Trabajo Fin de Máster
Máster Universitario en Ingeniería Aeronáutica

Desarrollo de software de interfaz gráfica para la predicción de vida a fatiga multiaxial

Autor: David García Serrano

Tutor: Carlos Navarro Pintado

Dpto. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster
Máster Universitario en Ingeniería Aeronáutica

Desarrollo de software de interfaz gráfica para la predicción de vida a fatiga multiaxial

Autor:
David García Serrano

Tutor:
Carlos Navarro Pintado
Catedrático de Ingeniería Mecánica

Dpto. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster: Desarrollo de software de interfaz gráfica para la predicción de vida a fatiga multiaxial

Autor: David García Serrano
Tutor: Carlos Navarro Pintado

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Sin duda alguna, son muchas las personas a las que quiero dedicar parte o la totalidad de este trabajo por su ayuda tanto en la realización del mismo como en el transcurso de todos mis estudios.
En primer lugar, quisiera agradecer al Catedrático de Ingeniería Mecánica D. Carlos Navarro Pintado su ayuda en la realización del mismo, tanto a la hora de guiarme en su desarrollo, como para cederme recursos y bibliografía para poder enriquecer más el contenido del trabajo.
Sobre todo tengo que mencionar en estos agradecimientos a mi familia y amigos, quienes han sido un soporte y un apoyo anímico durante toda mi vida académica, tanto durante el grado como durante el máster.

"Per Aspera Ad Astra"
David García Serrano

Coria, Extremadura, 2021

Resumen

La fatiga juega un papel muy importante dentro de los fenómenos relacionados con el desgaste de los materiales usados en muchos sectores industriales. Especialmente tiene una importante relevancia dentro del sector aeronáutico, la cual ha sido una de las mayores causantes de problemas estructurales cuando dicho fenómeno no ha sido tenido en cuenta en los cálculos para el diseño de las aeronaves. Como definición básica, podemos decir que este fenómeno se produce cuando una estructura un material se ve sometido a cargas fluctuantes en el tiempo, ya sean determinadas o aleatorias. Dichas cargas pueden producir microgrietas, Que con el paso del tiempo podrían desencadenar los condicionantes necesarios para una propagación más rápida de la grieta, pudiéndose producir un fallo catastrófico dentro de la estructura.

La estructura interna de los materiales metálicos, que son los que se utilizan en mayor medida para el diseño estructural de una aeronave, está conformada por gránulos diferentes. Por tanto, dos piezas producidas del mismo modo, el mismo acabado y el mismo material pueden llegar al fallo de fatiga en diferentes instantes, haciendo la predicción de este fenómeno aún más difícil. El fallo por fatiga no puede predecirse de manera exacta aunque puede aproximarse de manera estadística. Aunque los resultados de predicción para diferentes piezas sean dispersos, es indispensable desarrollar modelos de predicción que puedan calcular el momento en el que se va a producir el fallo. Esta predicción aproximada del punto de fatiga podría suponer una mejora en el diseño y en la seguridad de los componentes aeronáuticos.

Para la realización de este trabajo se ha utilizado un lenguaje de programación denominado Python. Este lenguaje es uno de los más fáciles de aprender dentro de los lenguajes de programación. Este trabajo es una continuación del trabajo realizado por Alejandro Quirós Rodríguez[14], quien desarrolló diferentes scripts para la estimación de la vida a fatiga del componente, diferenciando entre las diferentes fases por las que pasa la vida de la grieta.

Mi trabajo ha consistido en la mejora del código, intentándolo hacer más simple de comprender para cualquier desarrollador que se enfrente a él en un futuro. El punto más importante de este trabajo ha sido ,aparte de lo anterior, la implementación de un software con interfaz gráfica de usuario para el cálculo de la fatiga multiaxial, de forma que cualquier persona sin conocimientos de programación en Python pudiera calcular de forma rápida y sencilla los datos de estimación de la vida a fatiga.

Abstract

Fatigue plays a very important role in the phenomena related to wear and tear of materials used in many industrial sectors. In particular, it has significant relevance within the aeronautical sector, which has been one of the biggest causes of structural problems when this phenomenon has not been taken into account in the calculations for aircraft design.
As a basic definition, we can say that this phenomenon occurs when a structure a material is subjected to fluctuating loads over time, whether determined or random. Such loads can produce micro-cracks, which over time could trigger the conditions necessary for faster crack propagation, and catastrophic failure may occur within the structure.
The internal structure of metallic materials, which are the ones most commonly used for the structural design of an aircraft, consists of different granules. Therefore, two pieces produced in the same way, the same finish and the same material can reach fatigue failure at different times, making the prediction of this phenomenon even more difficult. Fatigue failure cannot be accurately predicted, although it can be statistically approximated. Although the prediction results for different parts are dispersed, it is essential to develop prediction models that can calculate when the failure will occur. This approximate fatigue point prediction could lead to an improvement in the design and safety of aeronautical components.
A programming language called Python has been used to perform this work. This language is one of the easiest to learn within programming languages. This work is a continuation of the work done by Alejandro Quirós Rodríguez[14], who developed different scripts for The estimation of life to fatigue of the component, differentiating between the different phases through which the life of the crack passes.
My job has been to improve the code, trying to make it easier for any developer to face it in the future. The most important point of this work has been, apart from the above, the implementation of a software with graphical user interface for the calculation of multi-axial fatigue, so that anyone without programming knowledge in Python could quickly and easily calculate life estimation data to fatigue.

Índice

Índice de Figuras

Índice de Tablas

Índice de Códigos

Notación

N	Fuerza normal
Q	Fuerza tangencial
σ	Tensión
GUI	Guide User Interface
R	Coeficiente de asimétria de carga, Matriz de rotación
FS,FS	Parámetro de Fatemi-Socie
$\Delta\gamma_{max}$	Incremento máximo de deformaciones tangenciales
k	Relación entre límite elástico y el límite a fatiga
σ_{max}	Tensión máxima
σ_y	Límite elástico
σ'_f	Límite de fatiga
SWT,SWT	Parámetro de Smith-Watson-Topper
$\Delta\varepsilon$	Incremento de deformaciones
E	Módulo de Young
N_t	Número de ciclos totales
b	Exponente de resistencia a fatiga
a	Longitud de grieta
m_1,m_2	Parámetros de los pesos
$w(s)$	Función de pesos propuesta por Bueckner
W	Anchura del espécimen
K_I	Factor de intensidad de tensiones de modo I
Φ	Factor de corrección del factor de tensiones
a/c	Relación entre la profundidad de la grieta y su radio
C	Coeficiente de la ley de crecimiento
ΔK	Incremento del factor de intensidad de tensiones
n	Exponente de la ley de crecimiento
$\Delta K_{th\infty}$	Umbral de crecimiento
f	Parámetro de aproximación al diagrama de Kitagawa-Takahashi
a_0	Parámetro de El Haddad
l_0	Distancia a la primera barrera microestructural
N_i	Número de ciclos de la fase de iniciación
N_p	Número de ciclos de la fase propagación
γ^c_{xy}	Deformación tangencial durante la fase compresión
γ^t_{xy}	Deformación tangencial durante la fase tracción
ε^c_x	Deformación normal durante la fase de compresión

ε_x^t	Deformación normal durante la fase de tracción
G	Módulo de cortadura
ν	Coeficiente de Poisson

1 Introducción

La fatiga es conocida por ser la principal causa de muchos fallos mecánicos. A pesar de este hecho los mecanismos producen la fatiga Están aún sin comprender. Esto es en parte debido a los complejas formas geométricas y/o a las complejas cargas de los componentes y estructuras ingenieriles que provocan estados cíclicos de tensión y de formación multiaxial en vez de uniaxial. Estructuras tales como turbinas de gas, plantas nucleares, estructuras aeronáuticas, vehículos terrestres... son ejemplos de este fenómeno.[7]

La fatiga por Fretting es un tipo de fatiga especial, que se produce cuando una pieza sometida a tracción y compresión está siendo sometida a su vez por cargas de contacto que generan una fricción y un esfuerzo mayor que limita aún más la vida a fatiga. Las cargas de contacto en cuestión podrían comportarse de una forma análoga a un concentrador de tensiones, lo que produce la aparición de microgrietas en la superficie de contacto. Durante este fenómeno las piezas se ven sometidas a dos cargas nuevas N y Q además de la carga σ fluctuante que provoca la fatiga tal y como se muestra en figura 1.1.[11]

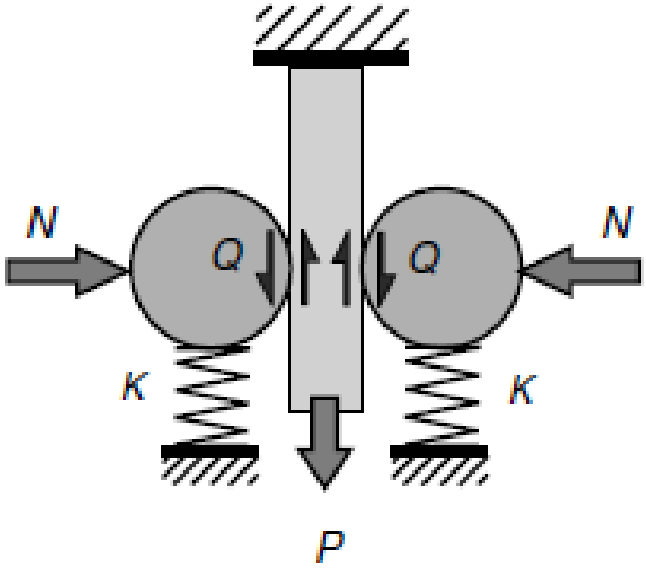

Figura 1.1 Ensayo de Fretting.[11].

Para cargas uniaxiales es más fácil estudiar la iniciación de las grietas durante la primera fase, puesto que se puede hacer uso de la curva S-N, que puede obtenerse de forma experimental. Dicha

curva relaciona las tensiones con el número de ciclos que se producen durante estados de carga uniaxiales. Por otra parte, la fatiga por *fretting* presenta un estado de tensiones multiaxial. Junto a este estado de tensiones también hay que tener en cuenta el fuerte gradiente que se produce cerca de la superficie del material; por lo que los métodos para la predicción de la vida fatiga en estado uniaxial no sirven para estas condiciones de carga. Existe una fase intermedia, denominada fase de propagación en la cual la grieta crece siguiendo la mecánica de la fractura a través de la Ley de Paris.

Los modelos de predicción debida a fatiga pueden suponer que toda la vida se produce dentro de la fase de iniciación, que toda la vida se produce dentro de la fase de propagación, o bien que existe un porcentaje de vida que se produce en una fase o en otra. En función de las diferentes fases podemos realizar unas suposiciones u otras. Por ejemplo, aquellos modelos en los que la fase de iniciación de la grieta es mayor que la fase de propagación podemos suponer que los modelos basados en la iniciación funcionan de una forma correcta. Sin embargo, este último modelo no tendría en cuenta aquellos fenómenos que se producen fuera de las zonas donde se están evaluando las tensiones y las deformaciones del material.

Por otra parte, a diferencia de los modelos basados en la iniciación, los modelos fundamentados en la propagación tienen la gran dificultad de tener que estimar la longitud a partir de la cual la grieta empieza a propagar. Dicha magnitud es función del tipo de ensayo o de la geometría, por lo que sería necesario la utilización de algún parámetro que permita definir esta longitud de iniciación de una forma idónea para los cálculos.

