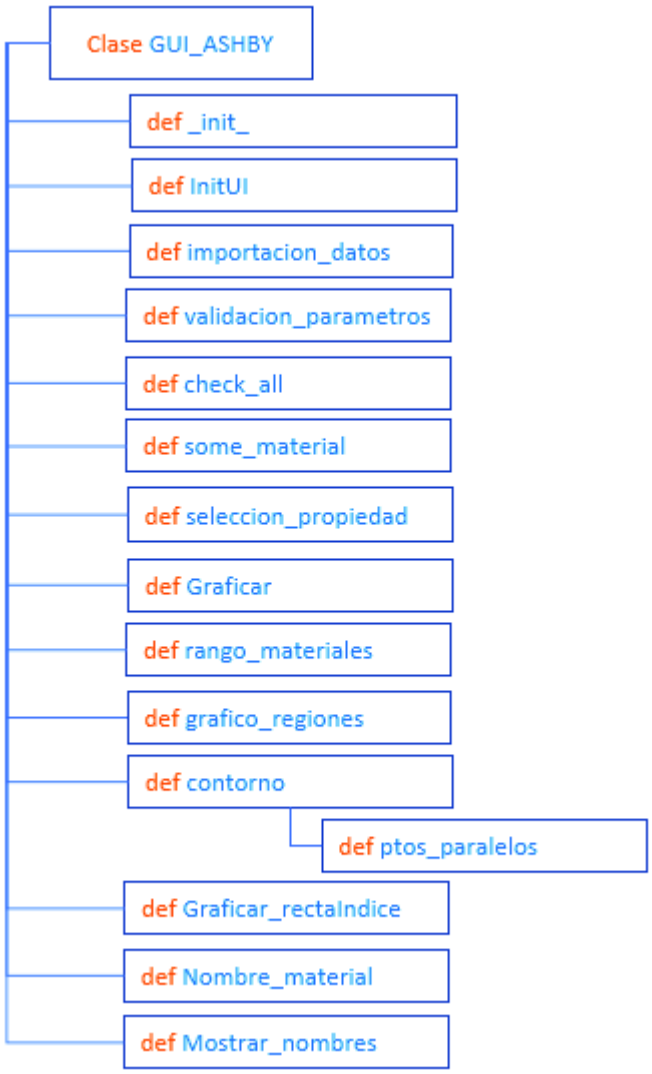

Figura 2.2. Funciones desarrolladas para la estructuración del código.
(Fuente: propia)

2.3.1. Librerías

Las librerías o módulos son simplemente archivos donde se guardan funciones útiles. El nombre del módulo es el nombre del archivo donde está guardado y pueden ser cargados en el programa con la función *<<import>>.* [27]

Para ejecutar el código de programación es necesario tener instaladas 5 librerías e importarlas, estas son: *matplotlib, numPy, scipy, xlrd* y *wxPython.*

2.3.1.1. Matplotlib

Matplotlib es una biblioteca de Python que realiza el trazado en dos dimensiones, produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos. Se puede usar para gráficas interactivas, publicaciones científicas y ayuda al desarrollo de interfaces de usuario.

Matplotlib intenta hacer más fácil las cosas ya sean éstas fáciles o difíciles. Puede generar gráficos, histogramas, espectros de potencia, gráficos de barras, diagramas de errores, diagramas de dispersión, etc., con solo unas pocas líneas de código. [10], [28]

2.3.1.2. NumPy

NumPy es un paquete fundamental para la informática científica con Python. Contiene, entre otras cosas un poderoso manejo de matriz N-dimensional y funciones sofisticadas, herramientas para integrar el código C / C ++ y Fortran, posee funciones de álgebra lineal, transformada de Fourier y generación de números aleatorios.
Además de sus usos científicos, NumPy también se puede usar como un contenedor multidimensional eficiente de datos genéricos. Se pueden definir tipos de datos arbitrarios. Esto permite a NumPy integrarse de manera rápida y sin problemas con una amplia variedad de bases de datos. [29]

2.3.1.3. Scipy

Scipy es una herramienta empleada para las matemáticas, la ciencia y la ingeniería. Esta biblioteca depende del paquete NumPy, que proporciona una rápida y cómoda manipulación de matrices n-dimensionales. El paquete Scipy está diseñado para trabajar con matrices NumPy y proporciona varias rutinas numéricas, fáciles de usar y eficientes para la integración numérica y optimización. Juntos son muy potentes en la manipulación de gran cantidad de datos. [10]

2.3.1.4. xlrd

Xlrd es una biblioteca para leer y modificar datos de archivos de hojas de cálculo con extensión .xls o .xlsx. [17] La principal ventaja de esta librería es que permite importar gran cantidad información de forma sencilla, eliminando la necesidad de importar desde archivos de texto.

2.3.1.5. wxPython

wxPython es un conjunto de herramientas GUI (Interfaz de Usuario Gráfica, por sus siglas en inglés) multiplataforma para el lenguaje de programación Python. Permite crear programas con una interfaz gráfica de usuario robusta y altamente funcional, simple y fácil. Se implementa como un conjunto de módulos de extensión de Python que envuelven los componentes de la interfaz gráfica de usuario de la popular biblioteca multiplataforma wxWidgets, que está escrita en C ++.

Con wxPython, los desarrolladores de software pueden crear interfaces de usuario verdaderamente nativas para sus aplicaciones Python, que se ejecutan con pocas o ninguna modificación en Windows, Mac y Linux u otros sistemas tipo Unix. [30]

2.3.2. Importación de datos

Este procedimiento se realiza con la función <<*importacion_datos*>>*,* la estructura de la función se ve representada en el diagrama de flujo de la figura 2.3.

Para la importación de datos se utiliza la librería *xlrd*, la cual lee el archivo con el nombre de <<*Base Materiales*>> e importa todos los datos de la *Hoja1* en una variable de tipo *lista* con el nombre de <<*tabla*>>*.*

Esta lista contiene 16 columnas. Las 4 primeras columnas almacenan los datos de los Campos 2, 3, 4 y 5. Las demás columnas contienen los valores de las propiedades. Es decir, ésta lista contiene datos alfabéticos y numéricos.

Figura 2.3. Diagrama de flujo importación de datos.
(Fuente: propia)

2.3.3. Selección de materiales

Este procedimiento se realiza con la función << *validacion_parametros*>> el cual extrae los parámetros definidos por el usuario, se construye una nueva *lista* llamada <<*LISTA_REGIONES*>> la cual almacena los materiales que cumplen los parámetros especificados y luego se la divide en 7 listas. Cada una de estas listas agrupa cada material en una de las 7 regiones. Finalmente, se convierte cada lista de regiones en una matriz sólo con los valores de las propiedades para posteriormente graficarlos en la interfaz. Todo este proceso si visualiza en la figura 2.4.

Los parámetros que se pueden definir son los valores máximos y mínimos de cualquiera de las 6 propiedades disponibles en la base de datos.

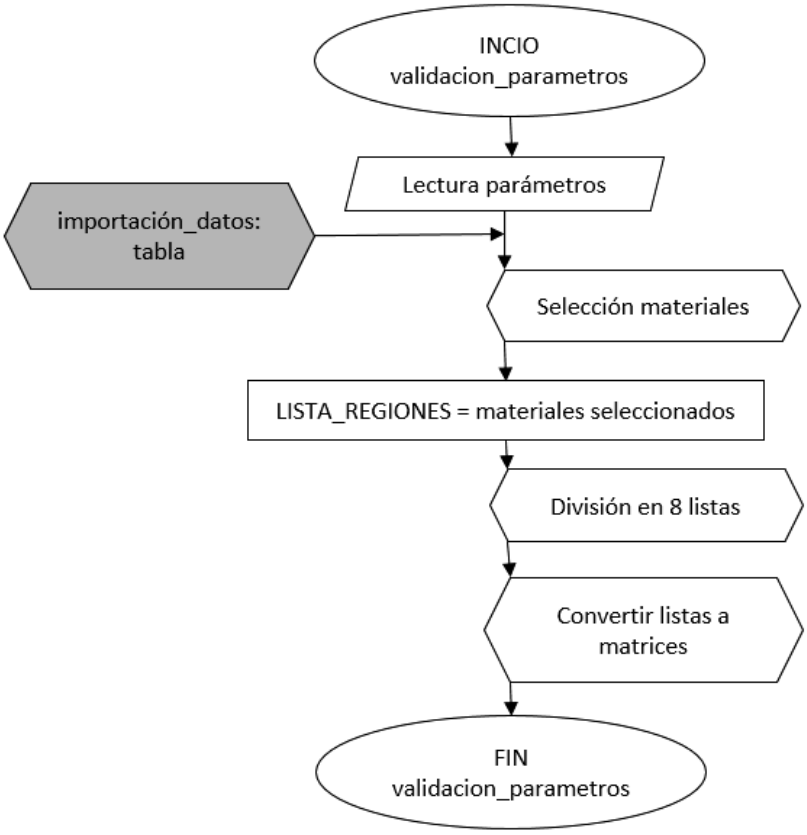

Figura 2.4. Diagrama de flujo para la selección de materiales a graficar.
(Fuente: propia)

2.3.4. Creación de gráficas

Este procedimiento se realiza con la función <<*Graficar*>> y es el procedimiento más importante debido a que grafica los diagramas de Ashby. Se compone de dos sub-procedimientos: el primero grafica las burbujas o elipses que representan los rangos de las propiedades de cada material; y el segundo grafica la envolvente que agrupa las burbujas de varios materiales. En la figura 2.5. se muestra el diagrama de flujo de este procedimiento, el cual empieza limpiando el área gráfica y el árbol de materiales. Luego se identifica las propiedades elegidas a graficar y se filtra la lista de materiales de acuerdo a los parámetros definidos por el usuario. A continuación, se ingresa a un bucle que recorre región por región graficando cada rango de materiales que cumpla con los parámetros y si la opción de graficar la envolvente está habilitada se la dibuja. Al final del bucle, se añaden los materiales disponibles al árbol de materiales para que el usuario pueda visualizarlos de forma sencilla agrupados por región, familia y subfamilia.

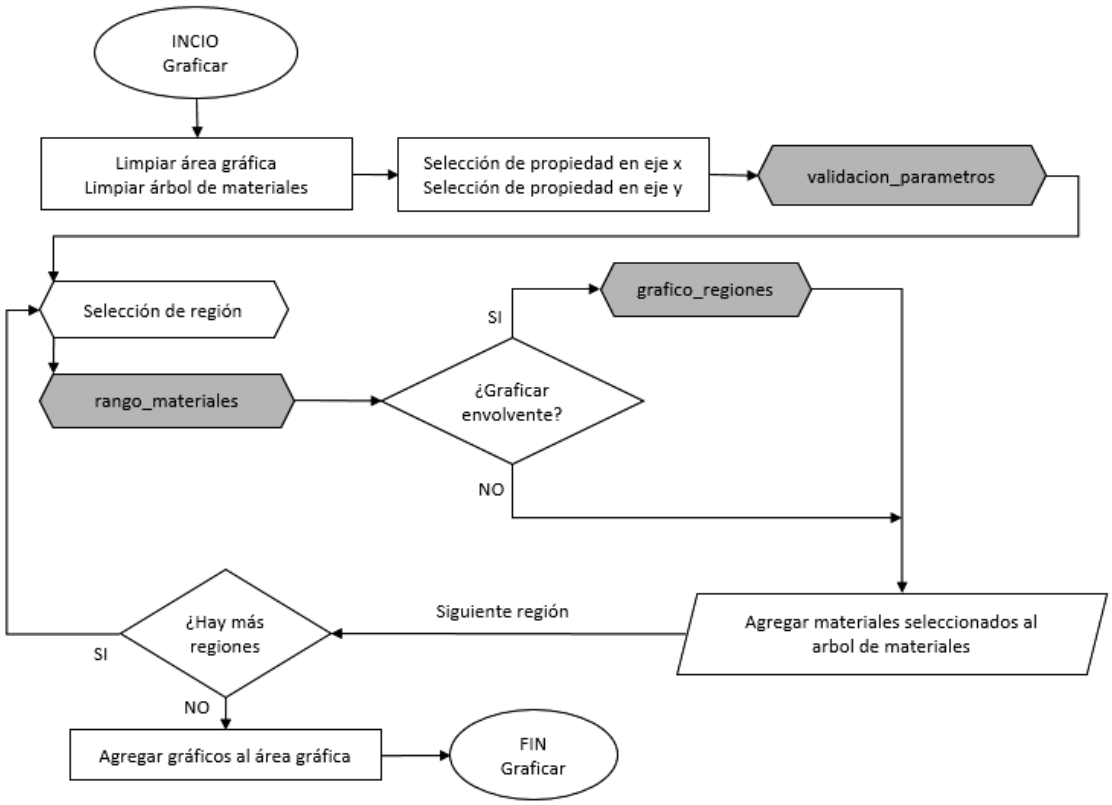

Figura 2.5. Diagrama de flujo de la función <<Graficar>>.
(Fuente: propia)

Para identificar los datos de las propiedades a graficar, cada una de éstas tiene definido un id:

- Densidad: id=0
- Módulo de Young: id=2
- Resistencia mecánica: id=4
- Conductividad térmica: id=6
- Coeficiente de dilatación térmica: id=8
- Temperatura máxima de servicio: id=10

El id es el espacio entre las propiedades en la matriz de datos, iniciando de izquierda a derecha tal como se muestra en la figura 2.6.

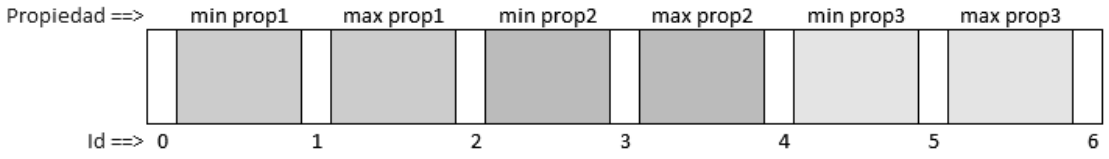

Figura 2.6. Esquema de la forma en que se establece el id de cada propiedad.
(Fuente: propia)

A continuación se detalla el proceso de elaboración de los diagramas de Ashby.

2.3.4.1. Gráfica de rangos

Este procedimiento se realiza con la función <<*rango_materiales*>>. En la figura 2.7. se muestra el diagrama de flujo donde se puede apreciar que se usa un bucle *for* para graficar la elipse de cada material.

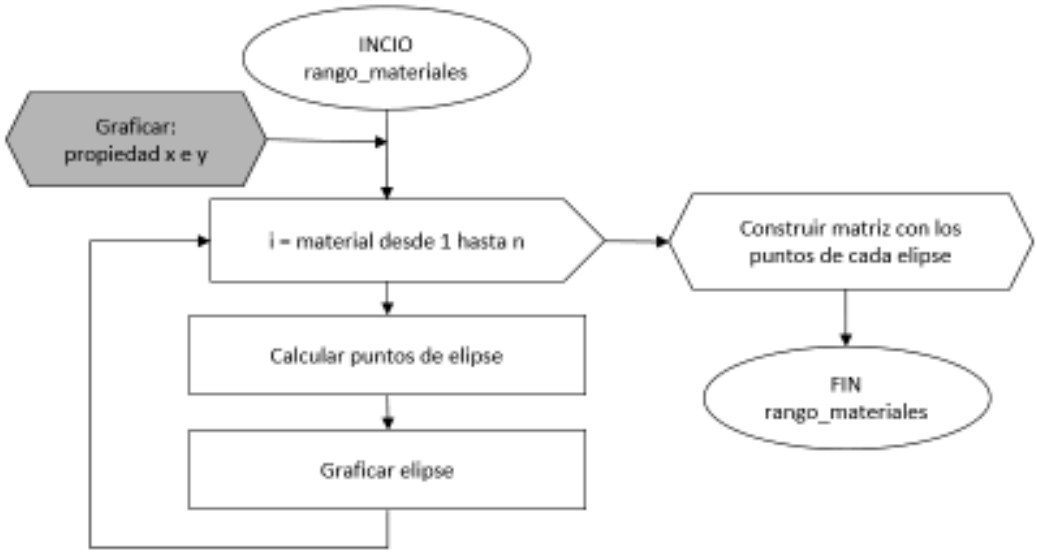

Figura 2.7. Diagrama de flujo de la función <<rango_materiales>>.
(Fuente: propia)

La librería de *Matplotlib* tiene una función sencilla para graficar elipses, sin embargo, esta función grafica elipses distorsionadas en escalas logarítmicas. Debido a eso se creó una función propia para graficar elipses.

Para graficar las elipses se construye un rectángulo en el cual estará inscrita la elipse, como se muestra en la figura 2.8. donde los lados del rectángulo son los rangos máximos y mínimos de cada propiedad.

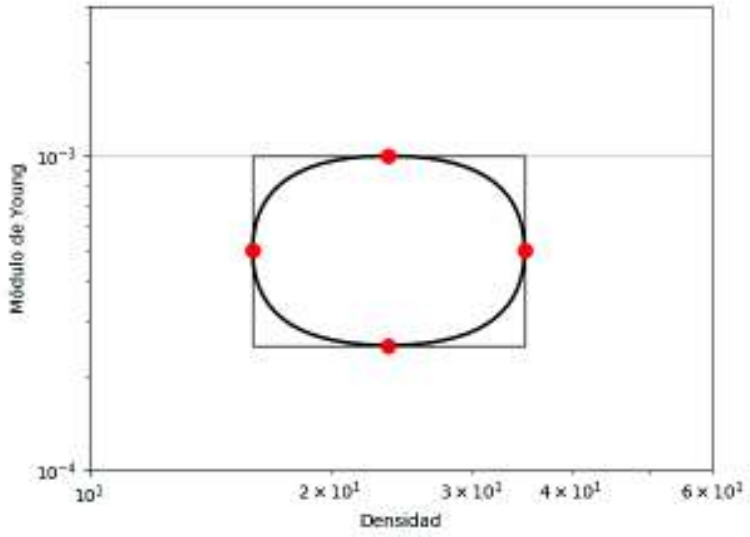

Figura 2.8. Elipse formada con los valores de los rangos de las propiedades.
(Fuente: propia)

2.3.4.2. Gráfica de regiones

Este procedimiento se lleva a cabo con la función <<*grafico_regiones*>>, la figura 2.9. muestra el procedimiento que sigue dicha función. El proceso inicia con la selección de las coordenadas de los puntos de todos los rectángulos que inscribe las elipses de cada material en cada región.

Con estos puntos se construye una matriz, se aplica la función *ConvexHull* disponible en la librería de *Scipy,* la cual retorna una matriz únicamente con los puntos externos que forman el borde de la región.

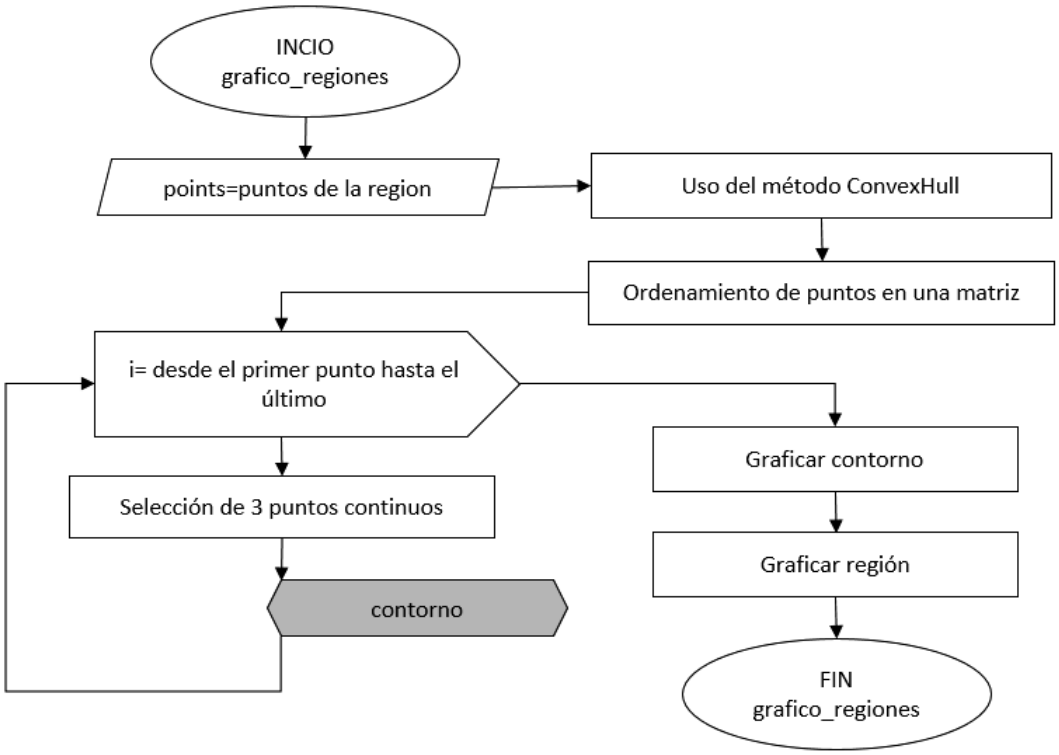

Figura 2.9. Diagrama de flujo del procedimiento <<grafico_regiones>>.
(Fuente: propia)

Luego de esto se ordenan los puntos en sentido antihorario tomando como punto inicial la coordenada con el menor valor en el eje y, tal como se aprecia en la figura 2.10. Esto implica que las gráficas varían en sus regiones de acuerdo a que materiales se encuentran en los extremos, por lo que no todos los materiales son necesarios para recrear y comparar con los diagramas de Ashby. Esto se comprueba en el siguiente capítulo.

Una vez ordenados los puntos se ingresa en un bucle *for* en donde se seleccionan 3 puntos continuos de borde para obtener los puntos que servirán para dibujar el contorno. Dentro de este bucle se usa la función <<*contorno*>> aplicado a 3 puntos continuos a lo largo de todo el borde de la región. Ver figura 2.11.