Por último, existen los modelos que combinan en mayor o menor medida tanto la iniciación como la propagación. El principal inconveniente de estos modelos, al igual que los basados en la propagación es la selección de una longitud de grieta a partir de la cual pasamos de la fase de iniciación a la fase propagación. Esta longitud de grieta es conocida como longitud de iniciación. Otros modelos más sencillos establecen una longitud fija, en función de un criterio determinado para simplificar los cálculos. En nuestro modelo en particular hemos utilizado una implementación que combina la fase de iniciación y la fase de propagación, sin tener en cuenta, en principio, la longitud de iniciación de la grieta.

El alcance de este trabajo es la implementación en lenguaje Python de un software de interfaz gráfica de usuario con el que poder predecir la vida a fatiga multiaxial y obtener resultados visuales de una forma fácil y rápida sin tener un conocimiento técnico de programación en este lenguaje. Los scripts utilizados por mi compañero Alejandro Quirós Rodríguez[14] han servido de referencia para la mejora de los diferentes códigos de implementación para el cálculo de la vida a fatiga multiaxial.

1.1 Python

Python es uno de los lenguajes de programación más sencillos de aprender. Es considerado un lenguaje de alto nivel, es decir la sintaxis del lenguaje es muy similar a la sintaxis del lenguaje natural. Este lenguaje esta administrado por Python Software Foundation© aunque tiene una licencia de código abierto con la que cualquier desarrollador puede acceder a él.
Python fue creado a finales de los ochenta, pero no ha sido hasta la década de los 2010 cuando ha sido utilizado en mayor medida, sobre todo para la estadística, más concretamente en la rama del data science[13].

Para el desarrollo de un proyecto normalmente se requieren más de un módulo diferente. Podemos definir módulo como una librería donde se encuentran diferentes funciones y variables específicas

que no se encuentran de forma nativa en Python y que son utilizadas para una función concreta dentro del programa que se está desarrollando. Para el software que hemos desarrollado se han utilizado varios módulos. Algunos de los que podemos destacar son los siguientes:

- NumPy: Es un módulo que permite el cálculo con vectores y matrices. Para usuarios habituales del lenguaje Matlab su uso es muy similar [1].
- SciPy: Es un módulo que importa funciones para la optimización de funciones, para el cálculo de extremos relativos [5].
- Pandas: Es un módulo especializado en la visualización de grandes cantidades de datos [2].
- Sklearn: Es un módulo que importa funciones para trabajos estadísticos como es el cálculo de regresión y otras operaciones de aprendizaje automático[4].
- Matplotlib: Es un módulo que se utiliza para la visualización de datos de forma gráfica [6].
- Re: Es un módulo utilizado para detectar es expresiones regulares y buscar dentro de cadenas de caracteres patrones repetidos [3].

1.2 Tkinter

El principal módulo utilizado en este proyecto es Tkinter. Este módulo se utiliza para crear ventanas con GUI (Guide User Interface) con Python. Dentro de este módulo existen diferente widgets que pueden utilizarse para interactuar con el usuario al igual que cualquier programa de escritorio de un ordenador. El desarrollo de este software no hubiera sido posible sin la ayuda de diferentes manuales para programación GUI, pues esta programación no se desarrolla dentro de una carrera de ingeniería aeronáutica[10].

2 Modelo de predicción de vida a fatiga

En este capítulo vamos a exponer el modelo teórico que se encuentra detrás de los cálculos de predicción de vida a fatiga que se han implementado dentro del programa desarrollado. Este modelo está basado en uno propuesto por C. Navarro[12] en el año 2005. Dicho modelo se basa en la diferenciación de dos fases dentro del proceso de fatiga, las cuales son la fase de iniciación y la fase de propagación. Según C. Navarro la vida total a fatiga para el caso del fretting sería la resultante de la suma de estas dos fases. La fase de iniciación es el proceso durante el cual el material que está viéndose sometido a una carga alternativa va produciendo una grieta apenas apreciable hasta que alcanza una cierta longitud denominada longitud de iniciación. Por otra parte, la fase de propagación es la que se produce después de la fase de iniciación y es cuando la grieta se propaga de una forma más rápida y descontrolada hasta que finalmente el material falla.

El proceso para calcular el número de ciclos necesarios para iniciar la grieta se lleva a cabo mediante la generación de un vector de valores discretos de longitudes de iniciación y a partir de la fase de propagación se calculan los ciclos necesarios para llevar el material a rotura. El resultado de este cálculo son las curvas de iniciación del material. Estas curvas son únicas para cada material y sólo dependen de las propiedades del material.

Las curvas de iniciación son utilizadas para determinar la vida de iniciación teniendo en cuenta el estado de tensiones y deformaciones al que está siendo sometido el material.

Una vez generados los resultados de iniciación para todas las longitudes de grieta se obtiene una tabla con la que se puede implementar una curva con la vida total, la vida de la iniciación y la vida a propagación. Con esta curva, es fácil obtener la vida a fatiga, pues el mínimo de esta curva representa el número máximo de ciclos que puede soportar dicho material para unas condiciones específicas.

Como se puede deducir, se trata de un proceso con multitud de cálculos, por lo que la carga computacional es importante. En las secciones posteriores se explican el proceso de cálculo en la fase de iniciación 2.1 y la fase propagación 2.2.

2.1 Fase de iniciación

Como ya hemos descrito anteriormente de una forma breve, la fase de iniciación se produce cuando material sometido a una carga alternante produce un debilitamiento de la estructura interna y externa del material que puede desencadenar en la iniciación de una grieta. El material a priori, puede estar en perfecto estado pero la propia naturaleza de las cargas alternantes es la que desencadena

defectos que pueden producir un concentrador de tensiones en una determinada zona. Aunque un concentrador de tensiones facilita la fase de iniciación, esta fase puede iniciarse incluso si no existen concentradores de tensiones apreciables en el material, es decir en una superficie perfectamente lisa. En nuestro caso concreto, los esfuerzos multiaxiales lo que producen son cargas tangenciales y normales a la superficie que aceleran el proceso de fatiga del material.

Hay que tener en cuenta que no existe un criterio único para la estimación de la iniciación. En nuestro caso hemos optado por los criterios propuestos por Fatemi-Socie[7] y Smith-Watson-Topper[17] los cuales serán descritos en las secciones 2.1.1 y 2.1.2.

A partir de estos parámetros, se obtienen diferentes curvas en función del parámetro de iniciación elegido. Estas curvas relacionan el parámetro elegido con la vida total en el caso de tensión uniaxial y ciclo de carga simétrico ($R = -1$). Para el cálculo de la longitud de iniciación de la grieta simplemente hay que restar la vida total anterior a los ciclos necesarios para propagación de la grieta desde dicha longitud de iniciación hasta el fallo por rotura. Las curvas de iniciación son obtenidas para el caso uniaxial donde la tensión uniforme. Como ya hemos mencionado anteriormente, las curvas de iniciación únicamente dependen de las propiedades del material y no del estado del mismo en los experimentos. Por tanto, un determinado material tendrá unas curvas de iniciación específicas.

2.1.1 Criterio de Fatemi-Socie

El primer criterio que vamos a describir es el propuesto por Fatemi y Socie. Dicho criterio hace uso del incremento de deformaciones tangenciales dentro del plano donde dichas deformaciones son máximas ($\Delta\gamma_{max}$) y de la tensión normal máxima(σ_{max}) que es perpendicular al plano donde se produce esas deformaciones tangenciales. En la ecuación 2.1 se define el parámetro de Fatemi-Socie.

$$FS = \frac{\Delta\gamma_{max}}{2}\left(1 + k\frac{\sigma_{max}}{\sigma_y}\right) \tag{2.1}$$

Donde k es la relación entre el límite elástico (σ_y) y el límite a fatiga (σ'_f) de forma que $k = \sigma_y/\sigma'_f$. Esta expresión es válida para el caso multiaxial. Para el caso uniaxial para ciclos simétricos y haciendo uso del cálculo elástico de tensiones es necesario modificar la expresión anterior y hacer uso de la siguiente, tal y como se muestra en la ecuación 2.2.

$$FS = (1+\nu)\frac{\sigma'_f}{E}(2N_t)^b + \frac{k}{2}(1+\nu)\frac{\sigma'^2_f}{E\sigma_y}(2N_t)^{2b} \tag{2.2}$$

Donde ν es el coeficiente de Poisson, E es el módulo de Young, b es el denominado exponente de resistencia a fatiga,N_t es el número de ciclos hasta la rotura . Aplicando la expresión 2.2 podemos obtener la curva $FS - N_t$ representada en la figura 2.1. Se puede apreciar como para altos números de ciclos apenas existen diferencias apreciables entre unas longitudes de iniciación y otras. Lo contrario ocurre si hablamos de bajo número de ciclos, donde la longitud de iniciación sí influye significativamente en la vida a fatiga.

2.1.2 Criterio de Smith-Watson-Topper

Otro criterio para la iniciación de la grieta es el propuesto por Smith et al. Dicho criterio tiene en cuenta el incremento que se produce en la deformación ($\Delta\varepsilon$) y la tensión normal máxima (σ_{max}) que se alcanza en el material. Este parámetro se calcula para la orientación en la que el producto

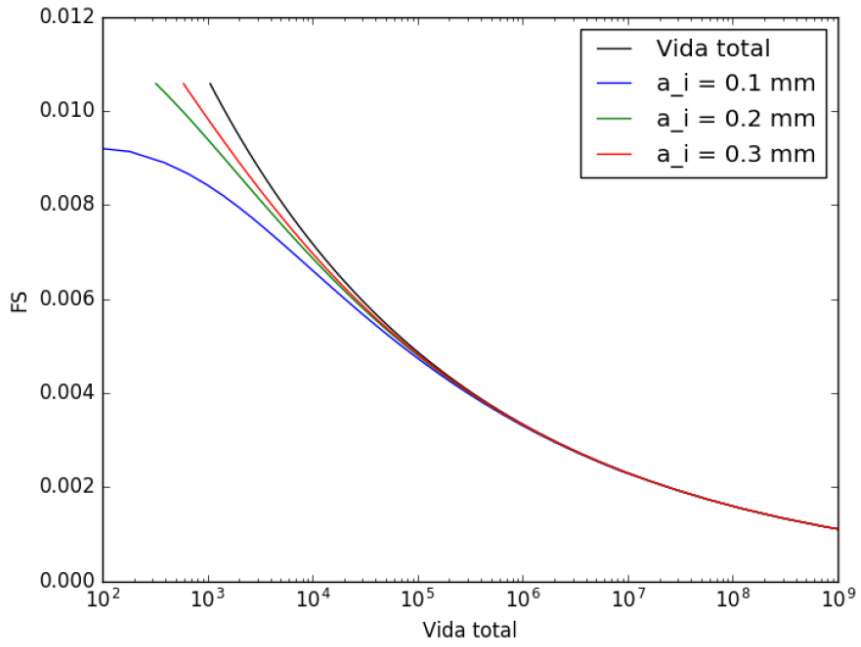

Figura 2.1 Curvas de iniciación $FS - N_i$[14].

entre ambos valores es máximo y por tanto, dicho parámetro también lo es.

$$SWT = \left(\sigma_{max}\frac{\Delta\varepsilon}{2}\right) \tag{2.3}$$

La expresión 2.3 representa el caso multiaxial. Para el caso de fatiga uniaxial con ciclos simétricos y teniendo en cuenta la hipótesis de que el cálculo de tensiones es elástico, podemos hacer uso de la expresión 2.4.

$$SWT = \frac{\sigma_f'^2}{E}(2N_t)^{2b} \tag{2.4}$$

De forma análoga, podemos obtener la curva que representa la variación de el parámetro SWT con respecto a los ciclos totales N_t en la figura 2.2.

El comportamiento de la gráfica es similar a lo que ocurre con el criterio de Fatemi-Socie, donde para ciclos altos no existen apenas diferencias apreciables para distintos longitudes de iniciación y para ciclos bajos, estas diferencias son más apreciables.

Todos estos cálculos vienen implementados en el script correspondiente a la fase de propagación que están en el apéndice A.2 al final de este documento.

2.2 Fase de propagación

La siguiente fase en el crecimiento de una grieta es la fase de propagación. En esta fase la grieta normalmente se propaga más rápidamente de lo que lo hace durante la fase de iniciación. Para los cálculos que se realizan en esta fase se hace uso de la teoría propuesta en la mecánica de la fractura. Puesto que a priori, no conocemos la longitud inicial, esta se define con anterioridad. Una

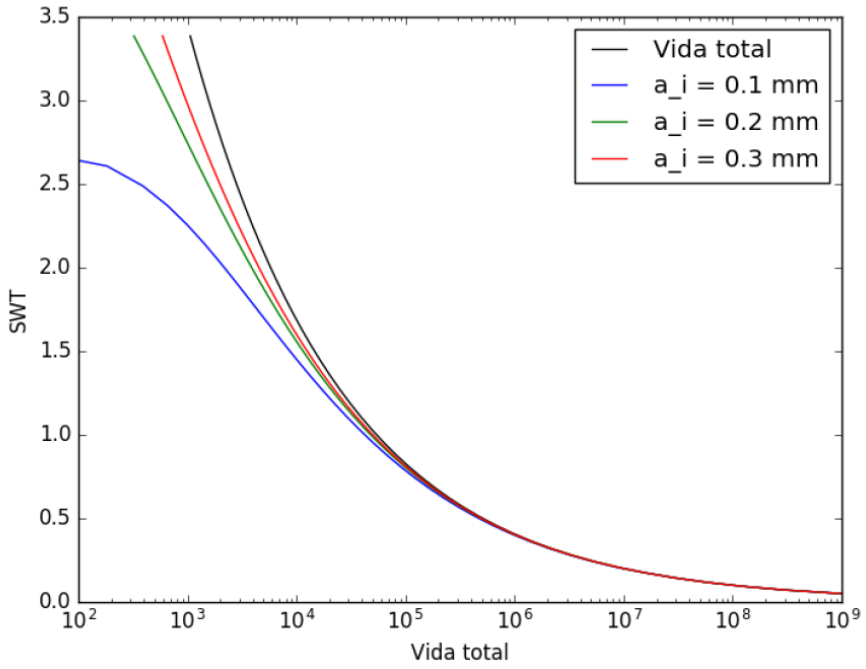

Figura 2.2 Curvas de iniciación $SWT - N_i$[14].

buena aproximación para hacer los cálculos más sencillos es suponer que la grieta crece en línea recta. Esta suposición, si bien no es cierta para toda la vida, sí lo es en la mayor parte de esta. Otra hipótesis importante es suponer que el factor de intensidad de tensiones 2.2.1 es principalmente en modo I y despreciar el modo II. A continuación se describen todas las expresiones utilizadas para el cálculo de esta fase.

2.2.1 Factor de intensidad de tensiones

Dentro de la mecánica de la fractura resulta fundamental la obtención del llamado factor de intensidad de tensiones. Esté factor determina la tensión que se produce en las cercanías de un concentrador de tensiones con respecto a la tensión nominal. Para el cálculo de este factor se hace uso de una función de peso propuesta por Bueckner[16] que es función de la profundidad de la grieta.

$$w(s) = \frac{1}{\sqrt{s}}\left(1 + m_1 \cdot \left(\frac{s}{a}\right) + m_2 \cdot \left(\frac{s}{a}\right)^2\right) \tag{2.5}$$

En la expresión 2.5, a representa la longitud de la grieta, s es la variable que representa la posición hasta dicha longitud de la grieta y m_1 y m_2 son dos parámetros obtenidos de forma empírica y que dependen de a y de W, el ancho del espécimen.

$$m_1 = 0.6147 + 17.1844\left(\frac{a}{W}\right)^2 + 8.7822\left(\frac{a}{W}\right)^6 \tag{2.6}$$

$$m_2 = 0.2502 + 3.2889\left(\frac{a}{W}\right)^2 + 70.0444\left(\frac{a}{W}\right)^6 \tag{2.7}$$

Teniendo los parámetros para definir la función de pesos podemos calcular factor de intensidad de tensiones correspondiente al modo I haciendo uso de la siguiente expresión.

$$K_I = \sqrt{\frac{2}{\pi}} \int_0^a w(s) \cdot \sigma_x(s) ds \tag{2.8}$$

Donde $\sigma_x(s)$ representa la tensión normal al plano de la grieta en función de la posición. La expresión 2.8 es válida para grietas bidimensionales o planas. Por tanto es necesario corregir el factor de intensidad de tensiones para grietas tridimensionales o elípticas, que son las que se producen cuando se ven sometidas a *fretting*. Para esta corrección se hace uso del factor Φ determinado por Irwin[9] en 1962.

$$\Phi = \int_0^{\pi/2} \sqrt{1 - \left(1 - \left(\frac{a}{c}\right)^2\right) \sin^2 \phi} d\phi \tag{2.9}$$

Como se puede observar en la expresión 2.9 el factor Φ depende de la relación a/c, es decir de la relación de aspecto. Si asumimos que la grieta puede considerarse elíptica, a es la profundidad de penetración en el material y c es la longitud del semieje de la elipse que se observa en la superficie. Se puede ver mejor esta geometría en la figura 2.3. El factor de intensidad de tensiones se corrige para una grieta elíptica, dividiéndolo por el factor ϕ.

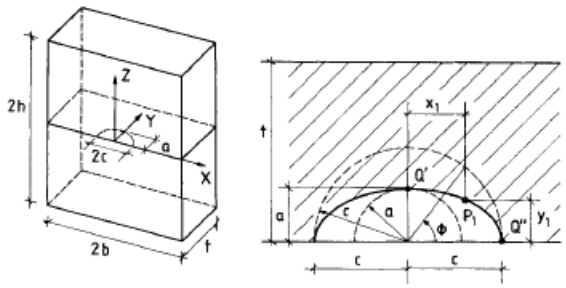

Figura 2.3 Esquema de una grieta tridimensional[8].

2.2.2 Modelo de propagación

Dentro de la mecánica de la fractura es necesario definir el modelo que determina la propagación de la grieta. Existen diferentes modelos para el crecimiento de grieta, uno de los más conocidos es el basado en la Ley de Paris, cuya expresión es la siguiente:

$$\frac{da}{dN} = C\Delta K^n \tag{2.10}$$

Como se observa, se trata de una ecuación diferencial donde ΔK, incremento del factor de intensidad, es función de a. En la ecuación anterior n es el exponente de la ley de crecimiento. Sin embargo la expresión 2.10 es muy simplificada y sólo es aplicable a condiciones muy ideales. A partir de esta ley se ha desarrollado otra expresión donde se tienen en cuenta el estado estructural del material[12].

$$\frac{da}{dN} = C\left(\Delta K^n - \left(\Delta K_{th\infty}\left(\frac{a^f}{a^f + a_0^f - l_0^f}\right)^{1/2f}\right)^n\right) \tag{2.11}$$

Donde $\Delta K_{th\infty}$ es el denominado umbral de crecimiento, f es un parámetro de aproximación al diagrama de Kitagawa-Takahashi, a_0 es el parámetro de El Haddad[1] y l_0 es la distancia a la primera barrera microestructural. En la figura 2.4 se puede observar el diagrama de Kitagawa-Takahashi y como se define el parámetro de El Haddad.

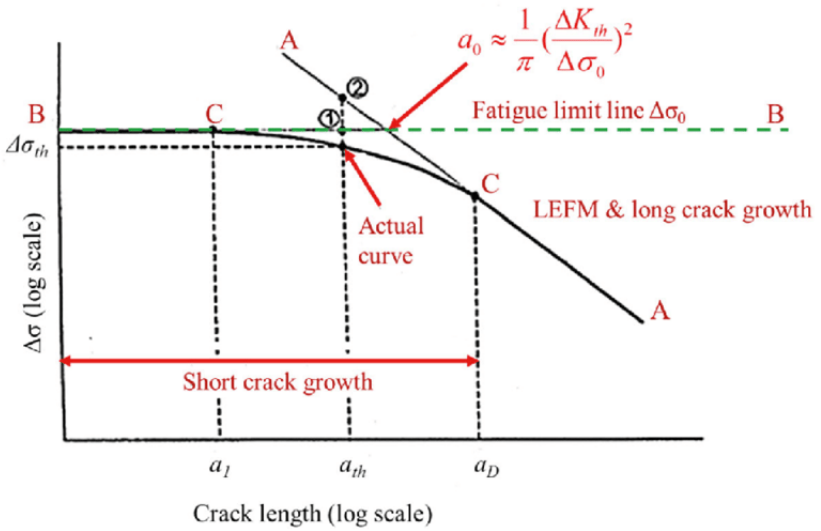

Figura 2.4 Diagrama de Kitagawa-Takahashi[15].

Todos los cálculos que se describen en esta sección se implementan en el código descrito en el anexo A.3, que contiene los cálculos para la fase de propagación.

2.3 Cálculo de la vida a fatiga

Una vez vistas las diferentes fases del crecimiento de una grieta, nos queda sumar ambas contribuciones para obtener los ciclos totales. Se obtiene una curva de ciclos totales en la cual el mínimo representa el número máximo de ciclos que el material es capaz de aguantar en esas determinadas condiciones.

$$N_t = N_i + N_p \tag{2.12}$$

Donde N_t, N_i y N_p son los ciclos totales, de iniciación y de propagación respectivamente. Antes de obtener la curva de vida a fatiga es necesario hacer una serie de cálculos previos:

2.3.1 Rotación de la matriz de tensiones

Para obtener los resultados en los ejes idóneos es necesario hacer transformaciones geométricas para rotar la matriz de tensiones. Normalmente los datos vienen dados en un vector de tensiones como este:

$$(\sigma_x, \sigma_y, \sigma_z, \sigma_{xy}, \sigma_{xz}, \sigma_{yz})$$

Este vector se debe representar como una matriz de tensiones, como la siguiente:

[1] Es una distancia ficticia que es utilizada para poder igualar el comportamiento de una grieta pequeña y una grieta grande.

$$\begin{pmatrix} \sigma_x & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z \end{pmatrix}$$

De forma análoga ocurre con las deformaciones, las cuales deben transformarse en una matriz para poder operar con ellas.