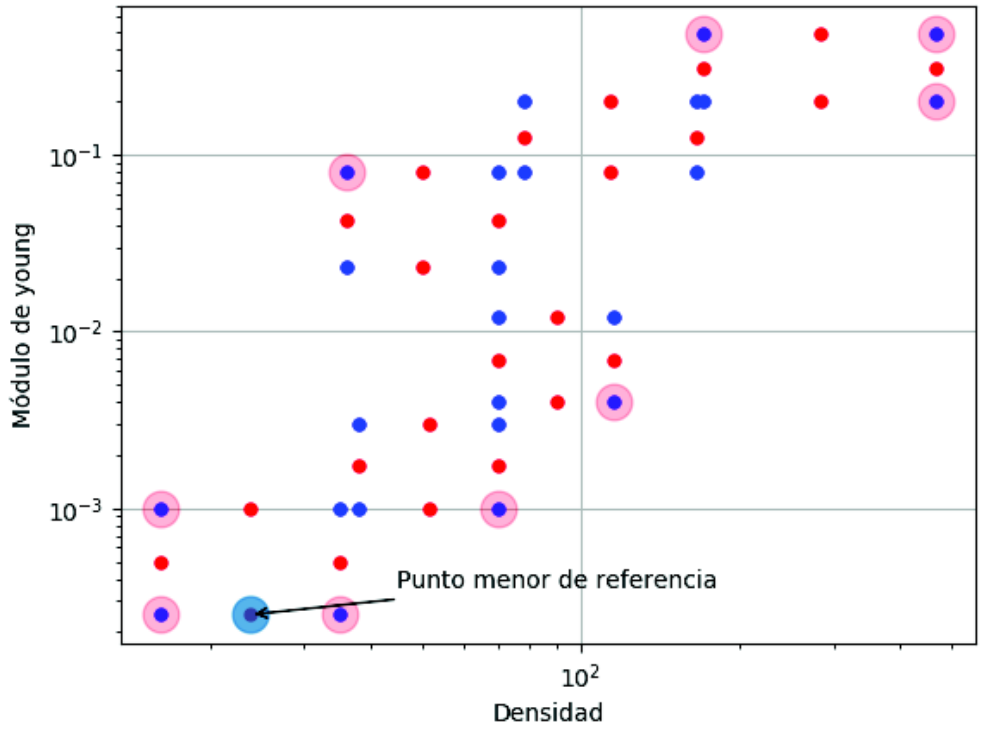

Figura 2.10. Puntos externos obtenidos de la función *ConvexHull* y el punto menor de referencia.
(Fuente: propia)

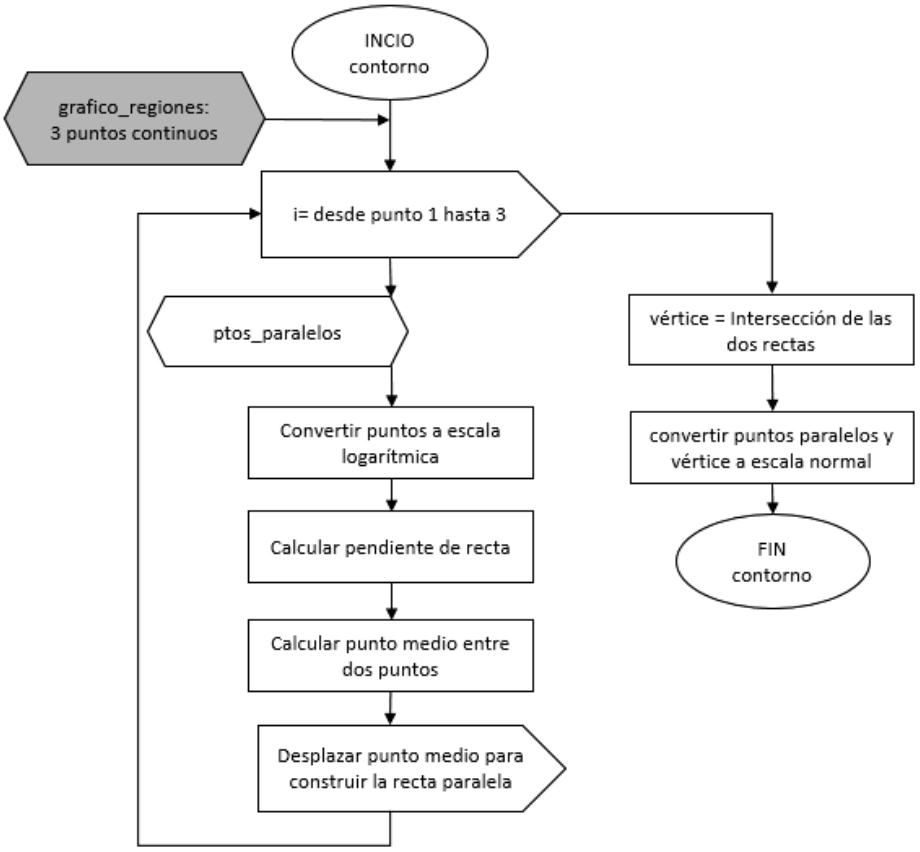

Figura 2.11. Diagrama de flujo de la función <<contorno>>.
(Fuente: propia)

La función <<*contorno*>> permite construir un polígono irregular mediante el uso de la función <<*pts_paralelos*>> la cual obtiene las coordenadas de los vértices de cada lado del polígono. Con estas coordenadas se define el polígono irregular que tendrá inscrita la región de cada material usando los puntos medios de cada lado del polígono y los vértices como puntos de control para usar la función *CURVE3* disponible en *Matplotlib* y de esta forma graficar arcos continuos. Ver figura 2.12.

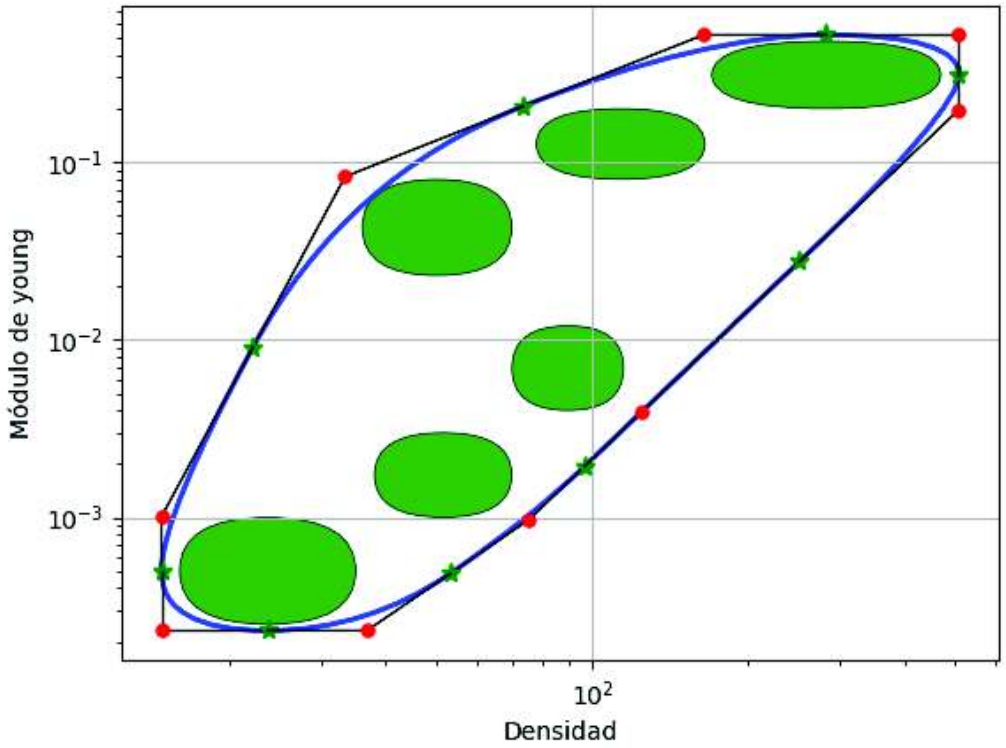

Figura 2.12. Polígono obtenido de la función <<contorno>> para obtener la envolvente de la región.
(Fuente: propia)

De esta forma se obtienen las envolventes de cada región de materiales. En donde los puntos rojos son los puntos de control, los verdes el inicio y fin de cada arco, y la línea azul es el borde obtenido al unir todos los arcos.

Es necesario hacer notar que esta metodología que usa polígonos irregulares se desarrolló con el fin de crear regiones redondeadas, valiéndose de los lados de los polígonos como rectas tangentes a las curvas y de esta forma procurar que todos los rangos de materiales queden dentro de la región.

2.3.5. Gráfica del índice

Para graficar la pendiente de la recta que representa el índice de rendimiento se desarrolló la función <<*Graficar_rectaIndice*>>. Dentro de esta función, se grafica la

recta tomando como parámetros un punto y la pendiente. El punto se obtiene al hacer clic sobre cualquier parte del área gráfica; la pendiente es especificada por el usuario de acuerdo al índice obtenido analíticamente en la resolución de un problema de ingeniería. En el anexo IV se encuentran todos los índices que pueden aplicarse en la interfaz de esta programación, en donde debido a la escala logarítmica el exponente define el valor de la pendiente.

2.4. Interfaz gráfica

Finalmente, siguiendo el diagrama de flujo que se mostró en la figura 2.1. se estructuró una interfaz gráfica que permita cumplir con las funcionalidades descritas.

La interfaz se divide en dos paneles principales, un panel de control y un panel gráfico, a su vez el panel de control se divide en seis secciones: sección de botones de mando, sección de selección de familia de materiales, sección de selección de propiedades, sección de parámetros, sección de árbol de materiales y sección de selección de índice. Todas estas secciones se observan en la figura 2.13.

En el panel de control están las opciones para definir los parámetros adecuados bajo los cuales se obtendrán las gráficas de los materiales que más se ajusten a las necesidades del usuario. La interfaz permite graficar una o varias familias de materiales, dibujar los rangos de los mismos, agruparlos por regiones y filtrarlos de acuerdo a los parámetros definidos y visualizarlos en el panel gráfico.

En el anexo IV se encuentra el manual de usuario de la interfaz gráfica, detallando todas las facilidades de uso implementadas.

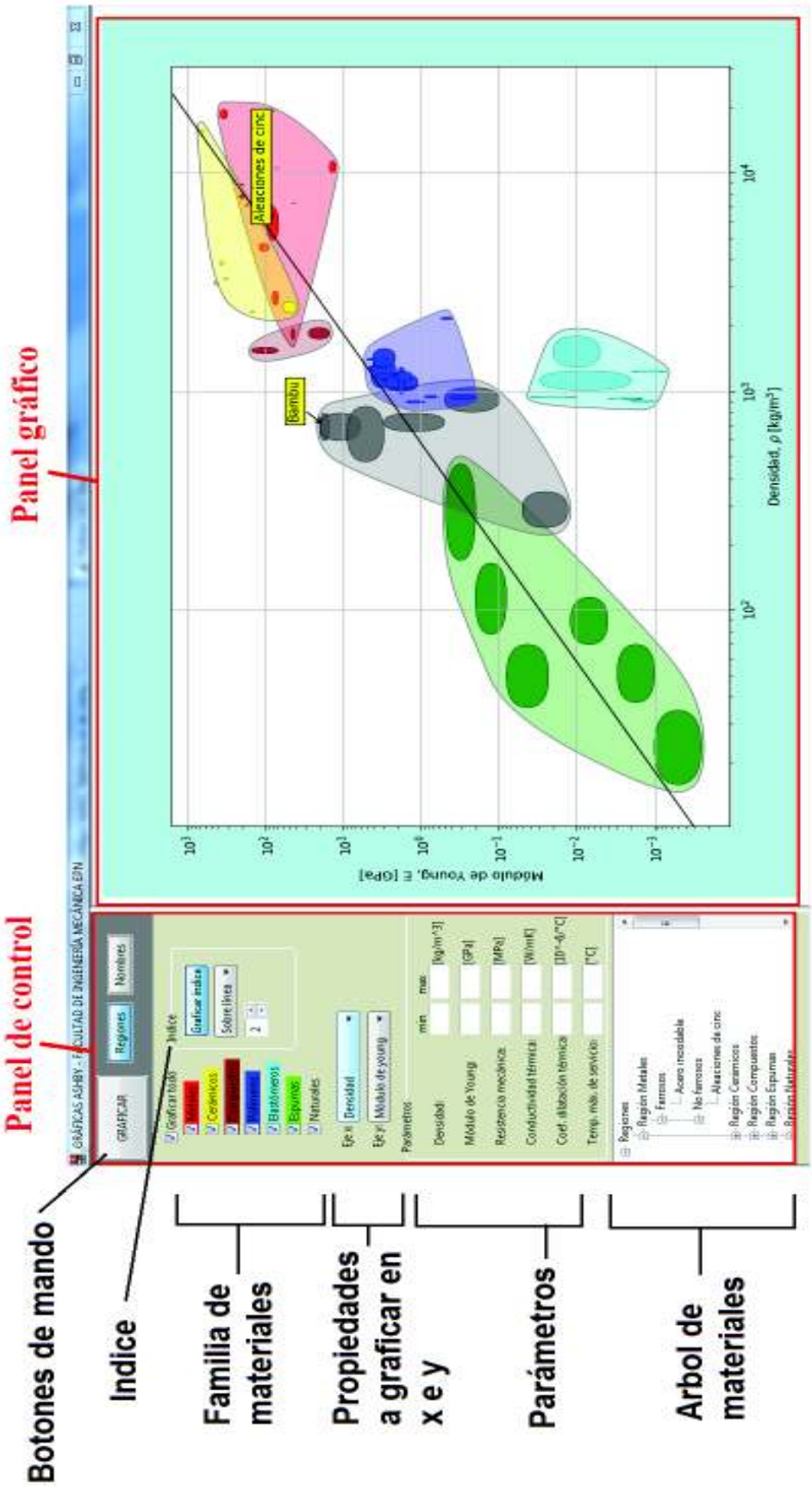

Figura 2.13. Interfaz gráfica desarrollada para visualizar las gráficas de Ashby. (Fuente: propia)

2.5. Comparación de las gráficas

Dado que la finalidad del código de programación es poder representar mapas de materiales, se realiza una comparación con los mapas de materiales más utilizados y presentados por su creador, los diagramas de Ashby. La comparación es realizada de manera visual por que plantea un error debido a los factores humanos que intervienen, la exigencia individual y el estado de ánimo durante la realización del experimento. Una herramienta que permite disminuir o eliminar estos errores es el software Inkscape. Sus ventajas han sido exploradas y enunciadas en este documento. Estas ventajas permiten que sea utilizado para realizar el análisis comparativo.

Los diagramas obtenidos se exportan como imagen, la misma que es compatible con Inkscape. En la interfaz de trabajo del software se colocan las dos imágenes como se muestra en la figura 3.14. y debido a que son fotografías de diferentes tamaños se procede a realizar una normalización de ambas.

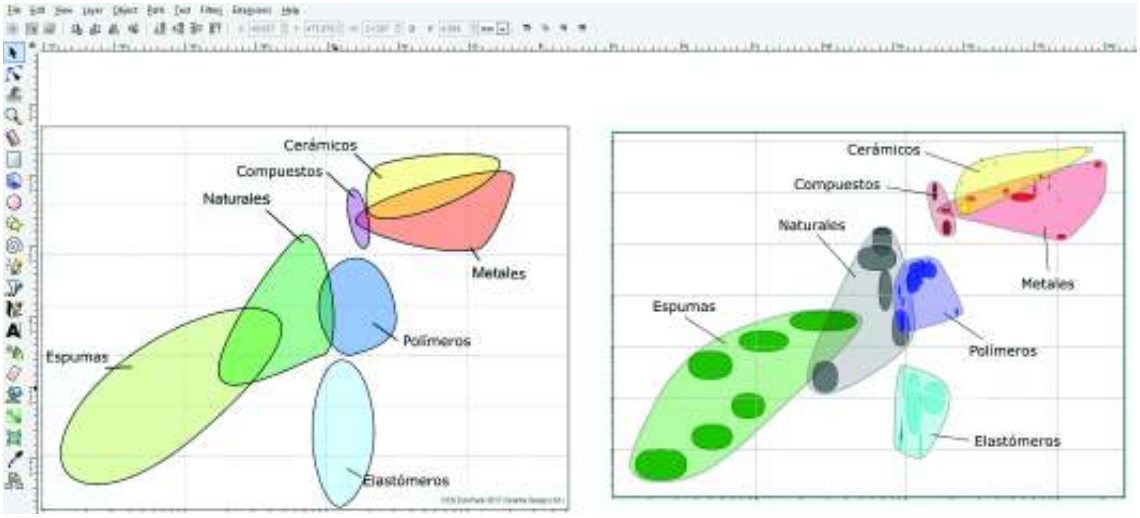

Figura 2.14. Comparación entre diagramas CES vs. Python.
(Fuente: propia)

Esta comparación se realiza de manera visual y con ayuda de las reglas que proporciona Inkscape, realizando así una correcta comparación de escalas. Una ayuda importante para este propósito es la herramienta opacidad, que permite a la imagen obtener un rango de opacidad de cero a cien, donde cero significa una imagen invisible y totalmente transparente mientras que cien es la imagen original. Gracias a esta herramienta es posible superponer las gráficas para verificar que sus escalas coincidan, este proceso está ejemplificado en la figura 3.15. en donde se utilizó una opacidad de cuarenta y cinco para la comparación.

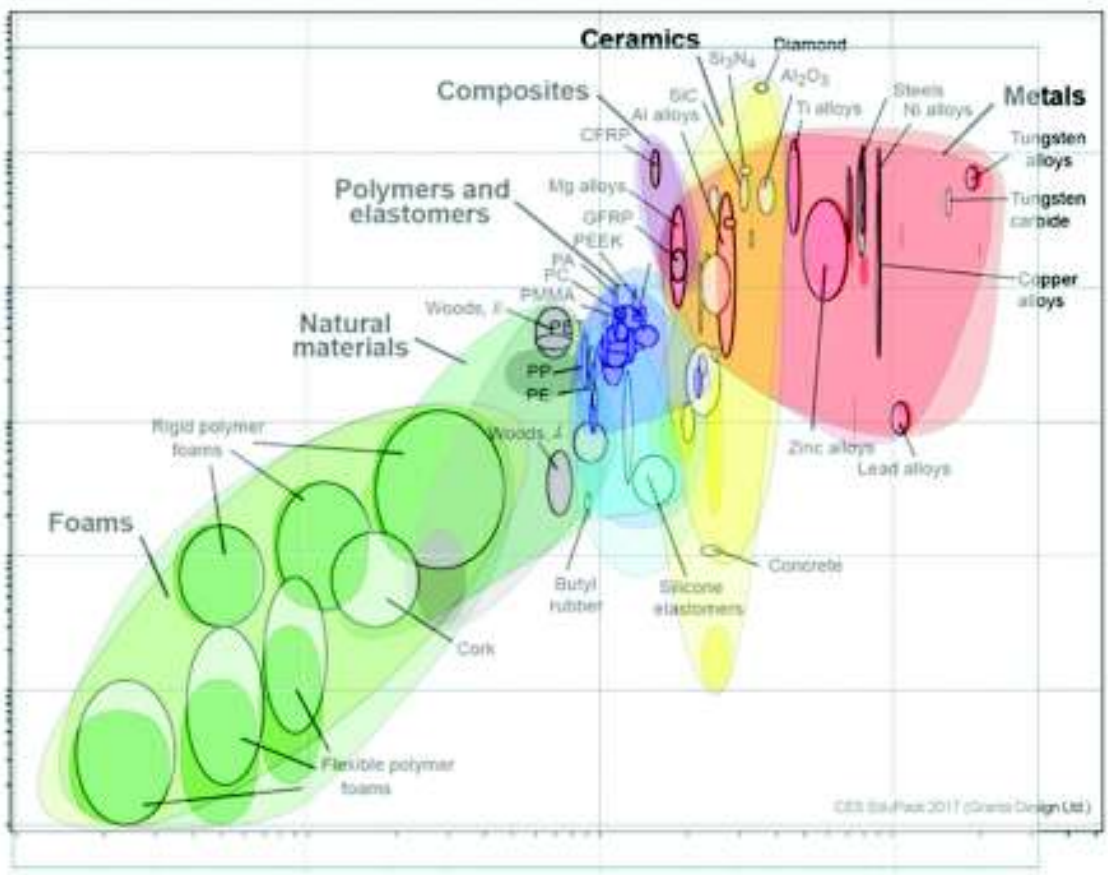

Figura 2.15. Superposición de imágenes, gráfica de resistencia vs. densidad.
(Fuente: propia)

En la figura 3.15. se puede apreciar que las imágenes de las gráficas no necesitan ser exactamente del mismo tamaño, ya que la escala logarítmica proporcionada por cada uno de los programas tiene una ligera variación. Las variaciones más importantes se enuncian a continuación:

El código de Python siempre empieza desde cero mientras que el CES omite esto para representar en su mayoría solo el rango donde se encuentran los materiales, pero ocurre lo contrario en los rangos superiores; Python ubica el techo de los rangos casi al mismo nivel que el rango máximo de los materiales mientras que el CES deja libre y presenta un poco más de la escala pese a que no exista materiales en esa zona. Cabe recalcar que ninguno de los programas está configurado para completar una escala exacta, es decir, que finalice en el orden de 10^n para cualquiera de sus ejes. Controlar estos detalles permite que los resultados sean verídicos ya que no se utiliza una metodología analítica de ninguna índole para la comparación.

Cuando se tiene las gráficas superpuestas de manera correcta se procede a separarlas y volver a su opacidad original, esto con el fin de vectorizar cada una de ellas. El término

vectorizar significa convertir imágenes que estén construidas por pixeles en imágenes formadas por vectores. La vectorización es importante ya que permite conocer el área de cada región de manera automática, cabe recalcar que no se vectoriza toda la imagen sino los puntos de interés, en este caso, las regiones formadas por las envolventes. Para realizar la vectorización se utiliza la herramienta curva de Bézier, la misma que permite redibujar las regiones originales tal como se representa en la figura 2.16.