Las matrices de rotación en los respectivos ejes son las siguientes:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_x) & -\sin(\alpha_x) \\ 0 & \sin(\alpha_x) & \cos(\alpha_x) \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos(\alpha_y) & 0 & -\sin(\alpha_y) \\ 0 & 1 & 0 \\ \sin(\alpha_y) & 0 & \cos(\alpha_y) \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos(\alpha_z) & -\sin(\alpha_z) & 0 \\ \sin(\alpha_z) & \cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La matriz de rotación en los tres ejes se obtiene por el producto matricial de las matrices de rotación en cada eje:

$$R = R_x \times R_y \times R_z = \begin{pmatrix} C\alpha_y C\alpha_z & -S\alpha_z C\alpha_y & S\alpha_y \\ -S\alpha_x S\alpha_y C\alpha_z + S\alpha_z C\alpha_x & S\alpha_x S\alpha_y S\alpha_z + C\alpha_x C\alpha_z & -S\alpha_x C\alpha_y \\ S\alpha_x S\alpha_z + S\alpha_y C\alpha_x C\alpha_z & S\alpha_x C\alpha_z - S\alpha_y S\alpha_z C\alpha_x & C\alpha_x C\alpha_y \end{pmatrix} \quad (2.13)$$

Donde C representa el coseno, S el seno y α_i el ángulo de rotación con respecto a un eje i.
Para rotar una matriz dados unos ángulos α simplemente hay que aplicar la siguiente expresión:

$$M' = R^T \times M \times R \quad (2.14)$$

2.3.2 Cálculo de los parámetros

Para el cálculo de los parámetros de Fatemi-Socie 2.1.1 y Smith-Watson-Topper 2.1.2 es necesario hacer uso de las matrices de rotación 2.3.1 para transformar las matrices de tensiones y deformaciones. El proceso de obtención de los parámetros no es tan sencillo como aplicar las diferentes expresiones. Es necesario calcular los ángulos α que maximizan el parámetro en cuestión.

Por una parte, para el parámetro de Fatemi-Socie es necesario obtener la máxima amplitud de deformaciones tangenciales. Esto es $\Delta\gamma = \gamma^t_{xy} - \gamma^c_{xy}$,es decir la diferencia entre la deformación tangencial en la tracción y en la compresión.
Posteriormente, es necesario rotar la matriz de tensiones con los ángulos obtenidos al maximizar la diferencia en deformaciones tangenciales. El valor que nos interesa de esta matriz es la tensión normal correspondiente a $\sigma_{33} = \sigma_{max}$. Haciendo uso de la ecuación 2.1 podemos obtener el parámetro de Fatemi-Socie.

En el caso de la obtención del parámetro de Smith-Watson-Topper es necesario maximizar la cantidad $\sigma_{max} \cdot (\varepsilon^t_x - \varepsilon^c_x)/2$ de la ecuación 2.3, donde σ_x corresponde a la tensión que se produce en el eje longitudinal de la grieta. Al igual que ocurre con el parámetro de Fatemi-Socie es necesario rotar el tensor de tensiones y deformaciones para determinar el valor máximo de este parámetro que

se utilizará en posteriores cálculos.

Se puede ver un ejemplo de la curva de vida a fatiga en la figura 2.5,donde el punto rojo representa el número de ciclos máximos que puede soportar el material de forma teórica.

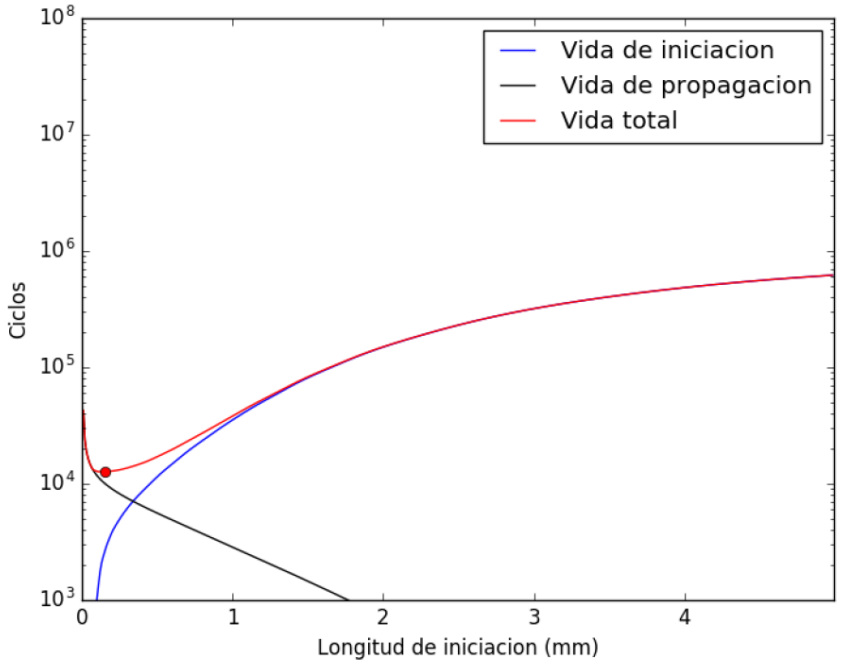

Figura 2.5 Ejemplo de curva de vida a fatiga[14].

Todos los cálculos explicados en esta sección se encuentran implementados en el código descrito en el anexo A.4 donde se calcula la vida a fatiga.

2.4 Propiedades de los materiales

En esta sección se describen las propiedades que caracterizan un material para determinar su vida a fatiga. Éstas son enumeradas en la tabla 2.1:

Tabla 2.1 Tabla de propiedades de los materiales.

Coeficiente de la ley de crecimiento	C
Exponente de la ley de crecimiento	n
Parámetro en aproximación al diagrama Kitagawa-Takahashi	f
Umbral de crecimiento	$\Delta K_{th\infty}$
Tenacidad a fractura	K_{IC}
Módulo de Young	E
Límite elástico	σ_y
Coeficiente de Poisson	ν
Exponente de resistencia a fatiga	b
Coeficiente de resistencia a fatiga	σ'_f
Límite de fatiga	$\Delta\sigma'_{fl}$
Módulo de cortadura	G
Parámetro de El Haddad	a_0
Distancia de la superficie a la primera barrera microestructural	l_0

Algunas de las propiedades enumeradas en la tabla anterior se puede obtener de forma empírica. Existen dos propiedades que derivan de otras propiedades. Éstas son el parámetro de El Haddad a_0 y el módulo de cortadura G. Ambos se calculan respectivamente de la siguiente forma:

$$a_0 = \frac{1}{\pi}\left(\frac{\Delta K_{th\infty}}{\sigma'_{fl}}\right) \tag{2.15}$$

$$G = \frac{E}{2(1+\nu)} \tag{2.16}$$

3 Interfaz Gráfica de Usuario (GUI)

Una vez que conocemos todas las expresiones necesarias para llevar a cabo el cálculo de la predicción de vida a fatiga que hemos visto en detalle en el capítulo 2 estamos en disposición de diseñar un software de interfaz gráfica de usuario (GUI), de forma que el usuario pueda realizar estos cálculos sin necesidad de conocer la programación ni la teoría del problema.

Como ya hemos mencionado anteriormente, la aplicación de escritorio se ha implementado con el módulo de Python llamado Tkinter 1.2, el cual contiene las funciones y métodos necesarios para crear los widgets que interactúen con el usuario. Una aplicación creada con este módulo tiene el mismo aspecto que cualquier aplicación de escritorio de cualquier ordenador convencional. Es necesario comentar que el software ha sido diseñado dentro de un sistema operativo Windows ®y por lo tanto es posible que el programa no funcione correctamente en cualquier otro sistema operativo.

El programa se ha dividido en cinco pestañas diferente con las que usuario puede realizar diferentes funciones. Estas pestañas serán desarrolladas en profundidad en secciones independientes, donde se explicará el funcionamiento y los posibles errores que se producen dentro de cada pestaña. Éstas son las siguientes:

- **Materiales**: En esta pestaña se especifican las propiedades del materiales que será utilizado durante los cálculos. Se desarrolla en mayor profundidad en la sección 3.2.
- **Iniciación**: En esta pestaña se realizan el cálculo de las curvas de iniciación para un material específico y unos parámetros específicos. Se desarrolla en mayor profundidad en la sección 3.3.
- **Datos**: En esta pestaña se cargan y se visualizan los datos de tensiones y deformación que se utilizarán para el cálculo de la vida a fatiga. Se desarrolla en mayor profundidad en la sección 3.4.
- **Cálculo**: En esta pestaña se realiza el cálculo de la vida a fatiga para unos datos de tensiones y deformaciones específicos. Se desarrolla en mayor profundidad en la sección 3.5.
- **Gráficas**: En esta sección se visualizan los resultados obtenidos y se comparan con los obtenidos experimentalmente. Se desarrolla en mayor profundidad en sección 3.6.

El código utilizado para el desarrollo de este programa se encuentra descrito en el anexo A.1, al final de este documento.

3.1 Inicio

Antes de describir las diferentes pestañas del programa es necesario exponer algunos aspectos generales del programa.

El primero de ellos es informar al usuario que utiliza el programa por primera vez que se trata de un software que se encuentra en fase de desarrollo y que por tanto pueden existir fallos de ejecución. Además de lo anterior también se le indica que puede acudir al presente documento para informarse de su funcionamiento. Esta información se le transmite al usuario mediante una ventana emergente que aparece al inicio del programa y pulsando en el menú "Acerca de" situado en la parte superior del programa. En la figura 3.1 se observa una captura de esta ventana al inicio del programa.

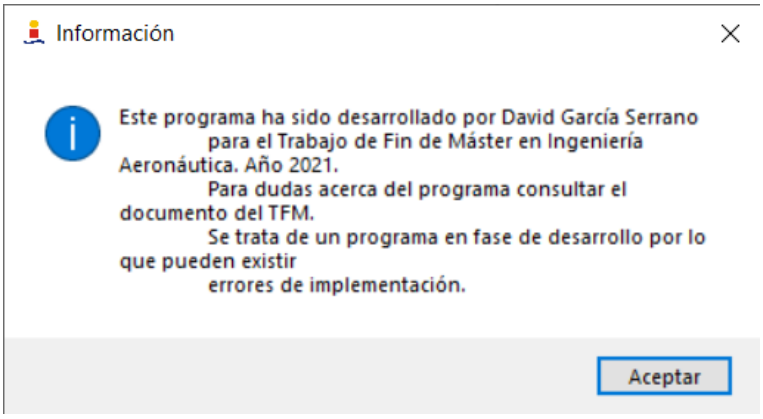

Figura 3.1 Ventana emergente de información al inicio.

Otro aspecto importante dentro del programa, es que existe un menú en la parte superior del programa que tiene diferentes opciones, tal y como se puede ver en la figura 3.2. A continuación se describen las diferentes opciones dentro del menú **Archivo** en diferentes subsecciones:

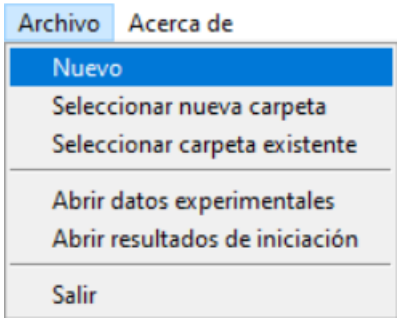

Figura 3.2 Menú de opciones.

3.1.1 Nuevo

Esta opción permite al usuario reiniciar el programa borrando todos los datos almacenados anteriormente. El usuario debe asegurarse de que ha guardado los resultados que quiere antes de presionar esta opción. Cuando se reinicia el programa, por defecto la carpeta de ejecución es donde se encuentra instalado el programa.

3.1.2 Seleccionar nueva carpeta

Esta opción abre una ventana de diálogo preguntando al usuario para seleccionar una determinada carpeta nueva donde realizar los correspondientes cálculos. La ventana de diálogo que le aparece al usuario es como la que se muestra en la figura 3.3.

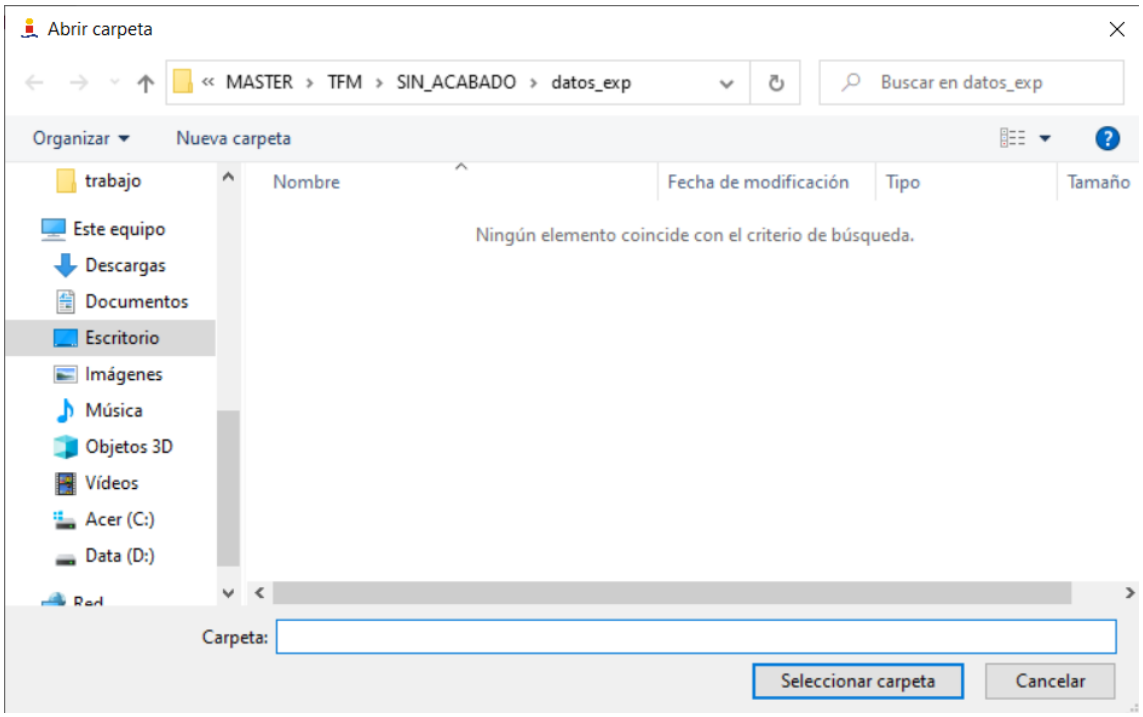

Figura 3.3 Ventana de diálogo para abrir una carpeta nueva..

Al abrir esta carpeta se crearan las diferentes subcarpetas necesarias para la ejecución del programa y para guardar los datos y gráficas obtenidas durante su ejecución. Esta opción es recomendable cuando pretendemos cambiar de material o de parámetros para que no se sobreescriban los resultados anteriores. Estas carpetas se muestran en la figura 3.4.

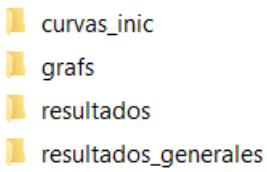

Figura 3.4 Subcarpetas creadas tras la selección de una nueva carpeta.

3.1.3 Seleccionar carpeta existente

Esta opción del menú permite al usuario abrir una carpeta existente con las subcarpetas necesarias para la ejecución y almacenamiento de los datos y resultados. Estas carpetas son las que se muestran en la figura 3.4. Estas carpetas pueden contener cálculos previos, por lo que es idónea para continuar con los cálculos de un mismo material y con unos mismos parámetros sin empezar de nuevo con los cálculos. La ventana de diálogo que le aparece al usuario es como la que se muestra en la figura 3.3.

3.1.4 Abrir datos experimentales

Esta opción del menú permite al usuario abrir la ubicación de los datos de tensiones y deformaciones necesarios para el cálculo de la estimación de la vida a fatiga. Estos datos deben estar dentro de esta carpeta en un formato *.dat* para que puedan ser procesados por el programa. La ventana de diálogo que le aparece al usuario tras ejecutar esta opción es la que se muestra en la figura 3.3. Si los archivos no fueran válidos o se cancelara la apertura de la carpeta se desplegará un mensaje de error indicando que no se han cargado los archivos de datos. El mensaje de error es el que se muestra en la figura 3.5.

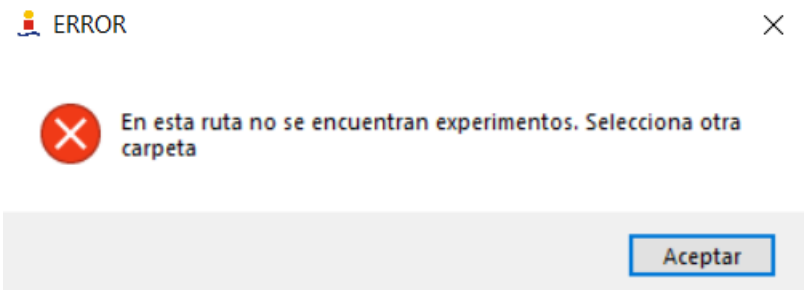

Figura 3.5 Mensaje de error de carga de datos.

Se pueden abrir los datos en esta opción o bien dentro de la pestaña **Datos** 3.4 en el botón **Cargar Datos**.

3.1.5 Abrir resultados de iniciación

Esta opción permite al usuario seleccionar un archivo de iniciación que haya sido realizado con anterioridad de forma independiente a este programa. Esta opción sustituye el archivo de iniciación que ejecutaría el programa por uno seleccionado por el usuario. La ventana que se despliega tras ejecutar esta opción es como la que se muestra en la figura 3.6. El programa sólo será capaz de leer archivos *.dat* por lo que la ventana de diálogo automáticamente filtrará los archivo por este tipo.

Si en el momento de la carga de los datos de iniciación el usuario decide cancelar la carga o ha seleccionado un archivo que no es válido para la realizar los cálculos de vida a fatiga el programa mostrará un error como el que se muestra en la figura 3.7.

3.1.6 Salir

La última opción del menú es la de salir del programa. Tras pulsar esta opción aparece un mensaje preguntando al usuario si quiere salir del programa. Este mensaje es como el que aparece en la figura 3.8.

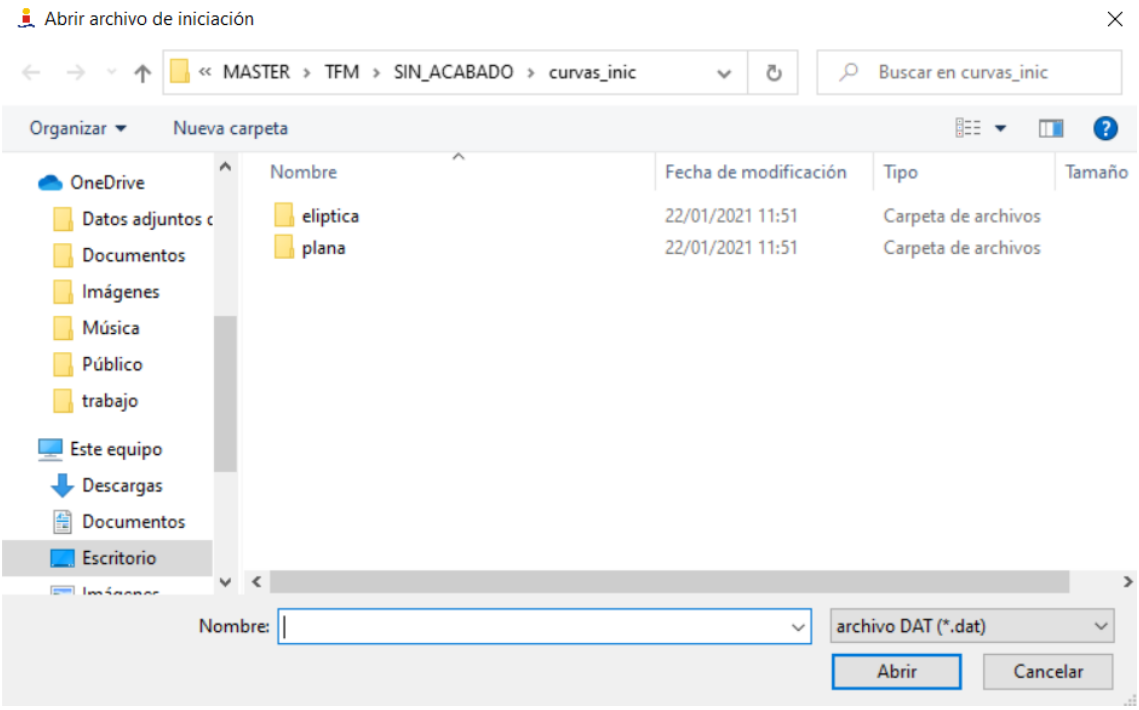

Figura 3.6 Ventana Abrir archivo de iniciación.

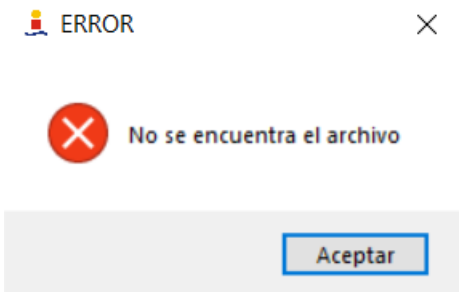

Figura 3.7 Error al cargar el archivo de iniciación.

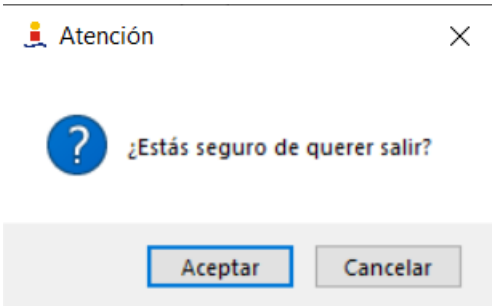

Figura 3.8 Mensaje al salir.

Si el usuario pulsa la opción *Aceptar* el programa se cerrara. En caso de pulsar la opción *Cancelar*, éste volverá al programa.

3.2 Pestaña de Materiales

La primera pestaña que vamos a desarrollar es la pestaña **Material**. En esta pestaña se establecen las propiedades necesarias para el cálculo de la vida a fatiga. Las propiedades en cuestión son las que se han visto en la sección 2.4 donde se indican cuales son éstas.

Esta pestaña es la primera que el usuario se encuentra al abrir el programa. En la figura 3.9 se muestra una captura de esta pestaña sin introducir ejecutar ninguno de los botones que aparecen en la parte inferior.