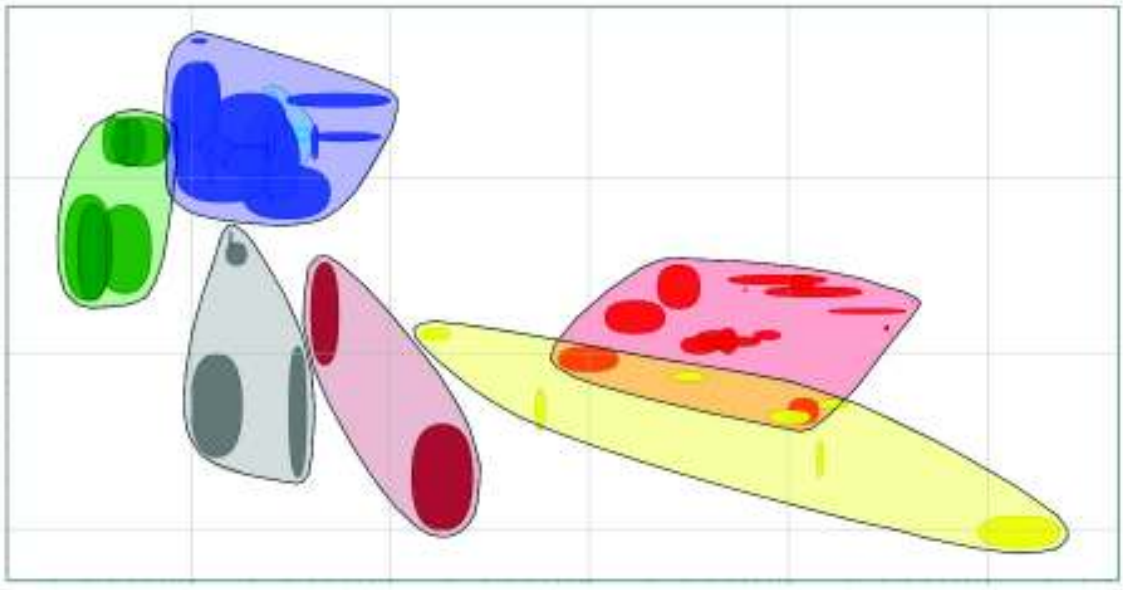

Figura 2.16. Regiones vectorizadas de la gráfica coeficiente de dilatación térmica vs. temperatura máxima de servicio.
(Fuente: propia)

Una región vectorizada obtiene una gran cantidad de características nuevas, la posibilidad de expandirse ilimitadamente sin disminuir su resolución es la más común pero una más específica es la posibilidad de utilizar herramientas computacionales para su medición, Inkscape ofrece estas herramientas- En la barra de herramientas, en la sección de extensiones, se tiene la opción de visualizar el camino vectorizado, este a su vez de divide en varias acciones específicas, una de ellas es la medición del vector, lo que abre la ventana emergente que se muestra en la figura 2.17.

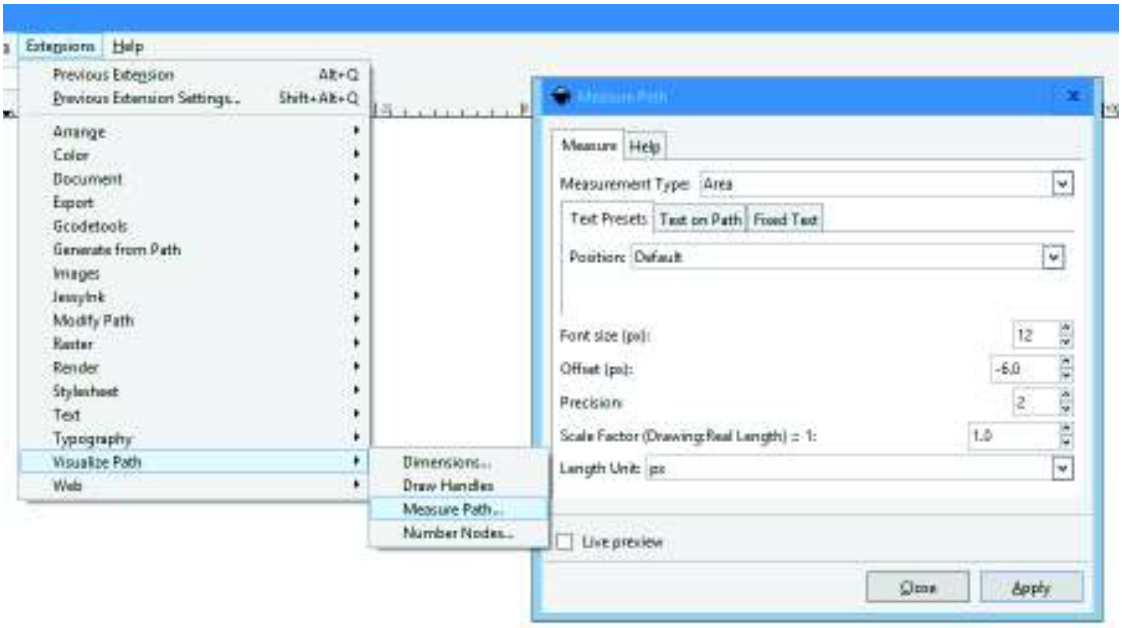

Figura 2.17. Proceso para realizar la medición del vector en Inkscape.
(Fuente: propia)

En la ventana emergente se llenan los parámetros del tipo de medición, ya que, en sí lo que se requiere es la medida total que ocupa el área. El factor de escala se deja en 1 y la unidad de medición utilizada son los pixeles porque se trabaja con herramientas computacionales para imágenes. Los otros campos se dejan por defecto debido a que no influyen en la medición, solo en la visualización del resultado. La medición se genera de forma automática como se muestra en la figura 2.18. y el valor numérico permite determinar el error relativo de cada una de las regiones de manera individual.

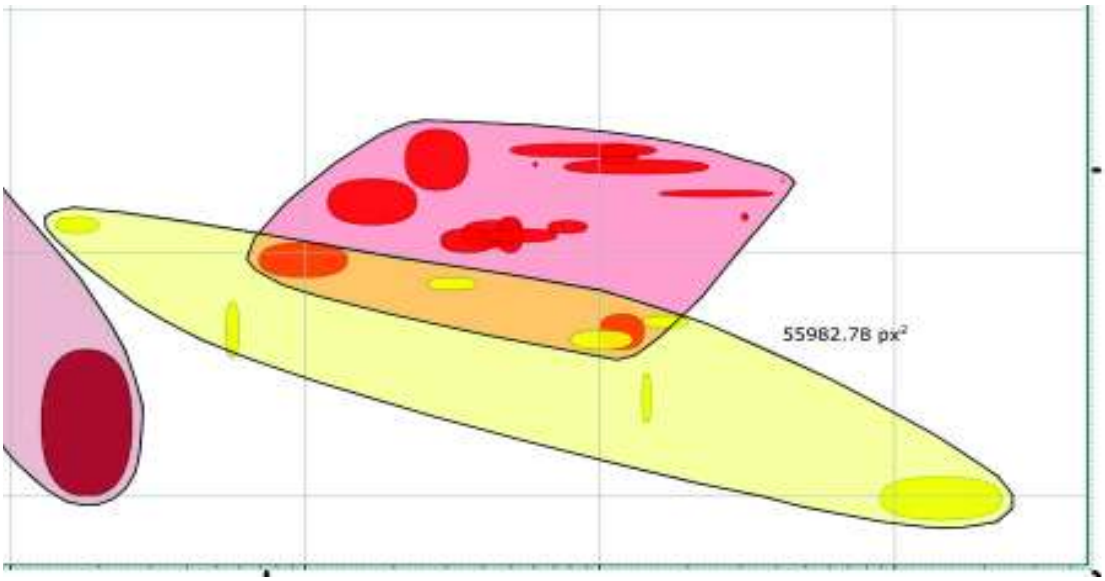

Figura 2.18. Resultado de la medición para la región “cerámicos” del código en Python.
(Fuente: propia)

3. RESULTADOS Y DISCUSIÓN

3.1. Resultados

Los resultados obtenidos son una comparación entre las gráficas generadas por el software CES Edupack 2017 y el código desarrollado en Python.

1. Módulo de Young vs. densidad

Esta es una de las gráficas más importantes para el diseño mecánico ya que abre la posibilidad de reducir los costos al realizar la optimización de la masa del elemento a diseñar, sus regiones están claramente definidas y es la gráfica más conocida de los mapas de materiales.

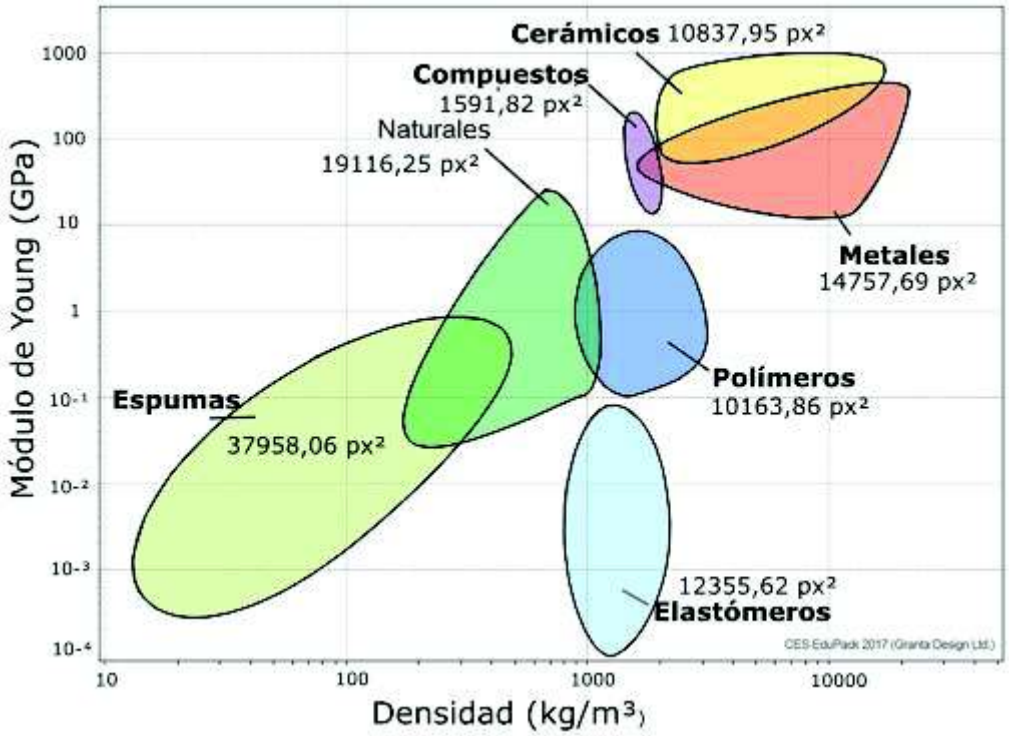

Figura 3.1. Módulo de Young vs. densidad CES.
(Fuente: [3])

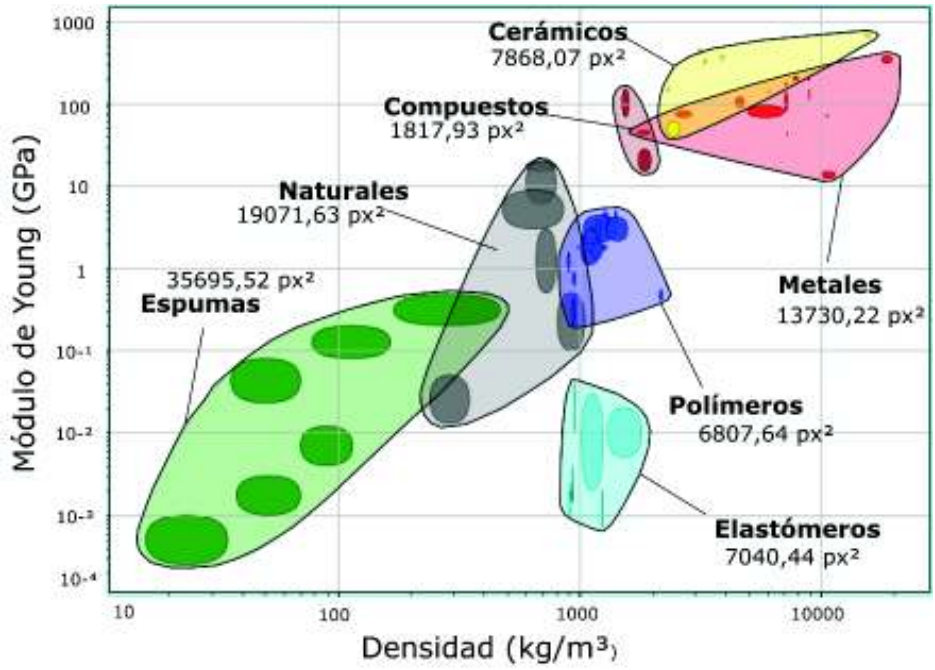

Figura 3.2. Módulo de Young vs. densidad Python.
(Fuente: Propia)

2. Resistencia mecánica vs. densidad

Esta gráfica permite evaluar la razón entre la resistencia y peso del material donde se procura encontrar un material que soporte el esfuerzo requerido pero que sea lo más liviano posible, es decir que tenga alta resistencia con la menor densidad.

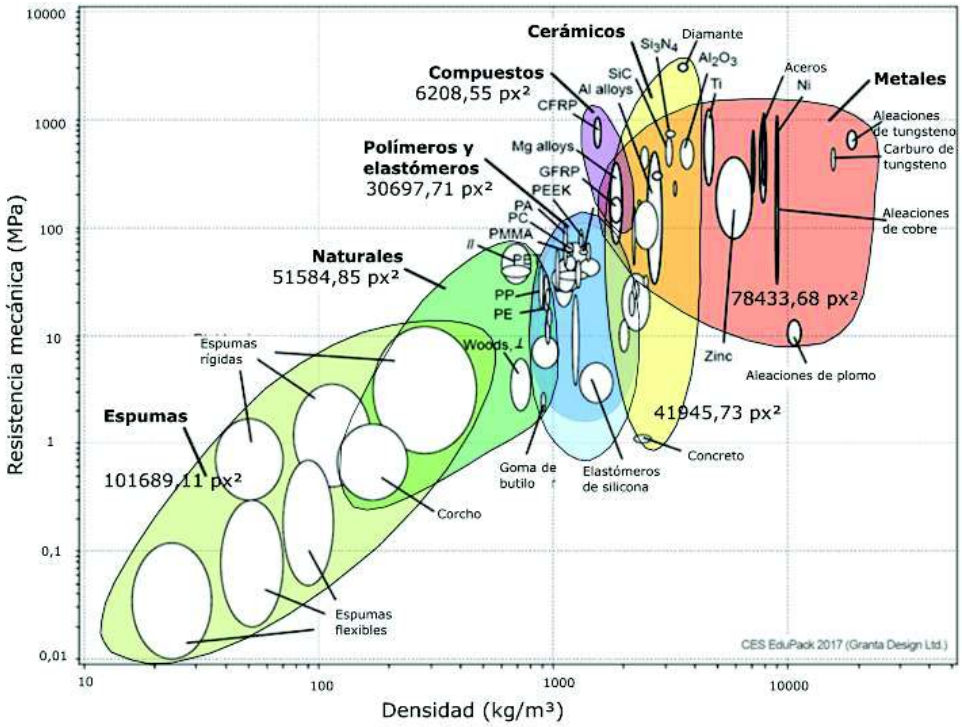

Figura 3.3. Resistencia mecánica vs. Densidad CES.
(Fuente: [3])

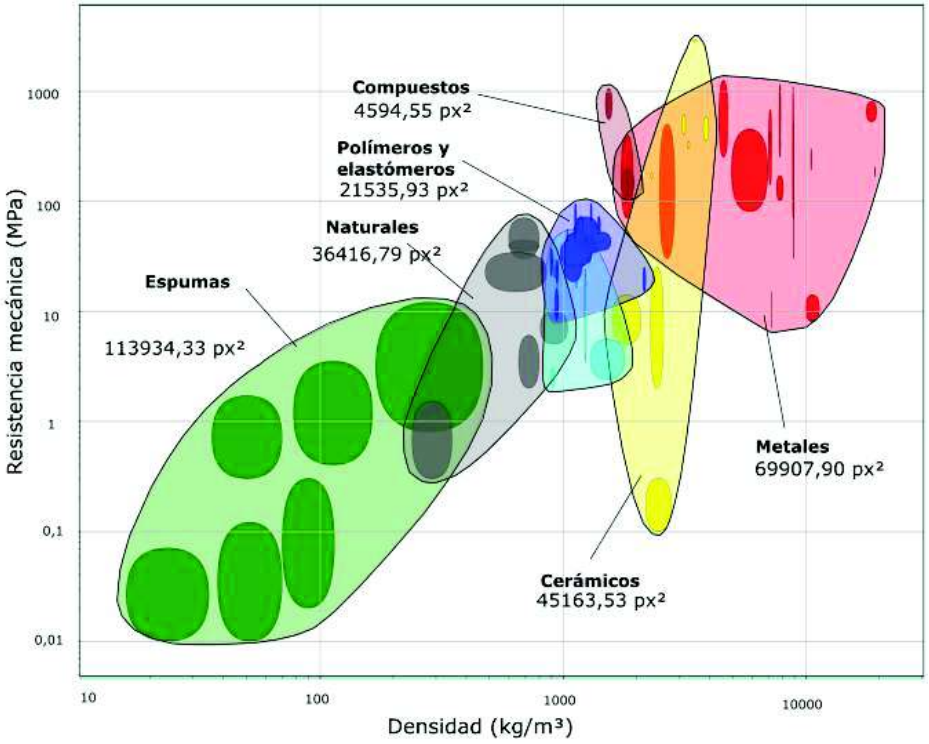

Figura 3.4. Resistencia mecánica vs. densidad Python.
(Fuente: propia)

3. Coeficiente de dilatación térmica vs. conductividad térmica

Esta gráfica representa dos propiedades térmicas requeridas en el diseño mecánico, su importancia radica en la aplicabilidad en el sector industrial como fundición y fabricación de componentes que trabajen a temperaturas extremas.

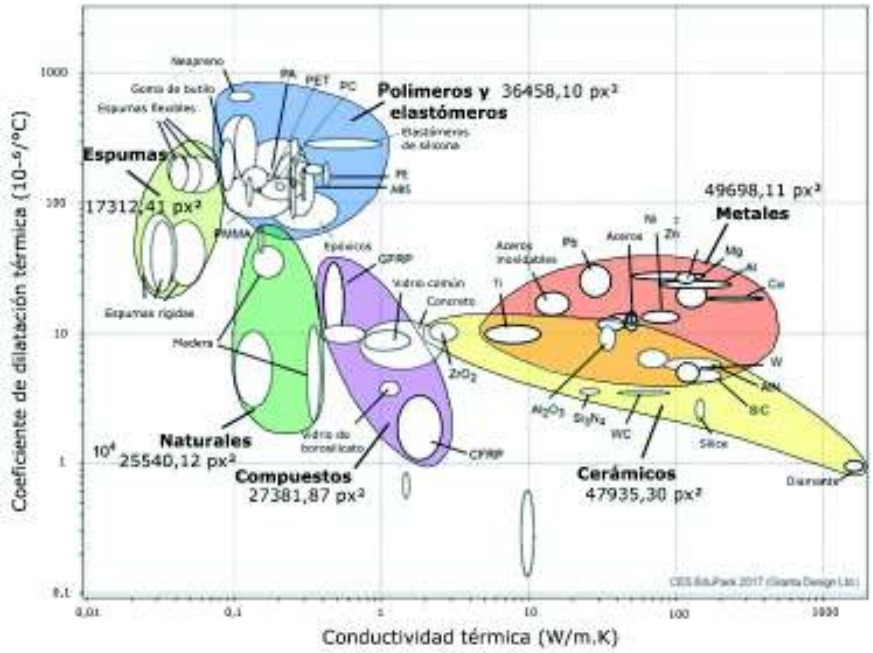

Figura 3.5. Coeficiente de dilatación térmica vs. conductividad térmica CES.
(Fuente: [3])

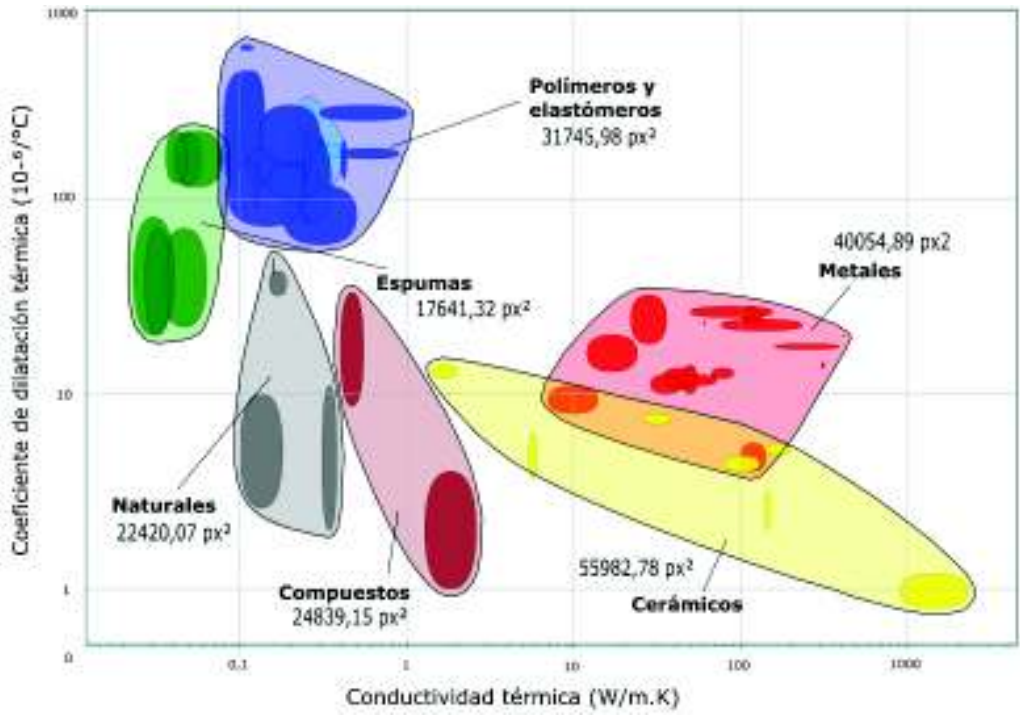

Figura 3.6. Coeficiente de dilatación térmica vs. conductividad térmica Python.
(Fuente: propia)

3.2. Discusión

En la comparación de las gráficas obtenidas del código de programación se hizo un análisis por medio de áreas con las gráficas de Ashby presentadas en el software CES Edupack 2017 y de libre acceso para cualquier persona, dado que las gráficas se encuentran en escala logarítmica abarcan todo el universo de materiales, aunque sugieren cambios más drásticos en los materiales cercanos al origen pese a que su variación sea casi nula.

Se analizó las gráficas de manera comparativa y utilizando el editor gráfico Inkscape, las herramientas que posee permiten de manera sencilla ajustar las escalas de las dos gráficas de manera que sean coincidentes, permite vectorizar las regiones y calcular el área de la misma. Las gráficas se analizaron individualmente comparando su forma y tamaño, lo que permite obtener un error relativo de cada gráfica, después se realiza un estudio de manera global ponderando el peso que tiene cada área en relación a toda la gráfica, de esta manera se obtiene un error absoluto, este error se basa en dos variables, la programación y los valores de los rangos de los materiales, es decir este error puede aumentar o disminuir tanto si se cambia (quitar o añadir) los materiales o se afina las curvas del código en la programación. El propósito es definir si el código es apto o no para recrear los diagramas de Ashby sin tener en cuenta los materiales que se utilicen,

por lo que para la realización de cada gráfica se utilizó solo ciertos materiales, los que se encuentran en los bordes de cada región, es decir los valores extremos de cada familia.

Para que las gráficas están validadas se necesita un error menor al 15%, esto se debe a algunos factores que afectan la recreación de las mismas:

- Los rangos de las propiedades de materiales tienden a sufrir ligeros cambios, al ser CES Edupack un software privativo, no se puede acceder libremente a su base de datos de materiales, lo que genera un error al no tener los valores exactos.
- El número de materiales utilizados en los diagramas realizados por el CES Edupack es imposible de saber debido a la misma condición privativa de licencia pagada que presenta, se realiza el análisis mediante la información disponible de manera gratuita.
- Inkscape tiene herramientas poderosas para la vectorización, pero siempre dependerá del diseñador gráfico la veracidad y aproximación a la realidad con la que sean calculados los valores, este denominado error humano también se toma en consideración al momento de realizar las gráficas.

3.2.1. Análisis de la gráfica de módulo de Young vs. densidad

Al analizar las gráficas de módulo de Young vs. Densidad se obtuvo el error de cada región y el global tal como se muestra en la tabla 3.1.

Tabla 3.1. Comparación entre los valores obtenidos por el CES y el código desarrollado en Python para módulo de Young vs. densidad.

	CES Edupack [px^2]	Python [px^2]	Error relativo [%]	Error absoluto [%]
Cerámicos	10837,95	7868,07	27,40	2,78
Metales	14757,95	13730,22	6,96	0,96
Espumas	37958,06	35695,52	5,96	2,12
Elastómeros	12355,62	7040,44	43,02	4,98
Polímeros	10163,86	6807,64	33,02	3,14
Compuestos	1591,82	1817,93	14,20	0,21
Naturales	19116,25	19071,63	0,23	0,04
Total	106781,51	92031,45		**14,24**

(Fuente: propia)

La gráfica se realizó utilizando 65 materiales presentados en el anexo III. Las envolventes de los materiales naturales y metales tienen un error ponderado de 0,04% y 0,96% respectivamente, esto es debido a que los materiales de estas regiones son conocidos y han sido estudiados desde el año 700 AC, donde empieza la edad de los

metales [17]. En la región de compuestos el error es de 0,21% debido a que solo se utiliza dos clases materiales.

Se nota que la región de los elastómeros es la que tiene el error más alto con 4,98% dado que la gráfica de Ashby considera materiales experimentales que tienen una probabilidad de tener un valor de módulo de Young inferior a 10e-3.

Para las regiones de cerámicos, espumas y polímeros el error está entre 2% y 4%, este error está sujeto a las variaciones en los rangos de valor de las propiedades.

3.2.2. Análisis de la gráfica de resistencia mecánica vs. densidad

La tabla 3.2. presenta el error obtenido al comparar los diagramas obtenidos por ambos softwares.

Tabla 3.2. Comparación entre los valores obtenidos por el CES y el código desarrollado en Python para resistencia mecánica vs. densidad.

	Ashby [px^2]	Python [px^2]	Error relativo [%]	Error absoluto [%]
Cerámicos	41945,73	45163,53	7,67	1,04
Metales	78433,68	69907,9	10,87	2,75
Espumas	101689,11	113934,33	12,04	3,94
Polímeros y elastómeros	30697,71	21535,93	29,85	2,95
Compuestos	6208,55	4594,55	26,00	0,52
Naturales	51584,85	36416,79	29,40	4,88
Total	310559,63	291553,03		**16,08**

(Fuente: propia)

Los materiales utilizados para recrear esta gráfica fueron 66 en total y están ubicados en el anexo III: Esta gráfica es la que presenta el error más alto de las 3 analizadas, de manera global la forma de las regiones es la misma en ambos diagramas, una variación apreciable se puede observar en las regiones de los polímeros y elastómeros, se superponen, es por ello que para poder realizar la comparación se unió ambos en una sola región por motivos de cálculo de áreas, la razón es la misma Ashby considera materiales experimentales y calcula la región total con la probabilidad de que estos se encuentren en esa área, Python, al trabajar con materiales existentes y accesibles tiene un mayor error en esa región, exactamente del 2,95%, los otros errores grandes son de las espumas y los naturales con un 3,94% y 4,88% de error respectivamente, sus regiones comparten la forma con las regiones de Ashby pero su error radica en los rangos de materiales, especialmente en las espumas ya que se encuentran dentro del intervalo de las centésimas y las décimas en la resistencia mecánica lo que implica que

una variación del orden de 0,1 en sus valores generará un error apreciable. Finalmente, los compuestos y los cerámicos ocupan los valores menores de error con un 0,52% y 1,04% respectivamente.

3.2.3. Análisis de la gráfica del coeficiente de expansión térmica vs. la conductividad térmica

Los errores relativo y global de esta gráfica se presentan en la tabla 3.3. detallada a continuación:

Tabla 3.3. Comparación entre los valores obtenidos por el CES y el código desarrollado en Python para el coeficiente de expansión térmica vs. la conductividad térmica.

	Ashby [px^2]	Python [px^2]	Error relativo [%]	Error absoluto [%]
Cerámicos	47935,3	55982,78	16,79	3,94
Metales	49698,11	40054,89	19,40	4,72
Espumas	17312,41	17641,32	1,90	0,16
Polímeros y elastómeros	36458,1	31745,98	12,92	2,31
Compuestos	27381,87	24839,15	9,29	1,24
Naturales	25540,12	22420,07	12,22	1,53
Total	204325,91	192684,19		**13,90**

(Fuente: propia)

Para esta gráfica se utilizaron los 63 materiales detallados en la tabla I.3: en el anexo III. Al realizar la comparación de la forma de las gráficas se puede apreciar que se encuentran en la posición correcta, cada región es similar tanto en el CES como en Python, como una consideración se unificó las regiones de elastómeros y polímeros en una sola, ocupan la misma zona de la gráfica y muchos de sus materiales se superponen por lo que es posible realizar esta consideración. Los dos errores más grandes existentes se encuentran en la región de los cerámicos y de los metales, estos materiales son los más estudiados, sus propiedades son de fácil acceso y no presentan muchos cambios, los materiales utilizados en su mayoría son los mismos pero sus errores son del 3,94% y 4,72% respectivamente, esto se debe a que la región de los cerámicos ocupa un vasto rango en la propiedad de la conductividad térmica, abarcando desde las decenas hasta los miles, por lo que variaciones en el rango de valores generarían un error como el encontrado, por otra parte, los metales poseen el error más grande de esta gráfica, en su mayoría se debe a la afinación de las curvas, el CES posee un gran afinamiento, mientras que en Python se puede realizar de mejor manera en versiones posteriores. Pese a eso el resto de materiales presenta un error bajo por lo

que el código se considera válido para la recreación y presentación de las propiedades de los materiales en gráficos de burbuja.

4. CONCLUSIONES

Si es posible desarrollar un código de programación en lenguaje Python para la recreación de las gráficas de Ashby aplicadas a la selección de materiales. El software libre presenta las herramientas necesarias para poder recrear un software de esta índole cuya precisión sea aceptable.

Las gráficas de Ashby obtenidas con el código son aptas para ser utilizadas en la selección de materiales mediante el método gráfico, debido a su gran similitud con los diagramas de Ashby originales, obteniéndose en general errores menores al 15%.

La forma y tamaño de la envolvente de las regiones depende de los materiales existentes en la base de datos y los rangos de sus propiedades, por lo que genera discrepancias con las envolventes de los diagramas de Ashby, pero su error es bajo como se muestra en el capítulo 3.

La diferencia de áreas se presenta debido a que los límites de los rangos recopilados de la bibliografía tienen ligeras diferencias en comparación a los valores de Ashby, Además, también se aproximó la distancia de separación entre el borde de las burbujas y el borde de la región.

Dependiendo de la velocidad de procesamiento de la computadora donde se ejecute el código se graficarán las envolventes casi instantáneamente. Se puede optimizar la rapidez de obtener las gráficas de Ashby si se tienen almacenados los valores en escala logarítmica. Ya que la metodología usada convierte cada valor de escala normal a logarítmica, realiza los cálculos pertinentes y el resultado se vuelve a convertir a escala normal para ser graficado. Todo este proceso se volverá más demoroso conforme aumente la cantidad de materiales en la base de datos.

La interfaz desarrollada permite obtener de forma sencilla los materiales con las mejores propiedades, pudiendo usar 6 parámetros de propiedades y un parámetro adicional de rendimiento llamado índice.

El código fue estructurado en varias funciones, lo cual facilita el uso y entendimiento del mismo para que otra persona que tenga conocimiento medio de Python pueda modificar

el código a conveniencia. Además, en el código se incluyen descripciones cortas del proceso de cada bucle, variables de entrada, variables de salida y sus funciones, facilitando aún más la compresión del mismo.

Debido a que Ashby no agrupa siempre los materiales en familias completas, se los agrupó en regiones, dentro de las cuales puede haber más de una subfamilia de materiales. Adicionalmente, si se desea cambiar un material de una región a otra, basta con cambiar el campo de región en la base de datos ofreciendo de esta forma versatilidad al modificar fácilmente las regiones.

Para graficar las envolventes se desarrolló una metodología basada en geometría analítica, inscribiendo la región dentro de un polígono irregular para así obtener siempre una forma convexa sin importar la cantidad de materiales usados ni su posición dentro de la gráfica. Además, usar una región inscrita en un polígono permitió utilizar el método CURVE3 disponible en la librería de Matplotlib, controlando de mejor manera la forma de la envolvente y así reducir el error del área de la región dibujada.

Referencias Bibliográficas

[1] M. F. Ashby, *Materials Selection in Mechanical Design*. 3rd Ed. Oxford: Elsevier, 2005.

[2] H. A. González and D. H. Mesa, "La importancia del método en la selección de materiales", *Scientia et Technica*, vol. 1, pp. 175-180, May 2004.

[3] Granta (2018, April 26), What is CES EduPack, [Online]. Available: http://www.grantadesign.com/education/edupack/index.htm.

[4] Granta (2018, April 26), CES Selector product options, [Online]. Available: http://www.grantadesign.com/products/ces/overview.htm.

[5] M. A. Guitiérrez et al., *Stepping into Virtual Reality*. 1st Ed. Switzerland: Springer, 2007.

[6] R. Chrobak and M. L. Benegas, "Herramientas computacionales y el aprendizaje significativo" in Technology Proc. of the First Int. Conference on Concept Mapping, Pamplona, Spain, 2004.

[7] V. H. Hidalgo, "Exergy Analysis of Atmospheric Vortex Engine for Chimney Cooling Tower with Flue Gas Discharge", M.S. thesis, China University of Mining and Technology, Xuzhou, China, 2012.

[8] V. Granja, "Modelación y análisis de la cinemática directa e inversa del manipulador Stanford de seis grados de libertad", bachelor thesis, Escuela Politécnica Nacional, Quito, Ecuador, 2014.

[9] V. H. Hidalgo, "Numerical study on unsteady cavitating flow and erosion based on homogeneous mixtura assumption", Ph. D. Dissertation, Thsinghua University, Beijing, China, 2016.

[10] F. Llulluna, "Procesamiento de imágenes mediante software libre Python para el análisis metalográfico en aceros de bajo contenido de carbono", bachelor thesis, Escuela Politécnica Nacional, Quito, Ecuador, 2014.

[11] A. Silva, "Desarrollo del código de programación para procesamiento de imágenes aplicada en fundiciones nodulares", bachelor thesis, Escuela Politécnica Nacional, Quito, Ecuador, 2017.

[12] R. G. Budynas and J. K Nisbett, *Shigley's Mechanical Engineering Design*. 9th ED. New York: Mc-Graw Hill, 2008.

[13] G. K. Vijayaraghavan and S. vishnupriyan*, Design of Machine Elements*. 4th ED. Tamil Nadu: Lakshmi, 2009.

[14] R. E. Hummel, *Understanding material science.* 2nd ED. New York: Springer, 2004.

[15] D. R. Jones and M. F. Ashby, *Engineering Materials 1*. 2nd ED. Oxford: Elsevier, 1996.

[16] R. Uribe, "Investigaciones de materias primas minerales no metálicas en el Ecuador", *Revista Politécnica*, vol. 36, no. 3, pp. 34-44, September 2015.

[17] V. H. Guerrero et al., *Nuevos Materiales: Aplicaciones estructurales e industriales.* 1st ED. Quito: imprefepp, 2011.

[18] V. Guerrero, "Propiedades mecánicas de compuestos biodegradables elaborados a base de ácido poliláctico reforzados con fibras de abacá", *Revista Politécnica*, vol. 33, no. 2, pp. 88-98, January, 2014.

[19] A. Marzal and I. Gracia, *Introducción a la programación con Python*. 1st ED. España: Universitat Jaume I, 2014.

[20] Python (2018, April 28), Python, [Online]. Available: https://www.python.org/.

[21] Inkspace (2018, May 02), Inkscape, [Online]. Available: https://inkscape.org/es/.

[22] C. Gémy, *Inskcape efficace: Réussir ses dessins vectoriels*. 1st ED. Paris: Eyrolles, 2009.

[23] Libre Office (2018, May 02), ¿Qué es LibreOffice?, [Online]. Available: https://es.libreoffice.org/.

[24] W. D. Callister, *Fundamentals of Materials Science and Engineering*. 4th ED. Salt Lake City: Wiley, 2011.

[25] N. P. Cheremisinoff, *Materials selection deskbook*. 1st ED. New Jersey: Noyes, 1996.

[26] F. Incropera et al., *Fundamentos de transferencia de calor y masa*. 7th ED. Jefferson City: Wiley, 2011.

[27] J. Kiusalaas, *Numerical Methods in Engineering with Python*. 1st ED. Cambridge: Camribge University Press, 2005.

[28] Matplotlib (2018, May 12), Matplotlib home, [Online]. Available: https://matplotlib.org/.

[29] NymPy (2018, May 12), NumPy, [Online]. Available: http://www.numpy.org/.

[30] Welcome to wxPython! (2018, May 16), wxPython The GUI toolkit for Python, [Online]. Available: https://www.wxpython.org/pages/overview/.

ANEXO I.