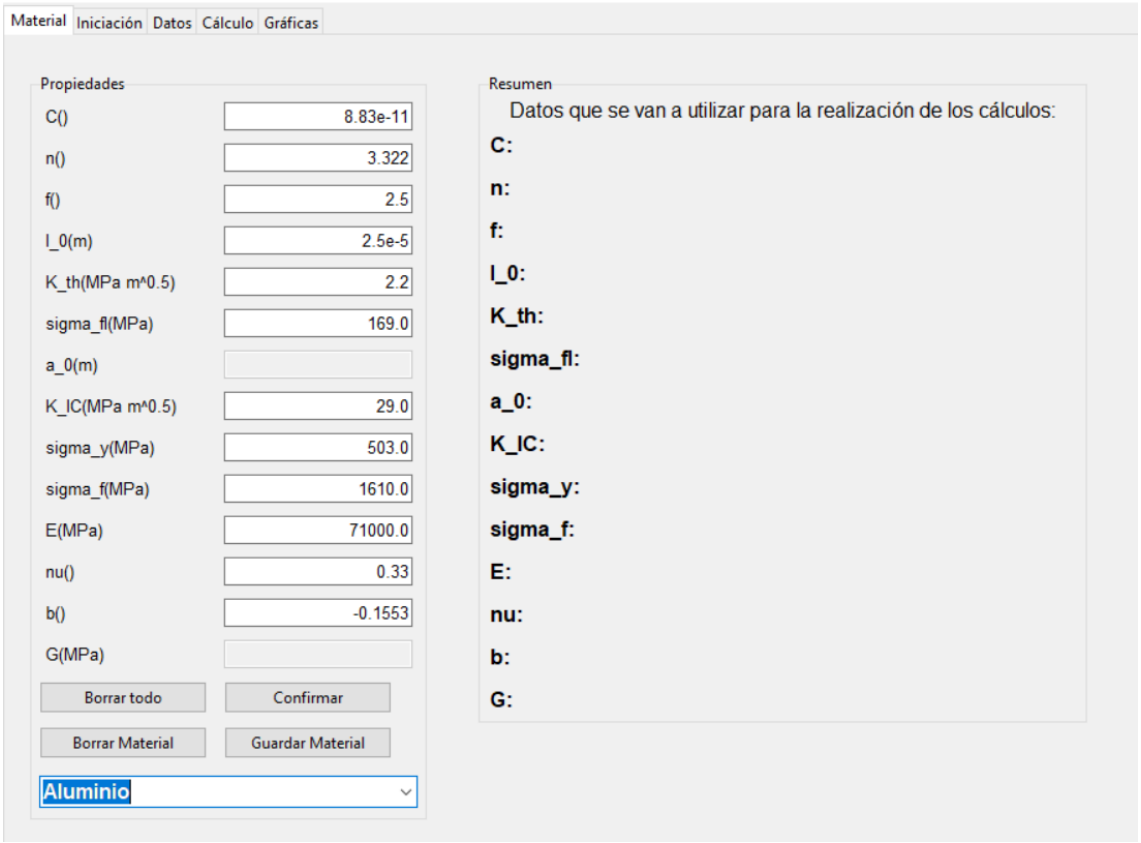

Figura 3.9 Captura de la pestaña Materiales sin confirmar los campos.

Como se observa en la figura 3.9, podemos ver dos cuadros llamados **Propiedades** y **Resumen**. El primero de ellos esta formado por 14 campos de introducción de datos correspondientes a las propiedades. El segundo indica al usuario cuáles son los datos de las propiedades del material que se van a usar para los cálculos posteriores.

En el cuadro **Propiedades** se puede ver como dos campos se encuentran sombreados. Estos campos corresponden al parámetro de El Haddad (a_0) y al parámetro del módulo de cortadura (G). Se encuentran sombreados debido a que dependen de otras propiedades y por tanto no pueden introducirse de forma manual por parte del usuario. Una vez rellenados todos los demás campos de forma correcta estos dos campos serán rellenados automáticamente mediante las expresiones indicadas en la sección 2.4 tras pulsar el botón **Confirmar**. Si los datos introducidos en los campos son correctos, éstos se indicarán en el cuadro **Resumen** tal y como se muestra en la figura 3.10.En el caso de que el valor de una determinada propiedad no sea valida, el programa borrará el campo

en cuestión y el usuario tendrá que introducir de nuevo un valor válido.

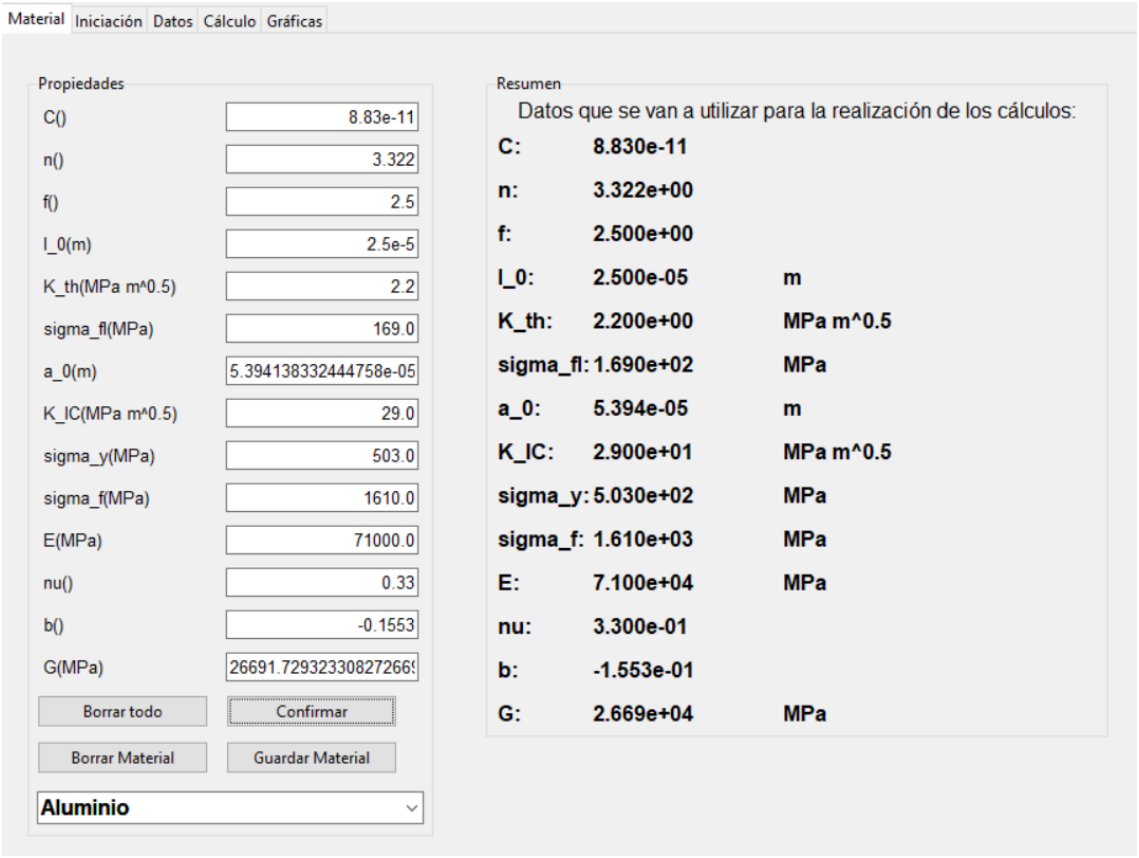

Figura 3.10 Captura de la pestaña tras confirmar los datos del material.

El usuario tiene la opción de borrar el contenido de todos los campos y de la memoria del programa pulsando el botón **Borrar todo** haciendo que se reinicien los valores.

Por otra parte, también es posible seleccionar un material cuyas propiedades han sido guardadas y rellenar todos los campos de las propiedades de forma automática. Estos materiales guardados están dentro de la lista situada en la parte inferior. Tras seleccionar el material, el usuario simplemente tendrá que confirmar las propiedades del material sino quiere hacer ninguna modificación.

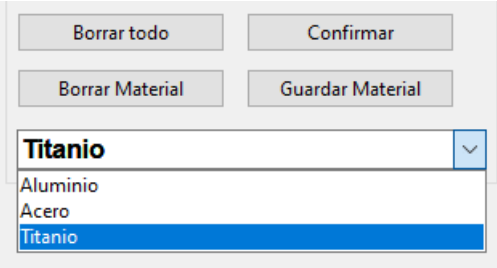

Figura 3.11 Lista de materiales guardados.

Como se observa en la figura 3.11 existen otros dos botones. Por una parte, el botón **Guardar Material** permite guardar las propiedades introducidas en los campos y asignarlas un nombre de

un material. Éste será añadido al final de la lista para que pueda ser seleccionado en un futuro. Por otra parte, el botón **Borrar Material** elimina un material seleccionado de la lista.
Para realizar las operaciones de guardar o eliminar un determinado material el programa carga un archivo interno llamado *materiales.xlsx* cada vez que se inicia en el cual se encuentran almacenados los datos de los materiales. Este archivo tiene un aspecto como el que se muestra en la figura 3.12.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		**Material**	**C**	**n**	**f**	**l_0**	**K_th**	**sigma_fl**	**K_IC**	**sigma_y**	**sigma_f**	**E**	**nu**	**b**
2	**0**	Aluminio	8.83E-11	3.322	2.5	0.000025	2.2	169	29	503	1610	71000	0.33	-0.1553
3	**1**	Acero	8.83E-11	3.322	2.5	0.000025	2.2	169	29	503	1610	72000	0.33	-0.1553
4	**2**	Titanio	8.81E-11	3.31	2.5	0.000025	2.2	186	42	503	1610	72000	0.33	-0.1553

Figura 3.12 Captura del archivo *materiales.xlsx*.

3.3 Pestaña de Iniciación

Una vez definidas las propiedades del material en la pestaña **Material** 3.2, estamos en disposición de obtener las curvas y los datos de iniciación para esas propiedades específicas del material. En la figura 3.13 se muestra una captura de esta pestaña a primera vista.

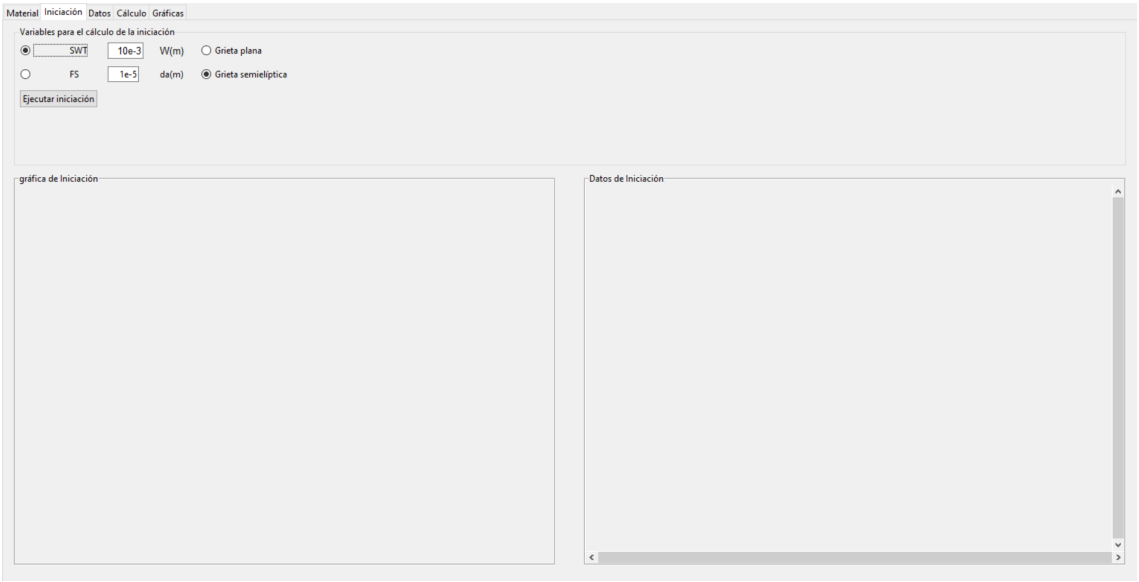

Figura 3.13 Captura de la pestaña Iniciación a primera vista.

Podemos observar en la figura 3.13 tres cuadros diferentes dentro de esta pestañas. En la parte superior, en el cuadro denominado **Variables para el cálculo de la iniciación**,se establecen las variables y parámetros específicos para el cálculo de las curvas de iniciación. En la parte inferior izquierda, en el cuadro denominado **Gráfica de iniciación** se representan todas las curvas de iniciación del material. Por último, en la parte inferior derecha, en el cuadro denominado **Datos de iniciación** se obtienen los datos numéricos del cálculo de las curvas de iniciación en una tabla de datos.
A continuación se describen con mayor detalle cada uno de éstos cuadros.
En la figura 3.14 se puede ver con mayor detalle el primer cuadro.
Dentro de este cuadro, el usuario deberá especificar diferentes parámetros para el cálculo de las

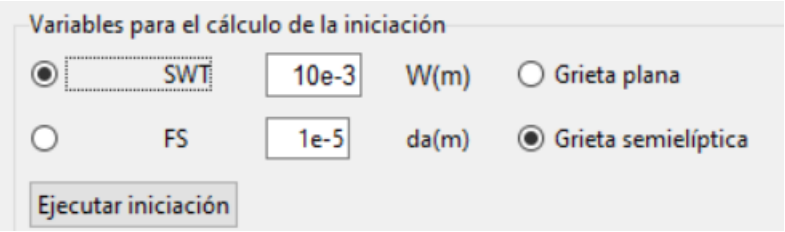

Figura 3.14 Captura del cuadro de Variables para el cálculo de la iniciación.

curvas de iniciación. Estos parámetros son: el criterio de iniciación, la anchura del espécimen y el tipo de propagación.
Primero, el usuario debe elegir entre los criterios **SWT** o **FS** correspondientes a los criterios de Smith-Watson-Topper 2.1.2 y Fatemi-Socie 2.1.1 respectivamente.

Segundo, debe especificar la anchura (**W**) en metros y el paso de grieta (**da**) en metros con el que se van a llevar a cabo el cálculo de las curvas de iniciación.

Por último, se debe definir el tipo de grieta 2.2, la cual puede ser **Grieta plana** o **Grieta semi-elíptica**.

Una vez definidos todos los parámetros, y comprobado que se quiere realizar las curvas dentro de la carpeta correcta, el usuario puede iniciar el cálculo de las curvas de iniciación presionando el botón **Ejecutar iniciación**. Si el cálculo no ha sido realizado anteriormente para los parámetros definidos, se obtendrán las curvas y los datos numéricos en los cuadros inferiores como se observa en las figuras 3.15 y 3.16.

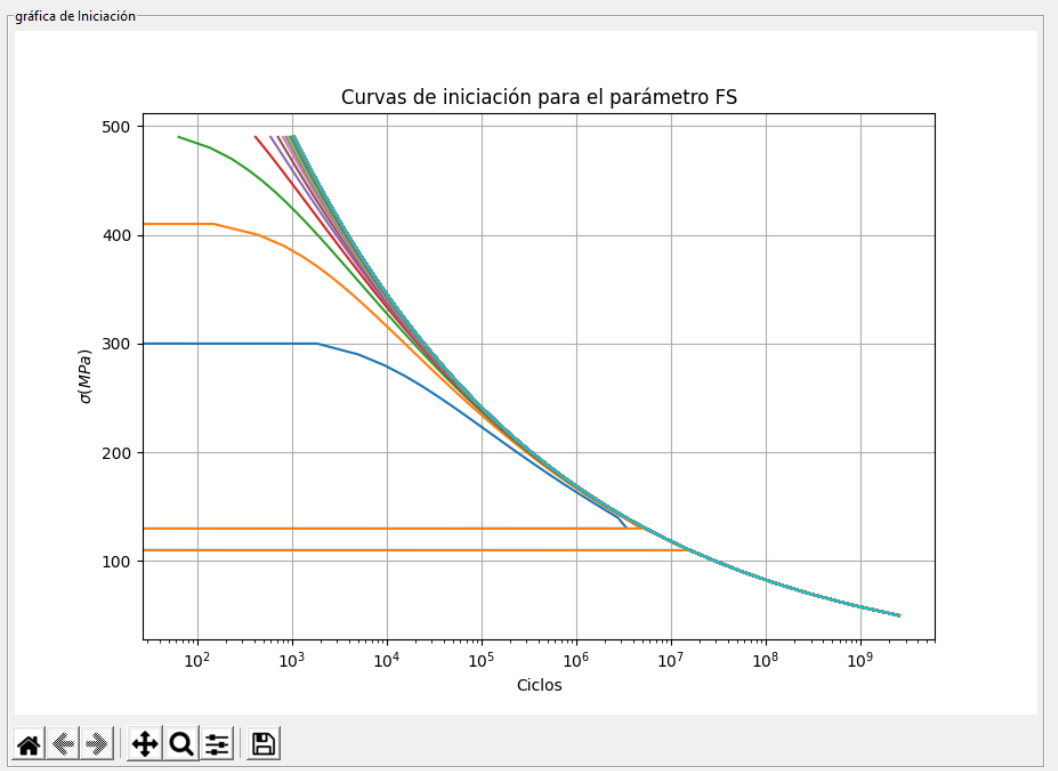

Figura 3.15 Ejemplo de curvas de iniciación.

Datos de Iniciación

0.000e+00	5.000e-05	1.149e-04	1.869e-04	2.6
0.0009512	2560000000.0	2560000000.0	2560000000.0	2560000000.0
0.001145	791400000.0	791400000.0	791400000.0	791400000.0
0.00134	293300000.0	293300000.0	293300000.0	293300000.0
0.001536	124100000.0	124100000.0	124100000.0	124100000.0
0.001733	58150000.0	58150000.0	58150000.0	58150000.0
0.001931	29510000.0	29510000.0	29510000.0	29510000.0
0.002131	15970000.0	15970000.0	15970000.0	15970000.0
0.002332	0.0	0.0	8946000.0	9027000.0
0.002534	3450000.0	4891000.0	5330000.0	5379000.0
0.002737	2756000.0	3171000.0	3295000.0	3328000.0
0.002941	1786000.0	2032000.0	2103000.0	2127000.0
0.003146	1164000.0	1332000.0	1380000.0	1398000.0
0.003353	769100.0	892700.0	928100.0	942100.0
0.00356	515400.0	610400.0	637500.0	648700.0
0.003769	349500.0	424900.0	446200.0	455300.0
0.003979	239300.0	300400.0	317600.0	325100.0
0.00419	164900.0	215300.0	229400.0	235800.0
0.004403	114100.0	156200.0	168000.0	173300.0
0.004616	78870.0	114500.0	124500.0	129000.0
0.004831	54250.0	84690.0	93200.0	97140.0
0.005047	36910.0	63140.0	70460.0	73880.0
0.005264	24600.0	47380.0	53730.0	56720.0
0.005482	15820.0	35740.0	41290.0	43910.0
0.005701	9544.0	27060.0	31940.0	34250.0
0.005922	5055.0	20550.0	24860.0	26910.0
0.006143	1853.0	15620.0	19450.0	21270.0
0.006366	0.0	11860.0	15280.0	16910.0
0.00659	0.0	8991.0	12050.0	13520.0
0.006815	0.0	6781.0	9534.0	10860.0
0.007042	0.0	5075.0	7559.0	8754.0

Figura 3.16 Ejemplo de tabla de datos numéricos de la iniciación.

El cálculo de las curvas de iniciación puede tardar del orden de minutos dependiendo del procesador del ordenador. Puesto que se requiere algo de tiempo, no es eficiente tener que calcular las curvas de iniciación cada vez que queremos hacer un cálculo posterior con los mismos parámetros. Por ello, en caso de que se hayan realizado los cálculos con los mismo parámetros se avisa al usuario que las curvas han sido realizadas con anterioridad. Este mensaje se muestra en la figura 3.17. En este caso sólo se cargarán los datos numéricos en la tabla de datos. La gráfica con las curvas se encuentra guardada en la carpeta *grafs* pero no se muestra en el cuadro.

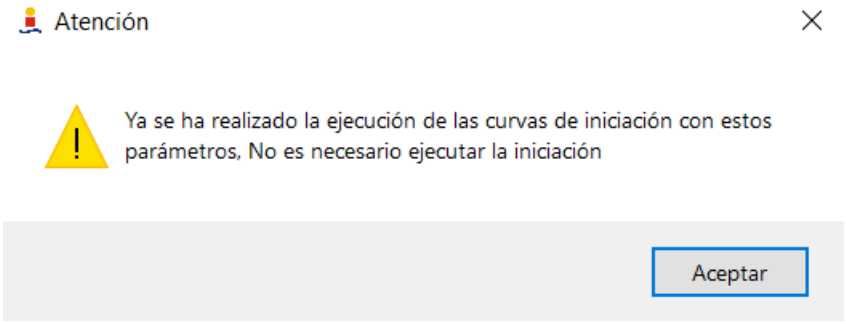

Figura 3.17 Mensaje de advertencia de que las curvas de iniciación han sido realizadas.

En caso de que el cálculo de las curvas de iniciación no se haya realizado con anterioridad y el usuario quiera ejecutarlo sin haber especificado las propiedades del material, esto producirá un mensaje de error como el que se muestra en la figura 3.18.

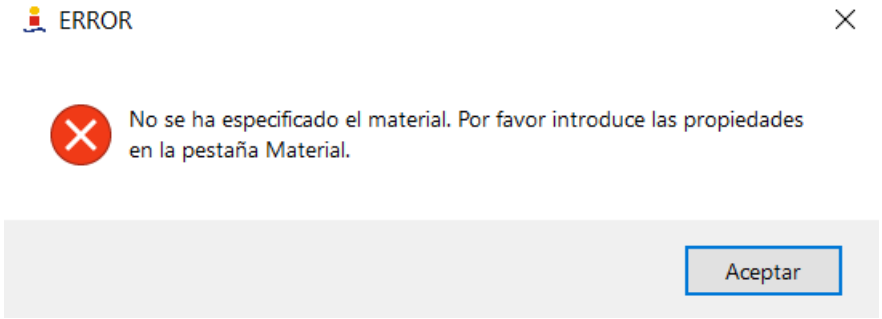

Figura 3.18 Mensaje de error al no especificar el material.

Otro posible error puede producirse si por alguna razón se genera el archivo que contiene los datos pero este no se completa. En este caso, el archivo existe pero el contenido del mismo esta dañado. El usuario será informado mediante un mensaje de error como el que se muestra en la figura 3.19.