CÓDIGO GENERADO EN EL LENGUAJE DE PROGRAMACIÓN PYTHON

```
"""
ECUADOR
Quito, junio 2018

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGNIERIA MECÁNICA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECÁNICO

Tesistas:
    Chiriboga del Valle Cristhian Leonardo
    Obando Martínez Esteban Andrés

Director:
    Ing. Hidalgo Díaz Víctor Hugo, DSc

Co-Director:
    Ing. Granja Ramírez Mario Germán, MSc
"""

import wx
import numpy as np

import matplotlib
matplotlib.use('WXAgg')
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg
from matplotlib.axes import Subplot
from matplotlib.figure import Figure
import matplotlib.pyplot as plt

'lectura de datos y grafico de rangos'
import matplotlib.patches as patches
import xlrd as xl

'gráfico de regiones'
from matplotlib.path import Path
from scipy.spatial import ConvexHull

class GUI_ASHBY(wx.Frame):
    def __init__(self, parent, title):
        super(GUI_ASHBY, self).__init__(parent, title=title, size=(800,600))
        self.SetBackgroundColour('white')   #es necesario fondo blanco para
que no se vea cuando se carga la ventana
        self.tabla=self.importacion_datos() #Importación de datos (7
regiones) desde la base al iniciar la interfaz
        self.Maximize()
        self.InitUI()
        icono=wx.Icon("EPNi.ico")   #guarda el incono en una variable
        self.SetIcon(icono) #agrega el icono a la ventana de la interfaz
        self.Show() #muestra la interfaz en la pantalla

    def enter_axes(self, event):
        print ('dentro')
```

```
self.cid=self.fig.canvas.mpl_connect('button_press_event',self.Graficar_recta
Indice)    #conecta el evento clic sobre la figura
        self.clic="in"

    def leave_axes(self, event):
        print ('fuera')
        self.clic="out"


    def InitUI(self):
        self.splitter = wx.SplitterWindow(self, -1)  #divide la ventana en
dos secciones
        'Crea el panel donde estarán los controles'
        self.panel_controles = wx.Panel(self.splitter, -1, size=(20, -1),
style=wx.BORDER_SUNKEN)
        self.panel_controles.SetBackgroundColour(wx.WHITE)
        'Crea el panel para graficar los diagramas de Ashby'
        self.panel_grafico = wx.Panel(self.splitter, -1,
style=wx.BORDER_SUNKEN)
        self.panel_grafico.SetBackgroundColour('white')
        'posiciona los paneles dentro de las secciones de la ventana'
        self.splitter.SplitVertically(self.panel_controles,
self.panel_grafico)
        self.splitter.SetMinimumPaneSize(300)
        self.splitter.SetSashPosition(1,300)

        #===============MODIFICACIÓN DE "PANEL_GRAFICO"=============
        box_grafico=wx.BoxSizer(wx.VERTICAL)
        'Creción del área donde se dibujarán los diagramas de Ashby'
        self.fig = Figure(facecolor='PowderBlue')
        self.canvas = FigureCanvasWxAgg(self.panel_grafico, -1, self.fig)
        self.ax = self.fig.add_axes([0.08,0.1,0.86,0.8])
        self.ax.autoscale(True)
        self.ax.semilogx()
        self.ax.semilogy()
        self.ax.grid(True)

        box_grafico.Add(self.canvas,1, wx.EXPAND| wx.ALL, 10 )  #configura la
figura para que se expanda con la ventana
        self.panel_grafico.SetSizer(box_grafico)    #agrega la figura al
panel_grafico

self.cid=self.fig.canvas.mpl_connect('button_press_event',self.Graficar_recta
Indice)    #conecta el evento clic sobre la figura
        self.fig.canvas.mpl_connect('axes_enter_event',self.enter_axes)
#conecta el evento clic sobre la figura
        self.fig.canvas.mpl_connect('axes_leave_event',self.leave_axes)
#conecta el evento clic sobre la figura

        #===================CREACIÓN DE PANELES===================
        #   PANEL OPCIONES
        #pl_opciones: panel que contiene 2 toggleButtons(regiones y nombres)
y un 1 botón (Graficar)
        pl_opciones=wx.Panel(self.panel_controles,-1,size=(-1,60))
        pl_opciones.SetBackgroundColour('grey')

        #   PANEL MATERIALES
```

```
        #pl_materiales: contiene las opciones para graficar la familia,
propiedades, parámetrose y selección de índice
        pl_materiales=wx.Panel(self.panel_controles,-1,size=(-1,450))
        pl_materiales.SetBackgroundColour('wheat')

        #   PANEL ARBOL DE MATERIALES
        #pl_arbol: contiene la lista de materiales que cumplen con los
parámetros dados
        pl_arbol=wx.Panel(self.panel_controles,-1,size=(-1,60))
        pl_arbol.SetBackgroundColour('wheat')

        #=================CREACIÓN DE OBJETOS DENTRO DE CADA
PANEL==============
        #   **SOBRE PANEL OPCIONES**

        'SECCIÓN: Botones de mando'
            #ToggleButtons
        self.tb_regiones=wx.ToggleButton(pl_opciones,-
1,label='Regiones',pos=(120,15),size=(70,25))
        self.tb_nombres=wx.ToggleButton(pl_opciones,-
1,label='Nombres',pos=(200,15),size=(70,25))
            #Button
        self.b_graficar=wx.Button(pl_opciones,-
1,label='GRAFICAR',pos=(5,5),size=(100,50))

        #   **SOBRE PANEL MATERIALES**

        'SECCIÓN: Selección de índice'
            #ToggleButton
        self.tb_indice=wx.ToggleButton(pl_materiales,-1,label='Graficar
índice',pos=(150,30),size=(85,25))
            #StaticBox
        wx.StaticBox(pl_materiales, -1, 'Indice', pos=(135, 10), size=(120,
130))
            #SpinControl
        self.sc_indice=wx.SpinCtrl(pl_materiales,-
1,'',pos=(150,90),size=(40,-1))
        self.sc_indice.SetRange(1,3)
            #ComboBox
        lista_filtro=['Superior','Inferior','Sobre línea']
        self.cmbo_filtro=wx.ComboBox(pl_materiales, -1, pos=(150, 60),
size=(85, 25),
                                     choices=lista_filtro,
                                     style=wx.CB_READONLY)

        'SECCIÓN: Selección de familias'
            #ChecBox
        self.cb_todo=wx.CheckBox(pl_materiales,id=4,label='Graficar
todo',pos=(25,10))

self.cb_metales=wx.CheckBox(pl_materiales,id=5,label='Metales',pos=(35,30))

self.cb_ceramicos=wx.CheckBox(pl_materiales,id=6,label='Cerámicos',pos=(35,50
))

self.cb_compuestos=wx.CheckBox(pl_materiales,id=7,label='Compuestos',pos=(35,
70))
```

```
self.cb_polimeros=wx.CheckBox(pl_materiales,id=8,label='Polímeros',pos=(35,90
))

self.cb_elastomeros=wx.CheckBox(pl_materiales,id=9,label='Elastómeros',pos=(3
5,110))

self.cb_espumas=wx.CheckBox(pl_materiales,id=10,label='Espumas',pos=(35,130))

self.cb_naturales=wx.CheckBox(pl_materiales,id=11,label='Naturales',pos=(35,1
50))
            #color de fondo
        self.cb_metales.SetBackgroundColour('red')
        self.cb_ceramicos.SetBackgroundColour('yellow')
        self.cb_compuestos.SetBackgroundColour('brown')
        self.cb_polimeros.SetBackgroundColour('blue')
        self.cb_elastomeros.SetBackgroundColour('cyan')
        self.cb_espumas.SetBackgroundColour('green')
        self.cb_naturales.SetBackgroundColour('DimGrey')

        'SECCIÓN: Selección de propieadades'
            #ComboBox
        lista_materiales=['Módulo de young','Densidad','Resistencia
mecánica',
                          'Conductividad térmica','Coef. dilatación térmica',
                          'Temp. máx. de servicio']
        self.cmbo_prop_x=wx.ComboBox(pl_materiales, -1, pos=(50, 180),
size=(130, -1),
                                     choices=lista_materiales,
                                     style=wx.CB_READONLY)
        self.cmbo_prop_y=wx.ComboBox(pl_materiales, -1, pos=(50, 210),
size=(130, -1),
                                     choices=lista_materiales,
                                     style=wx.CB_READONLY)
            #StaticText
        self.lb_ejex=wx.StaticText(pl_materiales, -1, 'Eje x:',pos=(20, 182))
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Eje y:',pos=(20, 212))

        'SECCIÓN: Selección de parámetros'
            #StaticBox
        wx.StaticBox(pl_materiales, -1, 'Parámetros', (5, 240), size=(290,
205))
            #StaticText
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Densidad:',pos=(20,
270))
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Módulo de
Young:',pos=(20, 300))
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Resistencia
mecánica:',pos=(20, 330))
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Conductividad
térmica:',pos=(20, 360))
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Coef. dilatación
térmica:',pos=(20, 390))
        self.lb_ejey=wx.StaticText(pl_materiales, -1, 'Temp. máx. de
servicio:',pos=(20, 420))
        wx.StaticText(pl_materiales, -1, 'min',pos=(155, 255))
        wx.StaticText(pl_materiales, -1, 'max',pos=(200, 255))
            #unidades
        wx.StaticText(pl_materiales, -1, '[kg/m^3]',pos=(235, 270))
```

```
        wx.StaticText(pl_materiales, -1, '[GPa]',pos=(235, 300))
        wx.StaticText(pl_materiales, -1, '[MPa]',pos=(235, 330))
        wx.StaticText(pl_materiales, -1, '[W/mK]',pos=(235, 360))
        wx.StaticText(pl_materiales, -1, '[10^-6/°C]',pos=(235, 390))
        wx.StaticText(pl_materiales, -1, '[°C]',pos=(235, 420))
            #TextBox
                #valores mínimos
        self.txt_densidad_min = wx.TextCtrl(pl_materiales, -1,pos=(150,
270),size=(40,20))
        self.txt_young_min = wx.TextCtrl(pl_materiales, -1,pos=(150,
300),size=(40,20))
        self.txt_rmecanica_min = wx.TextCtrl(pl_materiales, -1,pos=(150,
330),size=(40,20))
        self.txt_ctermica_min = wx.TextCtrl(pl_materiales, -1,pos=(150,
360),size=(40,20))
        self.txt_cdilatacion_min = wx.TextCtrl(pl_materiales, -1,pos=(150,
390),size=(40,20))
        self.txt_temp_serv_min = wx.TextCtrl(pl_materiales, -1,pos=(150,
420),size=(40,20))
                #valores máximos
        self.txt_densidad_max = wx.TextCtrl(pl_materiales, -1,pos=(195,
270),size=(40,20))
        self.txt_young_max = wx.TextCtrl(pl_materiales, -1,pos=(195,
300),size=(40,20))
        self.txt_rmecanica_max = wx.TextCtrl(pl_materiales, -1,pos=(195,
330),size=(40,20))
        self.txt_ctermica_max = wx.TextCtrl(pl_materiales, -1,pos=(195,
360),size=(40,20))
        self.txt_cdilatacion_max = wx.TextCtrl(pl_materiales, -1,pos=(195,
390),size=(40,20))
        self.txt_temp_serv_max = wx.TextCtrl(pl_materiales, -1,pos=(195,
420),size=(40,20))

        box_materiales=wx.BoxSizer(wx.VERTICAL)
        self.panel_controles.SetSizer(box_materiales)

        #   **SOBRE PANEL ARBOL DE MATERIALES**
        'SECCIÓN: árbol de materiales'
        self.tree= wx.TreeCtrl(pl_arbol, 1, wx.DefaultPosition, (295,180) ,
style=wx.TR_DEFAULT_STYLE)

        #Ordena los paneles de opciones y materiales
        box_controles=wx.BoxSizer(wx.VERTICAL)
        box_controles.Add(pl_opciones,0,wx.EXPAND)
        box_controles.Add(pl_materiales,1,wx.EXPAND)
        box_controles.Add(pl_arbol,2,wx.EXPAND)

        self.panel_controles.SetSizer(box_controles)

        #=======CONDICIONES INICIALES DE LA INTERFAZ=====================
        'SECCIÓN: Selección de familias'
        self.cb_todo.SetValue(True), self.cb_metales.SetValue(True)
        self.cb_ceramicos.SetValue(True)
        self.cb_compuestos.SetValue(True)
        self.cb_polimeros.SetValue(True)
        self.cb_elastomeros.SetValue(True)
        self.cb_espumas.SetValue(True)
        self.cb_naturales.SetValue(True)
        'SECCIÓN: Selección de propieadades'
```

```
        self.cmbo_prop_x.SetSelection(1)
        self.cmbo_prop_y.SetSelection(0)
        'SECCIÓN: Selección de índice'
        self.tb_indice.Enable(False)
        self.sc_indice.SetValue(1)
        self.cmbo_filtro.SetSelection(0)

        #===============PROGRAMACIÓN DE OBTJETOS====================
        'SECCIÓN: Botones de mando'
        #Botón GRAFICAR
        self.b_graficar.Bind(wx.EVT_BUTTON,self.Graficar)
        #Botón NOMBRES
        self.tb_nombres.Bind(wx.EVT_TOGGLEBUTTON,self.Mostrar_nombres)
        'SECCIÓN: Selección de familias'
            #CheckBox
        self.cb_todo.Bind(wx.EVT_CHECKBOX,self.check_all)
        self.cb_metales.Bind(wx.EVT_CHECKBOX,self.some_material)
        self.cb_ceramicos.Bind(wx.EVT_CHECKBOX,self.some_material)
        self.cb_compuestos.Bind(wx.EVT_CHECKBOX,self.some_material)
        self.cb_polimeros.Bind(wx.EVT_CHECKBOX,self.some_material)
        self.cb_elastomeros.Bind(wx.EVT_CHECKBOX,self.some_material)
        self.cb_espumas.Bind(wx.EVT_CHECKBOX,self.some_material)
        self.cb_naturales.Bind(wx.EVT_CHECKBOX,self.some_material)
        'SECCIÓN: árbol de materiales'
        #ÁRBOL DE MATERIALES
        self.tree.Bind(wx.EVT_TREE_ITEM_ACTIVATED,self.Nombre_material)

    def Graficar_rectaIndice(self,event):
        'Grafica la recta del INDICE al hacer clic'
        if self.tb_indice.GetValue()==True:
            if event.inaxes != self.ax.axes: return

            xpos,ypos=event.xdata, event.ydata  # coordenadas del punto donde
se hace click

            px=np.log10(xpos)
            py=np.log10(ypos)

            self.m=self.sc_indice.GetValue() # valor de pendiente
            self.b=-self.m*px+py

            #punto inicial al borde para graficar la recta
            yi_log=self.m*np.log10(self.xmin)+self.b
            yf_log=self.m*np.log10(self.xmax)+self.b

            yi=10**yi_log
            yf=10**yf_log

            'Se debe mantener el orden de las siguientes 5 líneas'
            self.Graficar(event)    #vuelve a graficar (para borrar la recta
anterior)
            self.ax.autoscale(False)    #evita que la escala se distorsione
            self.ax.plot((self.xmin,self.xmax),(yi,yf),'-
',color='black',lw=1)  #agrega la recta a la gráfica
            self.ax.grid(True)
            self.canvas.draw()      #muestra la gráfica

    def Nombre_material(self,event):
```

```
        'Muestra en la gráfica el nombre del material al hacer doble clic
sobre su ítem en el árbol de materiales'
        item = self.tree.GetSelection()
        nombre=self.tree.GetItemText(item)
        #SELECCIÓN DE PROPIEDADES A GRAFICAR
        x_propiedad,y_propiedad,xlabel,ylabel=self.seleccion_propiedad()
        for i in range(len(self.LISTA)):    #i=lista de cada región'
            for j in range(len(self.LISTA[i])): #j=cantidad de materiales de
cada region'
                if self.LISTA[i][j][4]==nombre:

posx=(float(self.LISTA[i][j][x_propiedad+5])+float(self.LISTA[i][j][x_propied
ad+6]))/2

posy=(float(self.LISTA[i][j][y_propiedad+5])+float(self.LISTA[i][j][y_propied
ad+6]))/2

                    self.ax.annotate(nombre,
xy=(posx,posy),xytext=(posx+20,posy+20),bbox={'facecolor':'yellow',
'alpha':1, 'pad':2},
                                     arrowprops=dict(arrowstyle="->"))
        self.canvas.draw()

    def Mostrar_nombres(self,event):
        'Muestra en la gráfica los nombres de todos los materiales
visualizados'
        if self.tb_nombres.GetValue()==True:
            for i in range(len(self.LISTA_VALIDADA)):    #i=lista de cada
región'
                if self.LISTA_VALIDADA[i]!=[]:
                    for j in range(len(self.LISTA_VALIDADA[i])): #j=cantidad
de materiales de cada region'
                        nombre=self.LISTA_VALIDADA[i][j][4]
                        #SELECCIÓN DE PROPIEDADES A GRAFICAR

x_propiedad,y_propiedad,xlabel,ylabel=self.seleccion_propiedad()


posx=(float(self.LISTA_VALIDADA[i][j][x_propiedad+5])+float(self.LISTA_VALIDA
DA[i][j][x_propiedad+6]))/2

posy=(float(self.LISTA_VALIDADA[i][j][y_propiedad+5])+float(self.LISTA_VALIDA
DA[i][j][y_propiedad+6]))/2


self.ax.text(posx,posy,nombre,fontsize=8,bbox={'facecolor':'yellow',
'alpha':1, 'pad':2})
            self.canvas.draw()
        else:
            self.Graficar(event)

    def seleccion_propiedad(self):
        'Selecciona desde la base la propiedad a graficar en x e y'
        #Propiedad en x
        if self.cmbo_prop_x.GetValue()=='Módulo de young':
            x_propiedad=2
            xlabel=r"Módulo de Young, E [GPa]"
        elif self.cmbo_prop_x.GetValue()=='Densidad':
            x_propiedad=0
```

```
            xlabel=r"Densidad, $\rho$ [kg/m$^3$]"
        elif self.cmbo_prop_x.GetValue()=='Resistencia mecánica':
            x_propiedad=4
            xlabel='Resistencia mecánica, [MPa]'
        elif self.cmbo_prop_x.GetValue()=='Conductividad térmica':
            x_propiedad=6
            xlabel='Resistencia térmica, [W/mK]'
        elif self.cmbo_prop_x.GetValue()=='Coef. dilatación térmica':
            x_propiedad=8
            xlabel='Coeficiente de dilatación, [10$^-$$^6$/°C]'
        elif self.cmbo_prop_x.GetValue()=='Temp. máx. de servicio':
            x_propiedad=10
            xlabel='Temperatura máxima de servicio, [°C]'

        #Propiedad en y
        if self.cmbo_prop_y.GetValue()=='Módulo de young':
            y_propiedad=2
            ylabel=r"Módulo de Young, E [GPa]"
        elif self.cmbo_prop_y.GetValue()=='Densidad':
            y_propiedad=0
            ylabel=r"Densidad, $\rho$ [kg/m$^3$]"
        elif self.cmbo_prop_y.GetValue()=='Resistencia mecánica':
            y_propiedad=4
            ylabel='Resistencia mecánica, [MPa]'
        elif self.cmbo_prop_y.GetValue()=='Conductividad térmica':
            y_propiedad=6
            ylabel='Resistencia térmica, [W/mK]'
        elif self.cmbo_prop_y.GetValue()=='Coef. dilatación térmica':
            y_propiedad=8
            ylabel='Coeficiente de dilatación, [10$^-$$^6$/°C]'
        elif self.cmbo_prop_y.GetValue()=='Temp. máx. de servicio':
            y_propiedad=10
            ylabel='Temperatura máxima de servicio, [°C]'

        return x_propiedad,y_propiedad,xlabel,ylabel

    def Graficar(self,event):
        'Grafica los diagramas de Ashby'
        self.ax.cla()   #Limpia el área del gráfico
        self.tree.DeleteAllItems()  #Limpia los ítems en el arbol de
materiales
        self.tb_indice.Enable(True)

        #SELECCIÓN DE PROPIEDADES A GRAFICAR
        x_propiedad,y_propiedad,xlabel,ylabel=self.seleccion_propiedad()

        #LLAMA a la función de VALIDACIÓN DE PARÁMETROS para graficar de
acuerdo a los parámetros definidos por el usuario

self.R1,self.R2,self.R3,self.R4,self.R5,self.R6,self.R7,self.LISTA=self.valid
acion_parametros()

        #Árbol de materiales'
        root = self.tree.AddRoot('Regiones')

        self.LISTA_VALIDADA=[]
#==Condicionales para graficar RANGOS, REGIONES y RECTA DEL ÍNDICE=====
        if self.cb_metales.GetValue()==True:
            color='red'
```

```
            if len(self.R5)!=0:

R5_pts=self.rango_materiales(self.R5,x_propiedad,y_propiedad,color)
#metales
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R5_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R5_pts)/8)     #cantidad de materiales en la
región (cada materiale tiene 8 puntos)
                    i_metales=[]
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R5_pts[pos,1])
                        lim_sup_y=np.log10(R5_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R5_pts[pos+6,0])
#corte en eje y en la esquina superior izquierda del rectángulo del rango
                        b_inf=lim_inf_y-self.m*np.log10(R5_pts[pos+2,0])
#corte en eje y en la esquina inferior derecha del rectángulo del rango

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_metales.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:   #si b de la
recta del indice está entre el b_sup e inf entonces la recta pasa a traves
del rango del material
                                i_metales.append(pos)    #lista con las
posiciones del material en la matriz de la region de materiales
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_metales.append(pos)

                    R5_pts=R5_pts[i_metales, :]

                    Lista_R5=[]
                    for i in range(len(i_metales)):
                        Lista_R5.append(self.LISTA[4][int(i_metales[i]/8)])
                else:
                    R5_pts=self.R5
                    Lista_R5=self.LISTA[4]

                self.LISTA_VALIDADA.append(Lista_R5)

                if Lista_R5!=[]:    #solo si existe materiales en la lista
agrego la categoría al árbol de materiales
                    #Agrego los nombres de materiales al arbol visual
                    R_metales=self.tree.AppendItem(root, 'Región Metales')
                    for i in range(len(R5_pts)):
                        if Lista_R5[i][3]=='Ferrosos':

ferrosos=self.tree.AppendItem(R_metales,'Ferrosos')
                            break
```

```
                    for i in range(len(R5_pts)):
                        if Lista_R5[i][3]=='No ferrosos':
                            no_ferrosos=self.tree.AppendItem(R_metales,'No
ferrosos')
                            break

                    for i in range(len(R5_pts)):
                        if Lista_R5[i][3]=='Ferrosos':
                            self.tree.AppendItem(ferrosos,Lista_R5[i][4])
                        else:
                            self.tree.AppendItem(no_ferrosos,Lista_R5[i][4])

        if self.cb_ceramicos.GetValue()==True:
            color='yellow'
            if len(self.R1)!=0:

R1_pts=self.rango_materiales(self.R1,x_propiedad,y_propiedad,color)
#ceramicos
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R1_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R1_pts)/8)     #cantidad de materiales en la
región (cada materiale tiene 8 puntos)
                    i_ceramicos=[]
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R1_pts[pos,1])
                        lim_sup_y=np.log10(R1_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R1_pts[pos+6,0])
#corte en eje y en la esquina superior izquierda del rectángulo del rango
                        b_inf=lim_inf_y-self.m*np.log10(R1_pts[pos+2,0])
#corte en eje y en la esquina inferior derecha del rectángulo del rango

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_ceramicos.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:   #si b de la
recta del indice está entre el b_sup e inf entonces la recta pasa a traves
del rango del material
                                i_ceramicos.append(pos)    #lista con las
posiciones del material en la matriz de la region de materiales
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_ceramicos.append(pos)

                    R1_pts=R1_pts[i_ceramicos, :]

                    Lista_R1=[]
                    for i in range(len(i_ceramicos)):
                        Lista_R1.append(self.LISTA[0][int(i_ceramicos[i]/8)])
                else:
                    R1_pts=self.R1
```

```
                    Lista_R1=self.LISTA[0]

                self.LISTA_VALIDADA.append(Lista_R1)

                if Lista_R1!=[]:
                    R_ceramicos=self.tree.AppendItem(root, 'Región
Cerámicos')
                    for i in range(len(R1_pts)):
                        self.tree.AppendItem(R_ceramicos,Lista_R1[i][4])

        if self.cb_compuestos.GetValue()==True:
            color='brown'      #corregir color
            if len(self.R2)!=0:

R2_pts=self.rango_materiales(self.R2,x_propiedad,y_propiedad,color)
#compuestos
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R2_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R2_pts)/8)     #cantidad de materiales en la
región (cada materiale tiene 8 puntos)
                    i_compuestos=[]
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R2_pts[pos,1])
                        lim_sup_y=np.log10(R2_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R2_pts[pos+6,0])
#corte en eje y en la esquina superior izquierda del rectángulo del rango
                        b_inf=lim_inf_y-self.m*np.log10(R2_pts[pos+2,0])
#corte en eje y en la esquina inferior derecha del rectángulo del rango

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_compuestos.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:
                                i_compuestos.append(pos)    #lista con las
posiciones del material en la matriz de la region de materiales
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_compuestos.append(pos)

                    R2_pts=R2_pts[i_compuestos, :]

                    Lista_R2=[]
                    for i in range(len(i_compuestos)):

Lista_R2.append(self.LISTA[1][int(i_compuestos[i]/8)])
                else:
                    R2_pts=self.R2
                    Lista_R2=self.LISTA[1]

                self.LISTA_VALIDADA.append(Lista_R2)
```

```
                if Lista_R2!=[]:
                    R_compuestos=self.tree.AppendItem(root, 'Región
Compuestos')
                    comp_hibridos=self.tree.AppendItem(R_compuestos,
'Híbridos')
                    for i in range(len(R2_pts)):
                        self.tree.AppendItem(comp_hibridos,Lista_R2[i][4])

        if self.cb_espumas.GetValue()==True:
            color='green'
            if len(self.R4)!=0:

R4_pts=self.rango_materiales(self.R4,x_propiedad,y_propiedad,color)
#espumas
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R4_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R4_pts)/8)
                    i_espumas=[]
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R4_pts[pos,1])
                        lim_sup_y=np.log10(R4_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R4_pts[pos+6,0])
#corte en eje y en la esquina superior izquierda del rectángulo del rango
                        b_inf=lim_inf_y-self.m*np.log10(R4_pts[pos+2,0])
#corte en eje y en la esquina inferior derecha del rectángulo del rango

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_espumas.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:
                                i_espumas.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_espumas.append(pos)

                    R4_pts=R4_pts[i_espumas, :]

                    Lista_R4=[]
                    for i in range(len(i_espumas)):
                        Lista_R4.append(self.LISTA[3][int(i_espumas[i]/8)])
                else:
                    R4_pts=self.R4
                    Lista_R4=self.LISTA[3]

                self.LISTA_VALIDADA.append(Lista_R4)

                if Lista_R4!=[]:
                    R_espumas=self.tree.AppendItem(root, 'Región Espumas')
```

```
                    espuma_hibridos=self.tree.AppendItem(R_espumas,
'Híbridos')
                    for i in range(len(R4_pts)):
                        self.tree.AppendItem(espuma_hibridos,Lista_R4[i][4])

        if self.cb_naturales.GetValue()==True:
            color='DimGrey'
            if len(self.R6)!=0:

R6_pts=self.rango_materiales(self.R6,x_propiedad,y_propiedad,color)
#naturales
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R6_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R6_pts)/8)     #cantidad de materiales en la
región (cada materiale tiene 8 puntos)
                    i_naturales=[]     #
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R6_pts[pos,1])
                        lim_sup_y=np.log10(R6_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R6_pts[pos+6,0])
                        b_inf=lim_inf_y-self.m*np.log10(R6_pts[pos+2,0])

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_naturales.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:
                                i_naturales.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_naturales.append(pos)

                    R6_pts=R6_pts[i_naturales, :]

                    Lista_R6=[]
                    for i in range(len(i_naturales)):
                        Lista_R6.append(self.LISTA[5][int(i_naturales[i]/8)])
                else:
                    R6_pts=self.R6
                    Lista_R6=self.LISTA[5]

                self.LISTA_VALIDADA.append(Lista_R6)

                if Lista_R6!=[]:    #solo si existe materiales en la lista
agrego la categoría al árbol de materiales
                    R_naturales=self.tree.AppendItem(root, 'Región
Naturales')
                    natural_hibridos=self.tree.AppendItem(R_naturales,
'Híbridos')
                    for i in range(len(R6_pts)):
                        self.tree.AppendItem(natural_hibridos,Lista_R6[i][4])
```

```
        if self.cb_elastomeros.GetValue()==True:
            color='cyan'
            if len(self.R3)!=0:

R3_pts=self.rango_materiales(self.R3,x_propiedad,y_propiedad,color)
#elastomeros
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R3_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R3_pts)/8)
                    i_elastomeros=[]
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R3_pts[pos,1])
                        lim_sup_y=np.log10(R3_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R3_pts[pos+6,0])
#corte en eje y en la esquina superior izquierda del rectángulo del rango
                        b_inf=lim_inf_y-self.m*np.log10(R3_pts[pos+2,0])
#corte en eje y en la esquina inferior derecha del rectángulo del rango

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_elastomeros.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:
                                i_elastomeros.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_elastomeros.append(pos)

                    R3_pts=R3_pts[i_elastomeros, :]

                    Lista_R3=[]
                    for i in range(len(i_elastomeros)):

Lista_R3.append(self.LISTA[2][int(i_elastomeros[i]/8)])
                else:
                    R3_pts=self.R3
                    Lista_R3=self.LISTA[2]

                self.LISTA_VALIDADA.append(Lista_R3)

                if Lista_R3!=[]:
                    R_elastomeros=self.tree.AppendItem(root, 'Región
Elastómeros')
                    for i in range(len(R3_pts)):
                        self.tree.AppendItem(R_elastomeros,Lista_R3[i][4])

        if self.cb_polimeros.GetValue()==True:
            color='blue'
            if len(self.R7)!=0:
```

```
R7_pts=self.rango_materiales(self.R7,x_propiedad,y_propiedad,color)
#polimeros
                if self.tb_regiones.GetValue()==True:
                    self.grafico_regiones(R7_pts,color)

                if self.tb_indice.GetValue()==True and self.clic=="in":
                    n=int(len(R7_pts)/8)
                    i_polimeros=[]
                    for i in range(n):
                        pos=i*8

                        'rango en eje y'
                        lim_inf_y=np.log10(R7_pts[pos,1])
                        lim_sup_y=np.log10(R7_pts[pos+4,1])

                        b_sup=lim_sup_y-self.m*np.log10(R7_pts[pos+6,0])
                        b_inf=lim_inf_y-self.m*np.log10(R7_pts[pos+2,0])

                        #filtro de materiales dependiendo el filtro superior,
inferior o sobre línea
                        if self.cmbo_filtro.GetValue()=='Superior':
                            if self.b<=b_sup:
                                i_polimeros.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Sobre línea':
                            if self.b<=b_sup and self.b>=b_inf:
                                i_polimeros.append(pos)
                        elif self.cmbo_filtro.GetValue()=='Inferior':
                            if self.b>=b_sup:
                                i_polimeros.append(pos)

                    R7_pts=R7_pts[i_polimeros, :]

                    Lista_R7=[]
                    for i in range(len(i_polimeros)):
                        Lista_R7.append(self.LISTA[6][int(i_polimeros[i]/8)])
                else:
                    R7_pts=self.R7
                    Lista_R7=self.LISTA[6]

                self.LISTA_VALIDADA.append(Lista_R7)

                if Lista_R7!=[]:
                    R_polimeros=self.tree.AppendItem(root, 'Región
Polímeros')
                    for i in range(len(R7_pts)):
                        if Lista_R7[i][3]=='Termoestables':

termoestables=self.tree.AppendItem(R_polimeros,'Termoestables')
                            break

                    for i in range(len(R7_pts)):
                        if Lista_R7[i][3]=='Termoplasticos':

termoplasticos=self.tree.AppendItem(R_polimeros,'Termoplasticos')
                            break

                    for i in range(len(R7_pts)):
                        if Lista_R7[i][3]=='Termoestables':
```

```
self.tree.AppendItem(termoestables,Lista_R7[i][4])
                        else:

self.tree.AppendItem(termoplasticos,Lista_R7[i][4])

        if self.tb_nombres.GetValue()==True:
            for i in range(len(self.LISTA_VALIDADA)):    #i=lista de cada
región'
                if self.LISTA_VALIDADA[i]!=[]:
                    for j in range(len(self.LISTA_VALIDADA[i])): #j=cantidad
de materiales de cada region'
                        nombre=self.LISTA_VALIDADA[i][j][4]
                        #SELECCIÓN DE PROPIEDADES A GRAFICAR

x_propiedad,y_propiedad,xlabel,ylabel=self.seleccion_propiedad()


posx=(float(self.LISTA_VALIDADA[i][j][x_propiedad+5])+float(self.LISTA_VALIDA
DA[i][j][x_propiedad+6]))/2

posy=(float(self.LISTA_VALIDADA[i][j][y_propiedad+5])+float(self.LISTA_VALIDA
DA[i][j][y_propiedad+6]))/2


self.ax.text(posx,posy,nombre,fontsize=8,bbox={'facecolor':'yellow',
'alpha':1, 'pad':2})

        self.tree.Expand(root)   #expande la lista de regiones de materiales
disponibles

        self.ax.semilogx()
        self.ax.semilogy()
        self.ax.autoscale(True)
        self.ax.grid(True)
        self.ax.set_xlabel(xlabel)
        self.ax.set_ylabel(ylabel)
        self.canvas.draw()  #grafica lo seleccionado
        self.xmin,self.xmax,self.ymin,self.ymax=self.ax.axis()

    def check_all(self, event):
        'Selecciona o deselecciona todos los CheckBoxs si "Graficar todo"
cambia de estado'
        if self.cb_todo.GetValue():
            self.cb_metales.SetValue(True)
            self.cb_ceramicos.SetValue(True)
            self.cb_compuestos.SetValue(True)
            self.cb_polimeros.SetValue(True)
            self.cb_elastomeros.SetValue(True)
            self.cb_espumas.SetValue(True)
            self.cb_naturales.SetValue(True)
        else:
            self.cb_metales.SetValue(False)
            self.cb_ceramicos.SetValue(False)
            self.cb_compuestos.SetValue(False)
            self.cb_polimeros.SetValue(False)
            self.cb_elastomeros.SetValue(False)
            self.cb_espumas.SetValue(False)
            self.cb_naturales.SetValue(False)
```

```
    def some_material(self, event):
        'Cambia el estado del CheckBox "Graficar todo"'
        if self.cb_metales.GetValue()==False:
            self.cb_todo.SetValue(False)
        elif self.cb_ceramicos.GetValue()==False:
            self.cb_todo.SetValue(False)
        elif self.cb_compuestos.GetValue()==False:
            self.cb_todo.SetValue(False)
        elif self.cb_polimeros.GetValue()==False:
            self.cb_todo.SetValue(False)
        elif self.cb_elastomeros.GetValue()==False:
            self.cb_todo.SetValue(False)
        elif self.cb_espumas.GetValue()==False:
            self.cb_todo.SetValue(False)
        elif self.cb_naturales.GetValue()==False:
            self.cb_todo.SetValue(False)

        if self.cb_metales.GetValue()==True:
            if self.cb_ceramicos.GetValue()==True:
                if self.cb_compuestos.GetValue()==True:
                    if self.cb_polimeros.GetValue()==True:
                        if self.cb_elastomeros.GetValue()==True:
                            if self.cb_espumas.GetValue()==True:
                                if self.cb_naturales.GetValue()==True:
                                    self.cb_todo.SetValue(True)

    def rango_materiales(self,R,x_propiedad,y_propiedad,color_rango):
        'grafica las elipses que representan los rangos de las propiedades de
cada material'

        'aquí ingresa la familia o subfamilia con maximos y minimos para ser
graficados'

        color=color_rango
        datos_x=R[:,x_propiedad:x_propiedad+2]
        datos_y=R[:,y_propiedad:y_propiedad+2]

        pts_region=[]
        for i in range(len(R)):
            mx=(datos_x[i,0]+datos_x[i,1])/2
            my=(datos_y[i,0]+datos_y[i,1])/2
            if mx!=0 and my!=0:
                ancho=datos_x[i,1]-datos_x[i,0]
                alto=datos_y[i,1]-datos_y[i,0]

                'Se obtiene una matriz: pts_rango[fila]=[abajo, derecha,
arriba, izquierda]'
                p2=(np.log10(mx+ancho/2), np.log10(my-alto/2))
                p4=(np.log10(mx+ancho/2), np.log10(my+alto/2))
                p6=(np.log10(mx-ancho/2), np.log10(my+alto/2))
                p8=(np.log10(mx-ancho/2), np.log10(my-alto/2))
                'Vértices del rectángulo que contiene la elipse (rango)'
                p1=((p4[0]+p6[0])/2,p2[1])
                p3=(p2[0],(p2[1]+p4[1])/2)
                p5=(p1[0],p4[1])
                p7=(p6[0],p3[1])
                '8 puntos necesarios para dibujar la elipse'
                p1=[10**p1[0],10**p1[1]]
```

```
                p2=[10**p2[0],10**p2[1]]
                p3=[10**p3[0],10**p3[1]]
                p4=[10**p4[0],10**p4[1]]
                p5=[10**p5[0],10**p5[1]]
                p6=[10**p6[0],10**p6[1]]
                p7=[10**p7[0],10**p7[1]]
                p8=[10**p8[0],10**p8[1]]
                'agrega los puntos en una misma tupla'
                pts_region.append(p1)
                pts_region.append(p2)
                pts_region.append(p3)
                pts_region.append(p4)
                pts_region.append(p5)
                pts_region.append(p6)
                pts_region.append(p7)
                pts_region.append(p8)

                pts_ovalo=[p1,p2,p3,p4,p5,p6,p7,p8,p1]

                codes_ovalo = [Path.MOVETO,
                               Path.CURVE3,
                               Path.CURVE3,
                               Path.CURVE3,
                               Path.CURVE3,
                               Path.CURVE3,
                               Path.CURVE3,
                               Path.CURVE3,
                               Path.CURVE3]

                path = Path(pts_ovalo, codes_ovalo)
                patch = patches.PathPatch(path, facecolor=color, lw=0.5,
alpha=0.8)
                self.ax.add_patch(patch)

        'Convierte la tupla de puntos en una matriz'
        pts_rango=np.zeros((len(pts_region),2))
        for i in range(len(pts_region)):
            pts_rango[i,:]=[pts_region[i][0],pts_region[i][1]]

        return pts_rango
        'retorna una matriz con las coordenadas (x,y) de los puntos
extremos(abajo, derec,arriba,izq)...'
        '...de la elipse(rango) de cada material de la región'

    def contorno(self,pts):
        'dibuja un arco que cubre el punto 2'
        def ptos_paralelos(p1,p2):
            'encuentra la pendiente entre dos puntos'
            'p1=(x1,y1); p2=(x2,y2)'
            cd=0.035    #distancia paralela a cada lado

            p1xlog=np.log10(p1[0])
            p1ylog=np.log10(p1[1])
            p2xlog=np.log10(p2[0])
            p2ylog=np.log10(p2[1])

            if p2xlog-p1xlog==0:    #evito el error de tan(90°)
                m=9999999999
            else:
```

```
                m=(p2ylog-p1ylog)/(p2xlog-p1xlog)
                teta=np.arctan(m)

            if m==0:    #evito el error de división para cero al calcular la
pendiente perpendicular
                m=0.0000001

            m_p=-1/m #pendiente perpendicular
            teta_p=np.arctan(m_p)

            x_mediolog=(p1xlog+p2xlog)/2
            y_mediolog=(p1ylog+p2ylog)/2

            x_mp=10**x_mediolog
            y_mp=10**y_mediolog

            dx=abs(cd*np.cos(teta_p))
            dy=abs(cd*np.sin(teta_p))

            #condicionales para identificar en que dirección ubicar el punto
            # de la recta paralela (arriba o abajo del lado)

            if p2xlog>p1xlog:
                if p2ylog>p1ylog:
                    px=x_mediolog+dx
                    py=y_mediolog-dy
                elif p2ylog<p1ylog:
                    px=x_mediolog-dx
                    py=y_mediolog-dy
                elif p2ylog==p1ylog:
                    px=x_mediolog
                    py=y_mediolog-cd
            elif p2xlog<p1xlog:
                if p2ylog>p1ylog:
                    px=x_mediolog+dx
                    py=y_mediolog+dy
                elif p2ylog<p1ylog:
                    px=x_mediolog-dx
                    py=y_mediolog+dy
                elif p2ylog==p1ylog:
                    px=x_mediolog
                    py=y_mediolog+cd
            elif p2xlog==p1xlog:
                if p2ylog>p1ylog:
                    px=x_mediolog+dx
                    py=y_mediolog
                elif p2ylog<p1ylog:
                    px=x_mediolog-dx
                    py=y_mediolog

            pto_paralelo=np.array([px,py])

            return pto_paralelo,m   #coordenadas del punto de referencia de
la recta paralela
            'pto_paralelo=(px,py) coordenadas del punto para construir la
recta paralela'
            'm=pendiente de la recta'

        #Sentencia para ubicar el punto para la paralela del lado
```

```
        coordenadas_m=[]
        for i in range(2):
            coordenadas_m.append(ptos_paralelos(pts[i],pts[i+1]))
        'coordenadas_m=[[[px1,py1],m1] , [[px2,py2],m2]]'

        'descomposición del arreglo de "coordenadas"'
        pc1=coordenadas_m[0][0]
        pc2=coordenadas_m[1][0]
        m1=coordenadas_m[0][1]
        m2=coordenadas_m[1][1]
        'coordenadas de punto paralelo'
        px1=pc1[0]
        py1=pc1[1]
        px2=pc2[0]
        py2=pc2[1]
        'vértices para formar el arco'
        vx=((py2-m2*px2)-(py1-m1*px1))/(m1-m2)
        vy=m1*vx-m1*px1+py1

        vx=10**vx
        vy=10**vy

        px1=10**px1
        py1=10**py1
        px2=10**px2
        py2=10**py2

        p1=np.array([px1,py1])
        p2=np.array([px2,py2])
        v=np.array([vx,vy])

        return v,p2  #los 3 puntos para formar el arco en cada vertice del
poligono

    def grafico_regiones(self, region, color_region):
        'Extrae los puntos de borde para graficar las regiones'
        points = region
        points=np.log10(points) #escala logarítmica

        color=color_region
        #FUNCIÓN PRINCIPAL
        hull=ConvexHull(points)
        hull_indices=np.unique(hull.simplices.flat) #indices de los pares
ordenados que forman el borde
        hull_points = points[hull_indices, :] #extrae los pares ordenados
guiados por hull_indices

        #hull_points: PUNTOS DE BORDE desordenados
        ymin=np.argmin(points[:,1])   #mínimo valor en y
        pmin=points[ymin,:] #coordenadas con el mínimo valor en y

        points_order=[] #PUNTOS DE BORDE ordenados respecto a la coordenada y
menor
        for i in range(0,len(hull_points)):
            points_order.append(hull_points[i])

        points_order.sort(key=lambda p: np.arctan2(p[1]-pmin[1],p[0]-
pmin[0]))
        points_order=np.array(points_order)
```

```
        points_order=10**points_order   #ESCALA NORMAL

        n=len(points_order) #número de vertices del polígono o número de
puntos de borde

        pts_borde=np.zeros(((2*n)+1,2))
        indice=1   #indice de ubicación para la matriz de pts_borde

        for i in range(len(points_order)):
            pt1=points_order[i]   #punto de referencia para iniciar el
trazado: linea-arco
            if i==(len(points_order)-2):    #i= última coordenada de
points_order
                pt2=points_order[i+1]  #next_point es el punto inicial
                pt3=points_order[0]
            elif i==(len(points_order)-1):
                pt2=points_order[0]   #el siguiente punto es en sentido
horario debido al ordenamiento de points_order
                pt3=points_order[1]
            else:
                pt2=points_order[i+1]
                pt3=points_order[i+2]

            pts=np.zeros((3,2)) #matriz con las coordenadas de 3 puntos
continuos
            pts[0]=pt1
            pts[1]=pt2
            pts[2]=pt3
            v,p2=self.contorno(pts)
            pts_borde[indice]=v
            pts_borde[indice+1]=p2

            indice=indice+2

        pts_codes=[]
        pts_codes.append(Path.MOVETO)
        for i in range(1,len(pts_borde)):
            pts_codes.append(Path.CURVE3)

        'grafica las regiones'
        pts_borde[0]=pts_borde[1]   #agrego el punto inicial "MOVETO" a la
matriz de puntos de borde

        path = Path(pts_borde, pts_codes)
        patch = patches.PathPatch(path, facecolor=color, lw=0, alpha=0.3)
        self.ax.add_patch(patch)

        'grafica los bordes de las regiones'
        pts_borde[0]=pts_borde[len(pts_borde)-1]   #agrega el punto inicial
"MOVETO" a la matriz de puntos de borde

        path = Path(pts_borde, pts_codes)
        edge = patches.PathPatch(path, edgecolor='grey', facecolor='none',
lw=1)
        self.ax.add_patch(edge)

    def importacion_datos(self):
        'importa los datos desde las BASE DE MATERIALES'
```

```
        'y entrega una matriz con los valores de las 7 regiones de ASHBY'
        archivo_base_excel=xl.open_workbook('Base Materiales.xlsx')
        hoja_base=archivo_base_excel.sheet_by_name('Hoja1')
        total_filas=hoja_base.nrows #total filas con datos
        total_colums=hoja_base.ncols    #total columnas con datos

        tabla=[]    #tabla de Base de Materiales importada
        fila=[]
        for i in range(1,total_filas):
            for j in range (total_colums):
                fila.append(hoja_base.cell(i,j).value)
            tabla.append(fila)
            fila=[]
            i=i+1
        return tabla

    def validacion_parametros(self):
        'Selecciona los materiales de cada familia que cumplen los parámetros
definidos por el usuario'
        #PARÁMETROS PARA FILTRAR MATERIALES
        'obtención de parámetros mínimos'
        densidad_min = self.txt_densidad_min.GetValue()
        young_min = self.txt_young_min.GetValue()
        rmecanica_min = self.txt_rmecanica_min.GetValue()
        ctermica_min = self.txt_ctermica_min.GetValue()
        cdilatacion_min = self.txt_cdilatacion_min.GetValue()
        temp_serv_min = self.txt_temp_serv_min.GetValue()
        'obtención de parámetros máximos'
        densidad_max = self.txt_densidad_max.GetValue()
        young_max = self.txt_young_max.GetValue()
        rmecanica_max = self.txt_rmecanica_max.GetValue()
        ctermica_max = self.txt_ctermica_max.GetValue()
        cdilatacion_max = self.txt_cdilatacion_max.GetValue()
        temp_serv_max = self.txt_temp_serv_max.GetValue()

        tabla=self.tabla    #agrego la tabla de datos importados a una
variable interna
        #obtengo la lista de familias y subfamilias de la tabla
        regiones=list()
        familias=list()
        subfamilias=list()
        for i in range(len(tabla)):
            regiones.append(tabla[i][1])
            familias.append(tabla[i][2])
            subfamilias.append(tabla[i][3])

        nom_regiones=list(set(regiones))    #lista de regiones sin duplicados
        nom_regiones.sort()     #orden alfabético
        nom_familias=list(set(familias))    #lista de familias sin duplicados
        nom_familias.sort()  #orden alfabético
        nom_subfamilias=list(set(subfamilias))    #lista de subfamilias sin
duplicados
        nom_subfamilias.sort()  #orden alfabético
        #ceramicos,compuestos,elastomeros,espumas,metales,naturales,no
tecnicos,polimeros
        tabla_validada=[]    #tabla con los datos que cumplen cada parámetro
dado
        if tabla!=[]:   #tabla inicial debe tener datos
```

```
            if densidad_min !=  '': #condicional cuando existe parámetro de
densidad mínima
                for j in range(len(tabla)):
                    if float(tabla[j][6])>=float(densidad_min):    #Si la
densidad del material(tabla[j][5]) es mayor que densida_min
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if densidad_max !=  '': #condicional cuando existe parámetro de
densidad máxima
            if tabla!=[]:   #la nueva tabla debe contener datos aun para
filtrar
                for j in range(len(tabla)):
                    if float(tabla[j][5])<=float(densidad_max):    #Si la
densidad del material(tabla[j][5]) es mayor que densida_max
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if young_min !=  '': #condicional cuando existe parámetro de densidad
máxima
            if tabla!=[]:   #la nueva tabla debe contener datos aun para
filtrar
                for j in range(len(tabla)):
                    if float(tabla[j][8])>=float(young_min):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if young_max !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][7])<=float(young_max):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if rmecanica_min !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][10])>=float(rmecanica_min):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if rmecanica_max !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][9])<=float(rmecanica_max):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if ctermica_min !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][12])>=float(ctermica_min):
                        tabla_validada.append(tabla[j][0:17])
```

```
                tabla=tabla_validada
                tabla_validada=[]

        if ctermica_max !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][11])<=float(ctermica_max):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if cdilatacion_min !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][14])>=float(cdilatacion_min):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if cdilatacion_max !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][13])<=float(cdilatacion_max):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if temp_serv_min !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][16])>=float(temp_serv_min):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        if temp_serv_max !=  '':
            if tabla!=[]:
                for j in range(len(tabla)):
                    if float(tabla[j][15])<=float(temp_serv_max):
                        tabla_validada.append(tabla[j][0:17])
                tabla=tabla_validada
                tabla_validada=[]

        'HASTA AQUÍ SE TIENE LA TABLA CON DATOS FILTRADOS SEGÚN LOS
PARÁMETROS DEFINIDOS POR EL USUARIO'
        LISTA_REGIONES=[] #lista que contiene las tablas de cada familia
(sólo valores)
        tabla_region=[]    #lista que contiene la tabla de una familia
        for i in range(len(nom_regiones)):
            for j in range(len(tabla)):
                    if tabla[j][1]==nom_regiones[i]:
                            tabla_region.append(tabla[j][:])  #divido la
tabla por regiones
            LISTA_REGIONES.append(tabla_region)  #agrego la tabla de la
familia al diccionario
            tabla_region=[]

##        #Extraigo listas con los nombres de cada REGIÓN
##        R1_nombre=NOMBRE_REGIONES[0]    #CERÁMICOS
```

```
##        R2_nombre=NOMBRE_REGIONES[1]    #COMPUESTOS
##        R3_nombre=NOMBRE_REGIONES[2]    #ELASTOMEROS
##        R4_nombre=NOMBRE_REGIONES[3]    #ESPUMAS
##        R5_nombre=NOMBRE_REGIONES[4]    #METALES
##        R6_nombre=NOMBRE_REGIONES[5]    #NATURALES
##        R7_nombre=NOMBRE_REGIONES[7]    #POLÍMEROS
##        R8_nombre=NOMBRE_REGIONES[6]    #NO TÉCNICOS

        #Convierto cada región de tipo "lista" a "matriz"(permite hacer los
cálculos)
        R1=np.zeros((len(LISTA_REGIONES[0]),12))    #CERÁMICOS
        for i in range(len(LISTA_REGIONES[0])): #filas
            for j in range(5,len(LISTA_REGIONES[0][0])-1):  #columnas
                r=j-5
                R1[i,r]=float(LISTA_REGIONES[0][i][j])

        R2=np.zeros((len(LISTA_REGIONES[1]),12))    #COMPUESTOS
        for i in range(len(LISTA_REGIONES[1])): #filas
            for j in range(5,len(LISTA_REGIONES[1][0])-1):  #columnas
                r=j-5
                R2[i,r]=float(LISTA_REGIONES[1][i][j])

        R3=np.zeros((len(LISTA_REGIONES[2]),12))    #ELASTOMEROS
        for i in range(len(LISTA_REGIONES[2])): #filas
            for j in range(5,len(LISTA_REGIONES[2][0])-1):  #columnas
                r=j-5
                R3[i,r]=float(LISTA_REGIONES[2][i][j])

        R4=np.zeros((len(LISTA_REGIONES[3]),12))    #ESPUMAS
        for i in range(len(LISTA_REGIONES[3])): #filas
            for j in range(5,len(LISTA_REGIONES[3][0])-1):  #columnas
                r=j-5
                R4[i,r]=float(LISTA_REGIONES[3][i][j])

        R5=np.zeros((len(LISTA_REGIONES[4]),12))    #METALES
        for i in range(len(LISTA_REGIONES[4])): #filas
            for j in range(5,len(LISTA_REGIONES[4][0])-1):  #columnas
                r=j-5
                R5[i,r]=float(LISTA_REGIONES[4][i][j])

        R6=np.zeros((len(LISTA_REGIONES[5]),12))    #NATURALES
        for i in range(len(LISTA_REGIONES[5])): #filas
            for j in range(5,len(LISTA_REGIONES[5][0])-1):  #columnas
                r=j-5
                R6[i,r]=float(LISTA_REGIONES[5][i][j])

        R7=np.zeros((len(LISTA_REGIONES[6]),12))    #POLÍMEROS
        for i in range(len(LISTA_REGIONES[6])): #filas
            for j in range(5,len(LISTA_REGIONES[6][0])-1):  #columnas
                r=j-5
                R7[i,r]=float(LISTA_REGIONES[6][i][j])
        return R1,R2,R3,R4,R5,R6,R7,LISTA_REGIONES

if __name__ == '__main__':
    app = wx.App()
    GUI_ASHBY(None, title='GRÁFICAS ASHBY - FACULTAD DE INGENIERÍA MECÁNICA
EPN')
    app.MainLoop()
```

ANEXO II.

BASE DE DATOS DE MATERIALES GENERADA EN LA HOJA DE CALCULO CAL

Tabla I. 1. Valores de las diferentes propiedades.

No.	Región	Material	min [kg/m3]	Max [kg/m3]	min [Gpa]	Max [GPa]	min [MPa]	Max [MPa]	min [W/mk]	Max [W/mk]	min [10^-6/C]	Max [10^-6/C]	min [°C]	Max [°C]
1	Cerámicos	Ladrillos	1600	2100	15	30	5	14	0,46	0,73	5	8	920	1230
2	Cerámicos	Dióxido de circonio	6030	6160	135	141	32	60	0,8	2,4	5,9	6	480	510
3	Cerámicos	Piedra	2300	2600	40	60	2	25	5,4	6	3,7	6,3	327	1430
4	Cerámicos	Alúmina	3800	3980	343	390	350	588	26	38,5	7	7,9	1080	1300
5	Cerámicos	Carburo de silicio	3100	3210	400	460	400	610	80	130	4	4,8	1400	1700
6	Cerámicos	Carburo de wolframio	15300	15900	625	700	335	550	55	88	5,2	7,1	750	1000
7	Cerámicos	Nitruro de aluminio	3260	3330	302	348	300	350	140	200	4,9	5,5	1030	1730
8	Cerámicos	Silicio	2300	2350	140	155	160	180	140	150	2	3,2	527	577
9	Cerámicos	Vidrio corriente	2440	2490	68	72	30	35	0,7	1,3	9,1	9,5	170	400
10	Cerámicos	Vidrio de borosilicato	2200	2300	61	64	22	32	1	1,3	3,2	4	230	460
11	Cerámicos	Vidrio de sílice	2170	2220	68	74	45	155	1,4	1,5	0,55	0,75	897	1400
12	Cerámicos	Diamante	3440	3580	1050	1210	2800	2930	900	2320	0,8	1,2	1500	1700
13	Cerámicos	Concreto	2200	2800	10	50	0,1	0,3	1,43	2	12	14	50	125
14	Cerámicos	Cerámica de vidrio	2200	2800	64	110	750	2129	1,3	2,5	1	5	560	1600
15	Cerámicos	Carburo de boro	2350	2550	400	472	2583	5687	40	90	3,2	3,4	1100	1300
16	Compuestos	Material compuesto CFRP	1500	1600	69	150	550	1050	1,28	2,6	1	4	140	220
17	Compuestos	Material compuesto GFRP	1750	1970	15	28	110	192	0,4	0,55	8,64	33	140	220
18	Compuestos	Carburo de aluminio	2660	2900	81	100	280	324	180	160	15	23	225	327

19	Elastómeros	Caucho	920	930	0,0015	0,0025	20	30	0,1	0,14	150	450	68,9	107
20	Elastómeros	Elastómeros de silicona	1300	1800	0,005	0,02	2,4	5,5	0,3	1	250	300	227	287
21	Elastómeros	Goma de butilo	900	920	0,001	0,002	2	3	0,08	0,1	120	300	96,9	117
22	Elastómeros	Goma de polisopreno	930	940	0,0014	0,004	20	25	0,08	0,14	150	450	96,9	117
23	Elastómeros	Goma Eva	945	955	0,01	0,04	12	18	0,3	0,4	160	190	46,9	51,9
24	Elastómeros	Goma SBR	1130	1150	0,0038	0,006	16	26	0,4	0,9	160	180	70	110
25	Elastómeros	Neopreno	1230	1250	0,0007	0,002	3,4	24	0,1	0,12	575	610	102	112
26	Espumas	Espuma flexible de polímero de baja densidad	38	70	0,001	0,003	0,01	0,12	0,04	0,059	115	220	82,9	112
27	Espumas	Espuma flexible de polímero de media densidad	70	115	0,004	0,012	0,02	0,3	0,041	0,078	115	220	82,9	112
28	Espumas	Espuma flexible de polímero de muy baja densidad	16	35	0,00025	0,001	0,01	0,07	0,036	0,048	120	220	86,9	112
29	Espumas	Espuma rígida de polímero de alta densidad	170	470	0,2	0,48	0,8	12	0,034	0,063	22	70	66,9	167
30	Espumas	Espuma rígida de polímero de baja densidad	36	70	0,023	0,08	0,3	1,7	0,023	0,04	20	80	66,9	147
31	Espumas	Espuma rígida de polímero de media densidad	78	165	0,08	0,2	0,4	3,5	0,027	0,038	20	70	66,9	157
32	Metales	Acero de baja aleación	7800	7900	205	217	400	1100	34	55	10,5	13,5	500	550
33	Metales	Acero inoxidable	7600	8100	189	210	170	100	12	24	13	20	750	820
34	Metales	Acero ordinario de alto contenido de C	7800	7900	200	215	400	1160	47	53	11	13,5	350	400
35	Metales	Acero ordinario de bajo contenido de C	7800	7900	200	215	250	395	49	54	11,5	13	350	400
36	Metales	Acero ordinario de medio contenido de C	7800	7900	200	216	305	900	45	55	10	14	370	420
37	Metales	Fundición dúctil de hierro	7050	7250	165	180	250	680	29	44	10	12,5	350	450
38	Metales	Fundición gris de hierro	7050	7250	80	138	140	420	40	72	11	12,5	350	450

39	Metales	Aleaciones de aluminio	2500	2900	68	82	30	500	76	235	21	24	120	210
40	Metales	Aleaciones de cinc	4950	7000	68	95	80	450	100	135	23	28	79,9	110
41	Metales	Aleaciones de cobre	8930	8940	112	148	30	500	160	390	16,9	18	180	350
42	Metales	Aleaciones de magnesio	1740	1950	42	47	70	400	50	156	24,6	28	120	200
43	Metales	Aleaciones de níquel	8830	8950	190	220	70	1100	67	91	12	13,5	500	1200
44	Metales	Aleaciones de plomo	10000	11400	12,5	15	8	14	22	36	18	32	60	120
45	Metales	Aleaciones de titanio	4400	4800	90	120	250	1250	7	14	7,9	11	300	500
46	Metales	Aleaciones de wolframio	17800	19600	310	380	525	800	100	142	4	5,6	1350	1400
47	Metales	Estaño	7260	7270	41	45	7	15	60	61,5	22,5	23,5	90	100
48	Metales	Oro	19300	19400	77	81	165	205	305	319	13,5	14,5	130	220
49	Metales	Plata	10500	10600	69	73	190	300	416	422	19,5	19,9	96,9	190
50	Naturales	Bambú	600	800	15	20	35	44	0,1	0,18	2,6	10	117	137
51	Naturales	Corcho	240	350	0,013	0,05	0,3	1,5	0,035	0,048	130	230	117	137
52	Naturales	Cuero	810	1050	0,1	0,5	5	10	0,156	0,16	40	50	107	127
53	Naturales	Madera (dirección transversal)	660	800	0,5	3	2	6	0,15	0,19	31,8	42,5	117	137
54	Naturales	Madera (dirección longitudinal)	600	800	6	20	30	70	0,31	0,38	2	11	117	137
55	Naturales	Papel-cartón	480	860	3	8,9	15	34	0,06	0,17	5	20	77	130
56	Naturales	Concha	1800	2100	30	65	30	85	1	5	20	30	110	130
57	Polímeros	Poliuretano elastómero	1020	1250	0,002	0,03	25	51	0,28	0,3	150	165	66,9	86,9
58	Polímeros	Epoxis	1110	1400	2,35	3,08	36	71,7	0,18	0,5	58	117	140	180
59	Polímeros	Fenólicos	1240	1320	2,76	4,83	27,6	49,7	0,141	0,152	120	125	200	230
60	Polímeros	Poliéster	1040	1400	2,07	4,41	33	40	0,287	0,299	99	180	130	150
61	Polímeros	Ionómeros	930	960	0,2	0,424	8,27	15,9	0,239	0,276	180	306	48,9	61,9
62	Polímeros	Poliamida de nailon	1120	1140	2,62	3,2	50	94,8	0,233	0,253	144	149	110	140
63	Polímeros	Policarbonato	1140	1210	2	2,44	59	70	0,189	0,218	120	137	101	144
64	Polímeros	Polietileno	939	960	0,621	0,896	17,9	29	0,403	0,435	126	198	90	110
65	Polímeros	ABS	1010	1210	1,1	2,9	18,5	51	0,188	0,335	84,6	234	61,9	76,9

66	Polímeros	PEEK	1300	1320	3,75	3,95	65	95	0,24	0,26	72	194	239	260
67	Polímeros	PET	1290	1400	2,76	4,14	56,5	62,3	0,138	0,151	115	119	66,9	86,9
68	Polímeros	Acrílico	1160	1220	2,24	3,8	53,8	72,4	0,083	0,251	72	162	41,9	56,9
69	Polímeros	Acetal	1390	1430	2,5	5	48,6	72,4	0,221	0,35	75,7	202	76,9	96,9
70	Polímeros	Polipropileno	890	910	0,896	1,55	20,7	37,2	0,111	0,167	122	180	100	115
71	Polímeros	Poliestireno	1040	1050	1,2	2,6	28,7	56,2	0,121	0,131	90	153	76,9	103
72	Polímeros	PVC	1300	1580	2,14	4,14	35,4	52,1	0,147	0,293	100	150	60	70
73	Polímeros	Polímeros de celulosa	980	1300	1,6	2	25	45	0,13	0,3	150	300	52,9	89,9
74	Polímeros	Poliuretano	1120	1240	1,31	2,07	40	53,8	0,235	0,244	90	144	64,9	80
75	Polímeros	Teflón	2140	2200	0,4	0,552	15	25	0,242	0,261	126	216	250	271

(Fuente: propia)

ANEXO III.

MATERIALES UTILIZADOS EN LA COMPARACIÓN DE LAS GRÁFICAS

Tabla I. 2. Materiales utilizados para realizar la gráfica de módulo de Young vs. densidad.

Material	Min. densidad [kg/m3]	máx. densidad [kg/m3]	Min. módulo [Gpa]	máx. módulo [GPa]
Acero de baja aleación	7800	7900	205	217
Acero inoxidable	7600	8100	189	210
Acero ordinario de alto contenido de C	7800	7900	200	215
Acero ordinario de bajo contenido de C	7800	7900	200	215
Acero ordinario de medio contenido de C	7800	7900	200	216
Fundición dúctil de hierro	7050	7250	165	180
Fundición gris de hierro	7050	7250	80	138
Aleaciones de aluminio	2500	2900	68	82
Aleaciones de cinc	4950	7000	68	95
Aleaciones de cobre	8930	8940	112	148
Aleaciones de magnesio	1740	1950	42	47
Aleaciones de níquel	8830	8950	190	220
Aleaciones de plomo	10000	11400	12,5	15
Aleaciones de titanio	4400	4800	90	120
Aleaciones de wolframio	17800	19600	310	380
Estaño	7260	7270	41	45
Oro	19300	19400	77	81
Plata	10500	10600	69	73
Dióxido de circonio	6030	6160	135	141
Piedra	2300	2600	40	60
Alúmina	3800	3980	343	390
Carburo de silicio	3100	3210	400	460
Carburo de wolframio	15300	15900	625	700
Nitruro de aluminio	3260	3330	302	348
Silicio	2300	2350	140	155
Material compuesto CFRP	1500	1600	69	150
Material compuesto GFRP	1750	1970	15	28
Espuma flexible de polímero de baja densidad	38	70	0,001	0,003
Espuma flexible de polímero de media densidad	70	115	0,004	0,012
Espuma flexible de polímero de muy baja densidad	16	35	0,00025	0,001
Espuma rígida de polímero de alta densidad	170	470	0,2	0,48
Espuma rígida de polímero de baja densidad	36	70	0,023	0,08
Espuma rígida de polímero de media densidad	78	165	0,08	0,2
Bambú	600	800	15	20
Corcho	240	350	0,013	0,05

Cuero	810	1050	0,1	0,5
Madera (dirección transversal)	660	800	0,5	3
Madera (dirección longitudinal)	600	800	6	20
Papel-cartón	480	860	3	8,9
Caucho	920	930	0,0015	0,0025
Elastómeros de silicona	1300	1800	0,005	0,02
Goma de butilo	900	920	0,001	0,002
Goma de Polisopreno	930	940	0,0014	0,004
Goma Eva	945	955	0,01	0,04
Goma SBR	1130	1150	0,0038	0,006
Neopreno	1230	1250	0,0007	0,002
Poliuretano elastomérico	1020	1250	0,002	0,03
Epoxis	1110	1400	2,35	3,08
Fenólicos	1240	1320	2,76	4,83
Poliéster	1040	1400	2,07	4,41
Ionómeros	930	960	0,2	0,424
Poliamida de nailon	1120	1140	2,62	3,2
Policarbonato	1140	1210	2	2,44
Polietileno	939	960	0,621	0,896
ABS	1010	1210	1,1	2,9
PEEK	1300	1320	3,75	3,95
PET	1290	1400	2,76	4,14
Acrílico	1160	1220	2,24	3,8
Acetal	1390	1430	2,5	5
Polipropileno	890	910	0,896	1,55
Poliestireno	1040	1050	1,2	2,6
PVC	1300	1580	2,14	4,14
Polímeros de celulosa	980	1300	1,6	2
Poliuretano	1120	1240	1,31	2,07
Teflón	2140	2200	0,4	0,552

(Fuente: propia)

Tabla I. 3. Materiales utilizados para recrear la gráfica de resistencia mecánica vs. densidad.

Material	min densidad [kg/m3]	Max densidad [kg/m3]	min resistencia mecánica [MPa]	Max resistencia mecánica [MPa]
Acero de baja aleación	7800	7900	400	1100
Acero inoxidable	7600	8100	170	100
Acero ordinario de alto contenido de C	7800	7900	400	1160
Acero ordinario de bajo contenido de C	7800	7900	250	395
Acero ordinario de medio contenido de C	7800	7900	305	900
Fundición dúctil de hierro	7050	7250	250	680

Fundición gris de hierro	7050	7250	140	420
Aleaciones de aluminio	2500	2900	30	500
Aleaciones de cinc	4950	7000	80	450
Aleaciones de cobre	8930	8940	30	500
Aleaciones de magnesio	1740	1950	70	400
Aleaciones de níquel	8830	8950	70	1100
Aleaciones de plomo	10000	11400	8	14
Aleaciones de titanio	4400	4800	250	1250
Aleaciones de wolframio	17800	19600	525	800
Estaño	7260	7270	7	15
Oro	19300	19400	165	205
Plata	10500	10600	190	300
Ladrillos	1600	2100	5	14
Piedra	2300	2600	2	25
Alúmina	3800	3980	350	588
Carburo de silicio	3100	3210	400	610
Nitruro de aluminio	3260	3330	300	350
Silicio	2300	2350	160	180
Material compuesto CFRP	1500	1600	550	1050
Material compuesto GFRP	1750	1970	110	192
Espuma flexible de polímero de baja densidad	38	70	0,01	0,12
Espuma flexible de polímero de media densidad	70	115	0,02	0,3
Espuma flexible de polímero de muy baja densidad	16	35	0,01	0,07
Espuma rígida de polímero de alta densidad	170	470	0,8	12
Espuma rígida de polímero de baja densidad	36	70	0,3	1,7
Espuma rígida de polímero de media densidad	78	165	0,4	3,5
Bambú	600	800	35	44
Corcho	240	350	0,3	1,5
Cuero	810	1050	5	10
Madera (dirección transversal)	660	800	2	6
Madera (dirección longitudinal)	600	800	30	70
Papel-cartón	480	860	15	34
Caucho	920	930	20	30
Elastómeros de silicona	1300	1800	2,4	5,5
Goma de butilo	900	920	2	3
Goma de Polisopreno	930	940	20	25
Goma Eva	945	955	12	18
Goma SBR	1130	1150	16	26
Neopreno	1230	1250	3,4	24
Poliuretano elastomérico	1020	1250	25	51
Epoxis	1110	1400	36	71,7
Fenólicos	1240	1320	27,6	49,7
Poliéster	1040	1400	33	40
Ionómeros	930	960	8,27	15,9

Poliamida de nailon	1120	1140	50	94,8
Policarbonato	1140	1210	59	70
Polietileno	939	960	17,9	29
ABS	1010	1210	18,5	51
PEEK	1300	1320	65	95
PET	1290	1400	56,5	62,3
Acrílico	1160	1220	53,8	72,4
Acetal	1390	1430	48,6	72,4
Polipropileno	890	910	20,7	37,2
Poliestireno	1040	1050	28,7	56,2
PVC	1300	1580	35,4	52,1
Polímeros de celulosa	980	1300	25	45
Poliuretano	1120	1240	40	53,8
Teflón	2140	2200	15	25
Diamante	3440	3580	2800	2930
Concreto	2200	2800	0,1	0,3

Tabla I. 4. Materiales utilizados para recrear la gráfica de coeficiente de dilatación térmica vs. conductividad térmica.

Material	**min conductividad térmica [W/mk]**	**Max conductividad térmica [W/mk]**	**min expansión térmica [10^-6/C]**	**Max expansión térmica [10^-6/C]**
Acero de baja aleación	34	55	10,5	13,5
Acero inoxidable	12	24	13	20
Acero ordinario de alto contenido de C	47	53	11	13,5
Acero ordinario de bajo contenido de C	49	54	11,5	13
Acero ordinario de medio contenido de C	45	55	10	14
Fundición dúctil de hierro	29	44	10	12,5
Fundición gris de hierro	40	72	11	12,5
Aleaciones de aluminio	76	235	21	24
Aleaciones de cinc	100	135	23	28
Aleaciones de cobre	160	390	16,9	18
Aleaciones de magnesio	50	156	24,6	28
Aleaciones de níquel	67	91	12	13,5
Aleaciones de plomo	22	36	18	32
Aleaciones de titanio	7	14	7,9	11
Aleaciones de wolframio	100	142	4	5,6
Estaño	60	61,5	22,5	23,5
Oro	305	319	13,5	14,5
Plata	416	422	19,5	19,9
Piedra	5,4	6	3,7	6,3

Alúmina	26	38,5	7	7,9
Carburo de silicio	80	130	4	4,8
Nitruro de aluminio	140	200	4,9	5,5
Silicio	140	150	2	3,2
Material compuesto CFRP	1,28	2,6	1	4
Material compuesto GFRP	0,4	0,55	8,64	33
Espuma flexible de polímero de baja densidad	0,04	0,059	115	220
Espuma flexible de polímero de media densidad	0,041	0,078	115	220
Espuma flexible de polímero de muy baja densidad	0,036	0,048	120	220
Espuma rígida de polímero de alta densidad	0,034	0,063	22	70
Espuma rígida de polímero de baja densidad	0,023	0,04	20	80
Espuma rígida de polímero de media densidad	0,027	0,038	20	70
Bambú	0,1	0,18	2,6	10
Cuero	0,156	0,16	40	50
Madera (dirección transversal)	0,15	0,19	31,8	42,5
Madera (dirección longitudinal)	0,31	0,38	2	11
Caucho	0,1	0,14	150	450
Elastómeros de silicona	0,3	1	250	300
Goma de butilo	0,08	0,1	120	300
Goma de polisopreno	0,08	0,14	150	450
Goma Eva	0,3	0,4	160	190
Goma SBR	0,4	0,9	160	180
Neopreno	0,1	0,12	575	610
Poliuretano elastomérico	0,28	0,3	150	165
Epoxis	0,18	0,5	58	117
Fenólicos	0,141	0,152	120	125
Poliéster	0,287	0,299	99	180
Ionómeros	0,239	0,276	180	306
Poliamida de nailon	0,233	0,253	144	149
Policarbonato	0,189	0,218	120	137
Polietileno	0,403	0,435	126	198
ABS	0,188	0,335	84,6	234
PEEK	0,24	0,26	72	194
PET	0,138	0,151	115	119
Acrílico	0,0837	0,251	72	162
Acetal	0,221	0,35	75,7	202
Polipropileno	0,1113	0,167	122	180
Poliestireno	0,121	0,131	90	153
PVC	0,147	0,293	100	150
Polímeros de celulosa	0,13	0,3	150	300

Poliuretano	0,235	0,244	90	144
Teflón	0,242	0,261	126	216
Diamante	900	2320	0,8	1,2
Concreto	1,43	2	12	14

(Fuente: propia)

ANEXO IV.

MANUAL DE USUARIO DE LA INTERFAZ GRÁFICA

El código de programación utiliza 2 archivos: La base de materiales con extensión .xls con nombre “Base Materiales” y una imagen del escudo de la Escuela Politécnica Nacional que aparece como ícono en la parte superior izquierda de la interfaz. Esta imagen debe tener el nombre de “EPNi” con extensión .ico. Es necesario que estos dos archivos estén en la misma carpeta donde se encuentra el archivo .py que contiene el código y que los nombres sean exactamente los mismos, para que de esta forma el código reconozca los archivos. Tener en cuenta que Python diferencia entre mayúsculas y minúsculas.

Anteriormente se mencionó que la interfaz gráfica se divide en dos paneles principales. En la figura I.1. se muestran las partes de estos paneles.

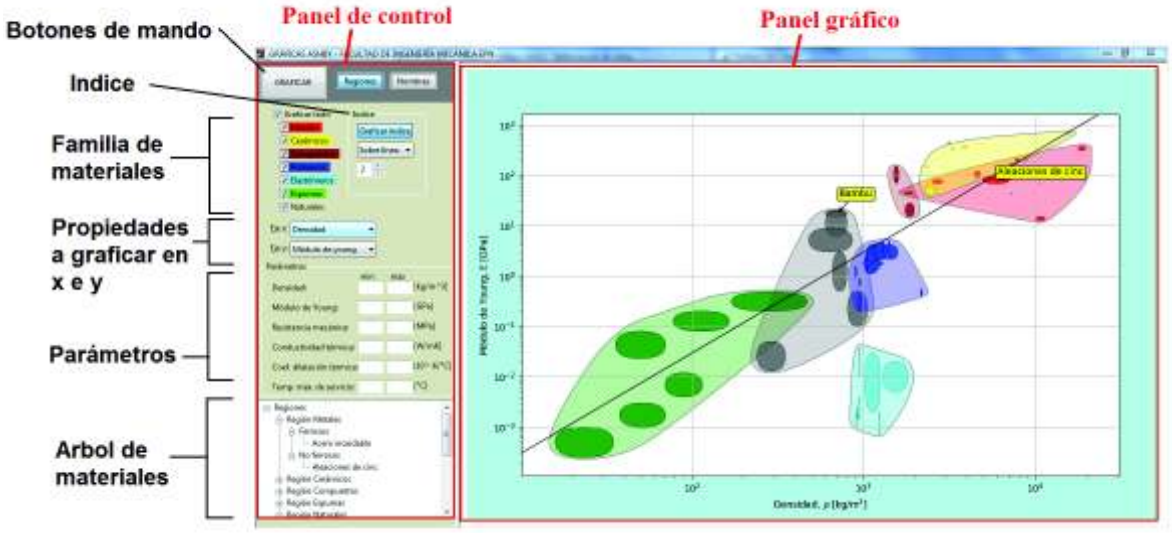

Figura I. 1. Secciones de la interfaz gráfica.
(Fuente: Propia)

En la sección de botones de mando se encuentran 3 botones: Graficar, Regiones y Nombres. El botón “Graficar” es el botón maestro, el cual al ser presionado grafica los diagramas de Ashby. El botón “Regiones” es un botón de dos estados, “presionado” o “no presionado”. Cuando está en estado “presionado” permite graficar las envolventes que agrupan a los materiales según su familia. La función del botón “Nombres” se describirá más adelante.

En la sección de familia de materiales, se puede elegir cual es la familia de interés para graficar. Por defecto, al abrir la interfaz están seleccionadas todas las familias. Adicional a esto, en la figura I.2. se puede observar que cada familia tiene un código de colores en la etiqueta, el cual hace referencia al color de las burbujas de los materiales y su envolvente en la gráfica. Ver figura

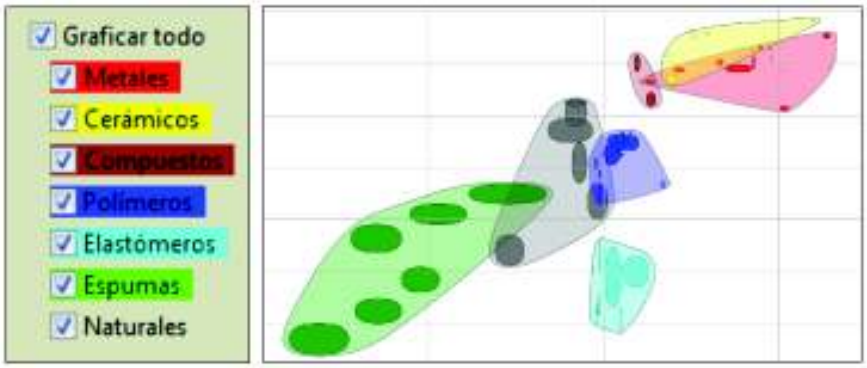

Figura I. 2. Selección de materiales y código de colores según la familia.
(Fuente: propia)

Debajo de la sección de selección de familia se encuentran dos cuadros combinados (ver figura I.3.) donde se selecciona la propiedad a graficar en el eje x e y.

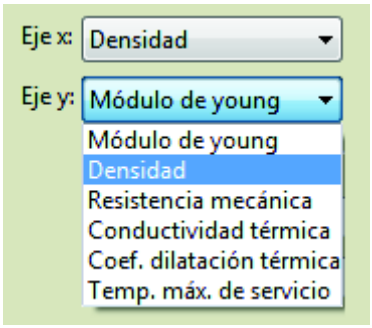

Figura I. 3. Opciones de propiedades a graficar.
(Fuente: propia)

En la sección de parámetros se pueden establecer 12 parámetros, un parámetro mínimo y uno máximo en cada una de las 6 propiedades disponibles. Los valores deben ser introducidos de acuerdo a las unidades mostradas en cada propiedad (ver figura I.4.). El usuario debe saber que si especifica demasiados parámetros posiblemente no exista un material acorde, por lo que se recomienda ingresar solo los parámetros estrictamente necesarios.

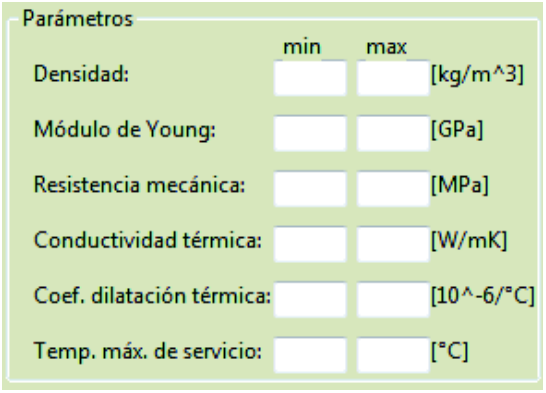

Figura I. 4. Sección de restricción de parámetros.
(Fuente: propia)

En la sección índice se puede añadir un parámetro adicional, siendo este el índice, el cual es un parámetro de rendimiento obtenido de forma analítica. En la figura I.5. se observa un botón, un cuadro combinado y un deslizador que permiten configurar este parámetro para que sea dibujado como una recta en la gráfica.

Figura I. 5. Sección para establecer el índice de rendimiento.
(Fuente: propia)

Para graficar la recta es necesario que el botón “Graficar índice” esté habilitado o presionado, luego se define la pendiente de la recta en el deslizador, y se selecciona la forma de filtrar los materiales en el cuadro combinado. Por último, para que aparezca la recta en la gráfica basta con hacer clic sobre la misma en un punto que sea de interés. Mientras esté presionado el botón “Graficar índice” cada vez que se haga clic sobre la gráfica se dibujará la recta en la posición donde se hizo clic.

Considerar que el deslizador tiene los valores 1, 2 y 3, los cuales son los índices que se usan generalmente.

En el cuadro combinado se tiene tres opciones de filtro: superior, inferior y sobre línea. Si se elige “Superior”, se obtendrá en el árbol de materiales solo aquellos materiales que estén en la zona por encima de la recta del índice (ver figura I.6.) lo cual implica que buscamos maximizar una propiedad. Si se elige la opción “Inferior” se obtendrán los materiales que se encuentren por debajo de la recta, lo que significa que se busca minimizar una propiedad. Si se selección la opción “sobre línea” se obtendrán los materiales en los cuales la recta cruce a través de las burbujas, lo que implica que son materiales que tienen la misma relación entre el eje x e y, o, dicho de otra forma, materiales que tienen el índice equivalente.

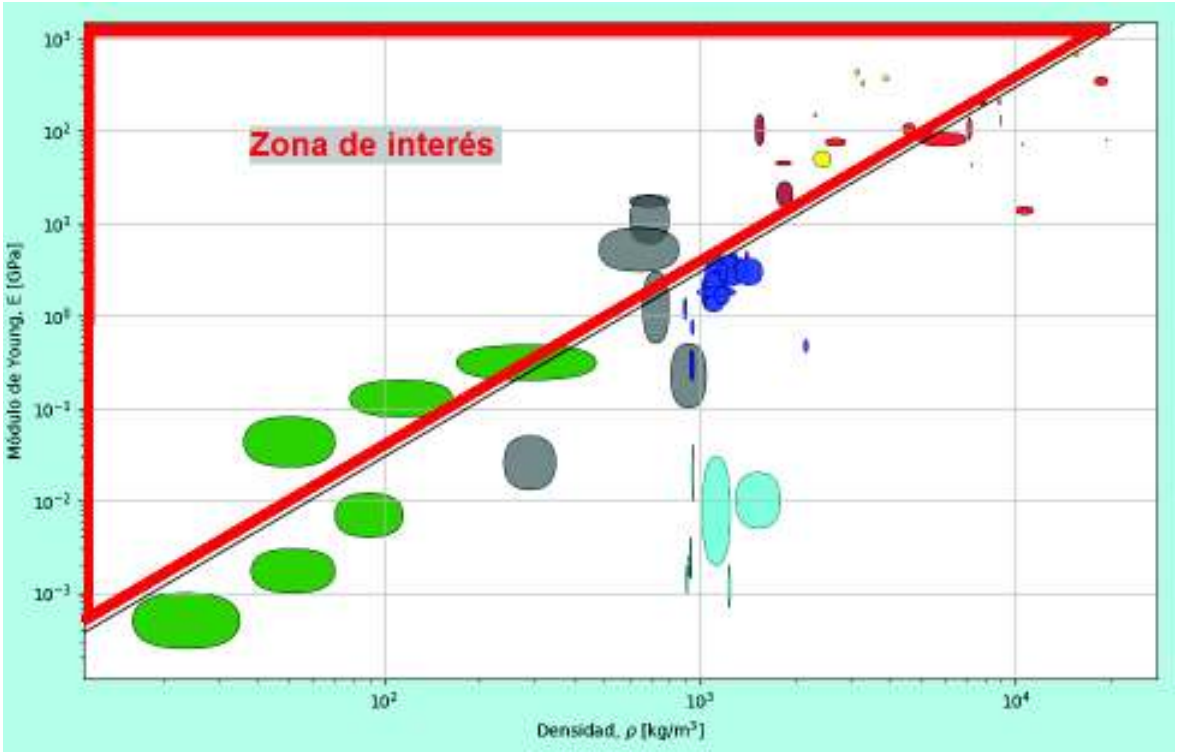

Figura I. 6. División de la zona de materiales mediante el índice.
(Fuente: propia)

Para tener facilidad en la selección del material se incorporó en la parte inferior izquierda un panel que contiene un árbol con la lista de los materiales disponibles, en el cual se muestran todos materiales que cumplan con los parámetros de diseño especificados por el usuario (ver figura I.7.).

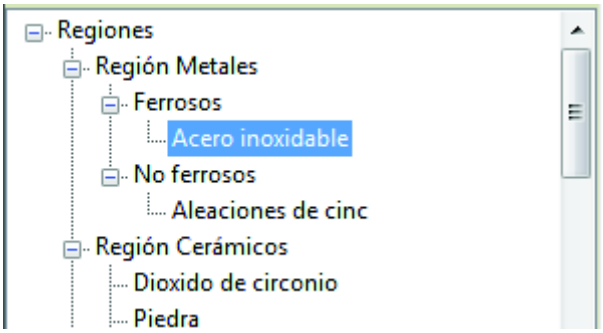

Figura I. 7. Panel del árbol con la lista de materiales disponibles.
(Fuente: propia)

La interfaz también cuenta con la facilidad de mostrar los nombres de cada material en la gráfica. Se debe presionar el botón “Nombres” si se desea mostrar todos los nombres de los materiales visualizados en la gráfica (ver figura I.8.). Si se desea ubicar un material específico se debe realizar doble clic sobe el nombre del material en el árbol de materiales y el nombre del mismo aparecerá en la gráfica (ver figura I.9.).

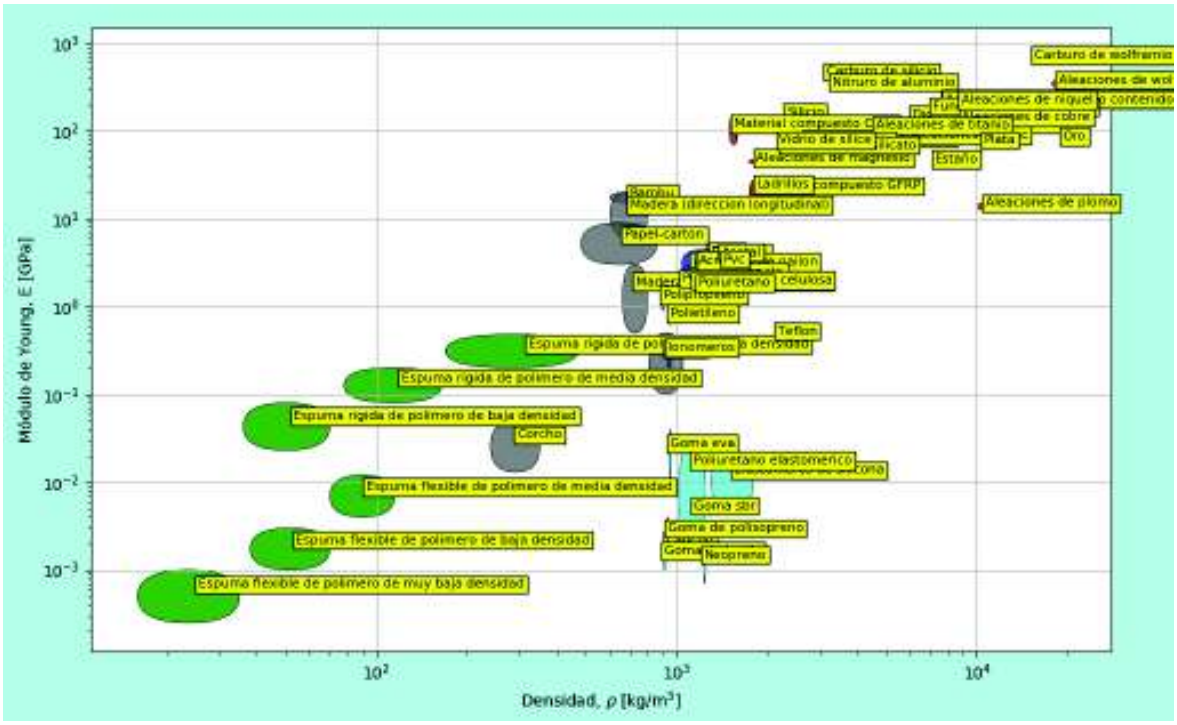

Figura I. 8. Visualización de los nombres de todos los materiales disponibles.
(Fuente: propia)

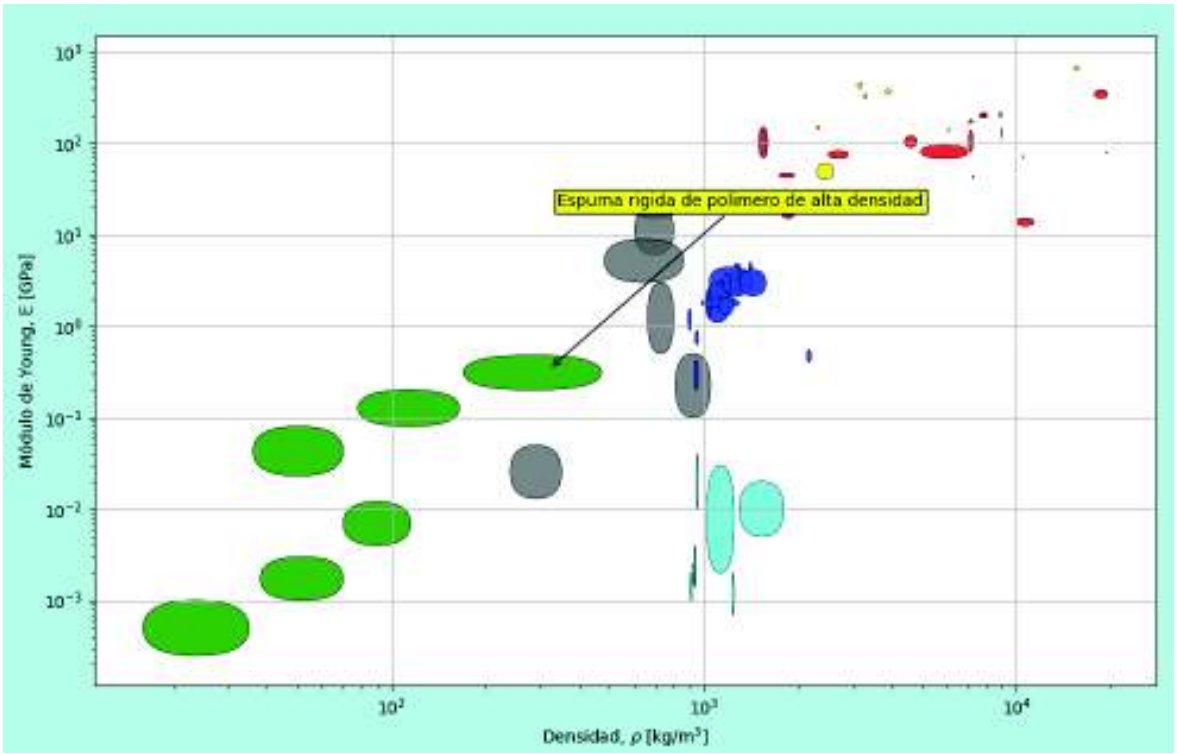

Figura I. 9. Visualización del nombre específico de un material de interés.
(Fuente: propia)

ANEXO V.

ÍNDICES DE RENDIMIENTO MÁS COMUNES PARA APLICACIONES DE INGENIERÍA MECÁNICA

Tabla I. 5. Diseño limitado por rigidez, objetivo mínima masa.

Elemento a tensión	
Rigidez Longitud especificada Área de sección libre	E/p
Viga (carga en flexión)	
Rigidez Longitud Forma especificada Área de sección libre	$E^{1/2}/p$
Rigidez Longitud Alto especificado Ancho libre	E/p
Rigidez Longitud Ancho especificado Altura libre	$E^{1/3}/p$
Elemento a compresión (columna)	
Carga de pandeo Longitud Forma específica Área de sección libre	$E^{1/2}/p$
Panel (Placa plana, carga a flexión)	
Rigidez Longitud Ancho especificado Espesor libre	$E^{1/3}/p$
Placa (Placa plana, compresión en el plano)	
Carga de colapso Longitud y ancho especificado Espesor libre	$E^{1/3}/p$
Cilindro con presión interna	

Distorsión elástica Presión y radio especificado Espesor de pared libre	E/p

(Fuente: [13])

Tabla I. 6. Diseño limitado por la resistencia para la mínima masa.

Elemento a tensión	
Rigidez Longitud especificada Área de sección libre	σ_f/p
Eje (carga a torsión)	
Carga Longitud Forma específica Área de sección libre	$\sigma_f^{2/3}/p$
Carga Longitud Radio externo especificado Espesor de pared libre	σ_f/p
Carga Longitud Espesor de pared especificado Radio externo libre	$\sigma_f^{1/2}/p$
Viga (Carga a torsión)	
Carga Longitud Forma específica Área de sección libre	$\sigma_f^{2/3}/p$
Carga Longitud Altura especificada Ancho libre	σ_f/p
Carga Longitud Ancho especificado Altura libre	$\sigma_f^{1/2}/p$
Elemento a compresión (columna)	

Carga Longitud Forma específica Área de sección libre	σ_f/p
Panel (Placa plana, carga a flexión)	
Rigidez Longitud Ancho especificado Espesor libre	$\sigma_f^{1/2}/p$
Placa (Placa plana, compresión en el plano)	
Carga de colapso Longitud y ancho especificado Espesor libre	$\sigma_f^{1/2}/p$
Cilindro con presión interna	
Distorsión elástica Presión y radio especificado Espesor de pared libre	σ_f/p
Casco esférico con presión interna	
Distorsión elástica Presión y radio especificado Espesor de pared libre	σ_f/p
Volantes (discos rotando)	
Máxima energía almacenada por unidad de volumen Velocidad dada	σ
Máxima energía almacenada por unidad de masa No falla	σ_f/p

(Fuente: [13])

Tabla I. 7. Diseño limitado por la resistencia para resortes para rendimiento máximo.

Resortes	
Máxima energía elástica almacenada por unidad de volumen No falla	σ_f^2/E
Máxima energía elástica almacenada por unidad de masa No falla	$\sigma_f^2/E\ p$
Articulaciones elásticas	
Radio de giro minimizado (máxima flexibilidad sin falla)	σ_f/E
Sellos de compresión y juntas selladoras	
Máxima conformabilidad Límite en la presión de contacto	$\sigma_f^{3/2}/E$ y $1/E$
Diafragmas	
Máxima deflexión bajo fuerza o presión especificada	$\sigma_f^{3/2}/E$
Tambores giratorios y centrífugos	
Máxima velocidad angular Radio fijo Espesor de pared libre	σ_f/p

(Fuente: [13])

Tabla I. 8. Diseño limitado por la vibración.

Elemento a tensión (tie) y columnas	
Máximas frecuencias de vibración longitudinales	E/p
Vigas	
Todas las dimensiones prescritas Máxima frecuencia de vibración de flexión	E/p
Longitud y rigidez prescrita Máxima frecuencia de vibración de flexión	$E^{1/2}/p$
Paneles	
Todas las dimensiones prescritas Máximas frecuencias de vibración longitudinales	E/p
Longitud, ancho y rigidez prescritas Máximas frecuencias de vibración longitudinales	$E^{1/3}/p$

(Fuente: [13])