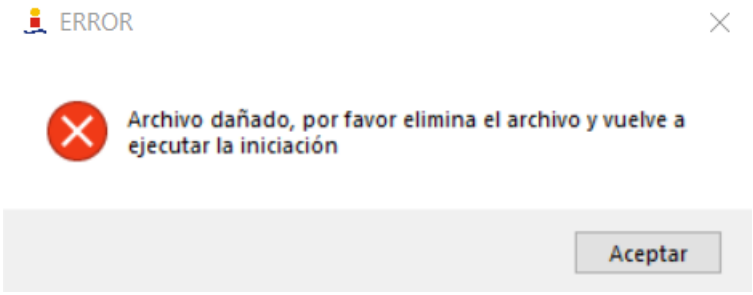

Figura 3.19 Mensaje de error de archivo dañado.

3.4 Pestaña de Datos

La siguiente pestaña que vamos a desarrollar es la pestaña de **Datos**. Esta pestaña se utiliza para cargar y visualizar los datos de tensiones y deformaciones que podrán usarse posteriormente para calcular la estimación de vida a fatiga que se realizará en la pestaña de **Cálculo** 3.5. En la figura 3.20 se muestra una captura de esta pestaña datos de tensiones y deformaciones cargados.

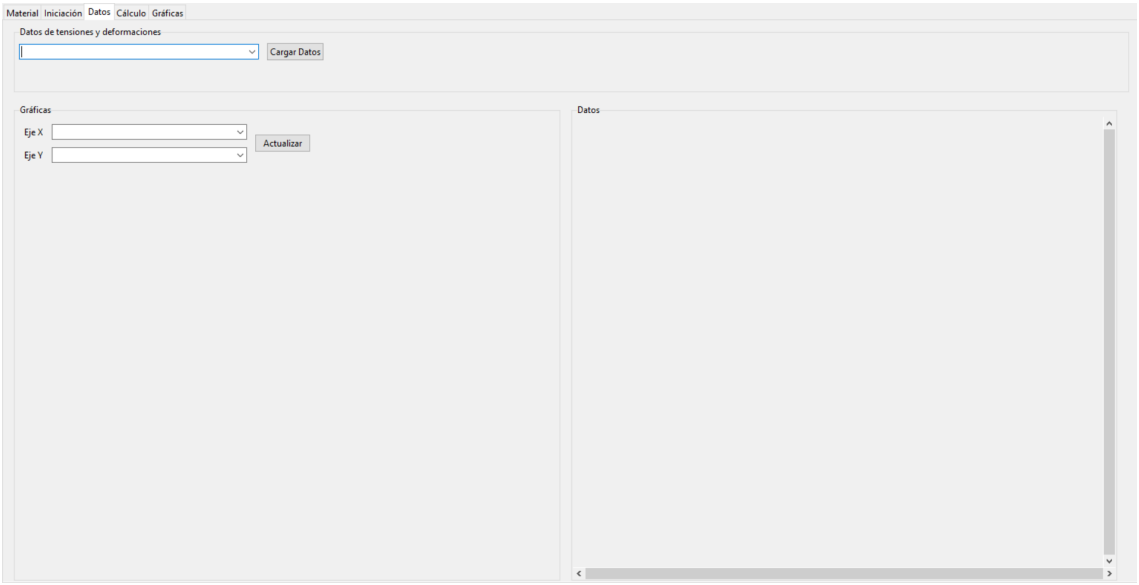

Figura 3.20 Captura de la pestaña Datos antes de cargar datos.

Esta pestaña esta formada por tres cuadros diferentes, los cuáles tienen funciones diferentes. El cuadro superior, llamado **Datos de tensiones y deformaciones** está compuesto por una lista y un botón. Esta lista, se encuentra vacía en principio. Para poder rellenar esta lista, el usuario debe pulsar el botón **Cargar Datos**. Esta acción también se puede realizar pulsando dentro del menú **Archivo** sobre la opción **Abrir datos experimentales**, abriendo una ventana de diálogo preguntando al usuario por la carpeta donde se encuentran los datos de tensiones y deformaciones. Esta ventana de diálogo es como la que se muestra en la figura 3.3.

Si la carpeta es válida, se realizará la carga de todos los datos en la lista, como se puede mostrar en la figura 3.21. Esta lista deberá contener los archivos de datos para el caso de compresión y para el caso de tracción. Los archivos que se encuentren dentro de la carpeta deberán llamarse comenzando por TENSOR_COMPR para el caso de compresión y por TENSOR_TRAC para el

caso de tracción. Además deberán estar en formato *.dat*.

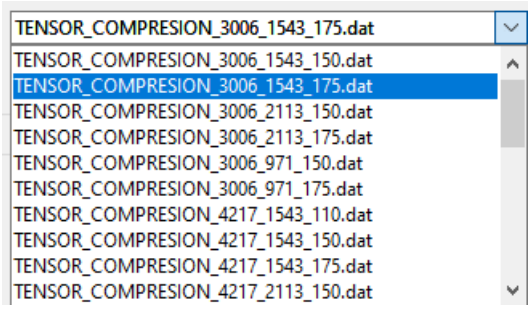

Figura 3.21 Lista de archivos de datos.

Una vez cargados los datos de tensiones y deformaciones de forma correcta, el cuadro inferior derecho, llamado **Datos** mostrará una tabla de datos numéricos del archivo seleccionado en la lista tal y como se puede ver en la figura 3.22.

Datos

Y	s_xx	s_yy	s_z
0.0	-432.021	-26.0427	-151.161
9.999999999996123e-06	-423.933	-31.6339	-150.337
1.9999999999992246e-05	-415.848	-37.2144	-149.511
3.0000000000002247e-05	-408.284	-40.8388	-148.211
3.999999999999837e-05	-400.739	-44.457	-146.915
4.999999999999449e-05	-394.992	-47.5091	-146.025
6.0000000000004494e-05	-389.256	-50.5542	-145.137
6.999999999998674e-05	-384.239	-53.1863	-144.35
7.999999999999674e-05	-379.233	-55.8115	-143.565
9.000000000000674e-05	-374.717	-58.1243	-142.838
9.999999999998899e-05	-370.211	-60.4302	-142.112
0.00010999999999999899	-366.059	-62.487	-141.42
0.00011999999999999511	-361.917	-64.5369	-140.73
0.00012999999999999123	-358.059	-66.3812	-140.065
0.00014000000000000123	-354.21	-68.2188	-139.401
0.00014999999999999736	-350.598	-69.8825	-138.758
0.00015999999999999348	-346.995	-71.5394	-138.116
0.00017000000000000348	-343.595	-73.0468	-137.492
0.0001799999999999996	-340.205	-74.5475	-136.868
0.00018999999999999573	-336.995	-75.9176	-136.261
0.00020000000000000573	-333.794	-77.2811	-135.655
0.00020999999999998797	-330.753	-78.5294	-135.063
0.00021999999999999797	-327.721	-79.7713	-134.472
0.0002299999999999941	-324.834	-80.9107	-133.896
0.00023999999999999022	-321.956	-82.0438	-133.32
0.0002500000000000002	-319.21	-83.0852	-132.757
0.00025999999999999635	-316.473	-84.1203	-132.196
0.00026999999999999247	-313.857	-85.073	-131.647

Figura 3.22 Tabla de datos numéricos de tensiones y deformaciones.

Por otra parte, en el cuadro inferior izquierdo, llamado **Gráficas**, aparecerá una gráfica como la que se muestra en la figura 3.23.

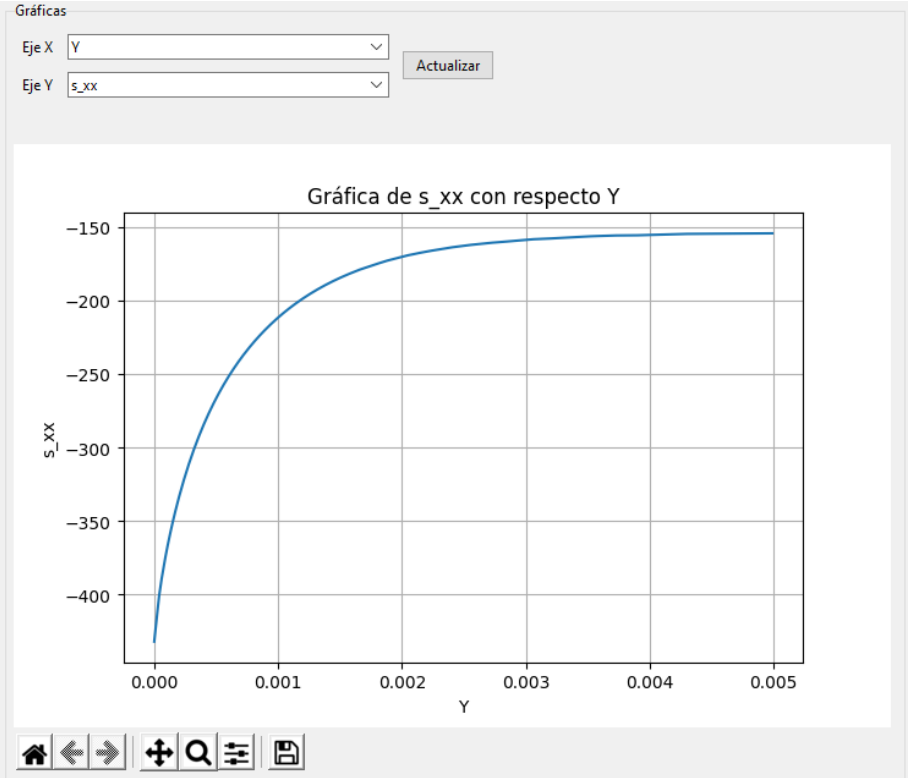

Figura 3.23 Captura del cuadro Gráficas.

Como se puede observar en la figura 3.23, en la parte superior de este cuadro existen dos listas con las que poder modificar los datos de los ejes y además un botón con el que actualizar la gráfica. Estas listas contienen las propiedades de los datos, como se puede ver en la figura 3.24. Estas listas coinciden con las columnas de la tabla de datos. Por defecto, al cargar los datos se seleccionan para **Eje X** la propiedad "Y", que es la profundidad de la grieta en metros y para **Eje Y** la propiedad "s_xx", que es la tensión principal del estado de fatiga. En resumen l

En la tabla 3.1 se hace una descripción de las propiedades que se encuentran en los datos de tensiones y deformaciones.

Cada vez que el usuario quiera cambiar de ejes para la gráfica, tendrá que seleccionar las propiedades en las listas y posteriormente pulsar en el botón **Actualizar**.

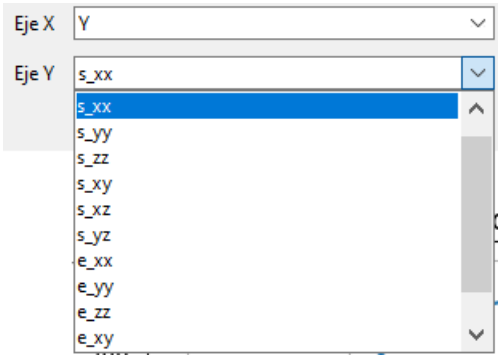

Figura 3.24 Lista de propiedades de los datos.

Tabla 3.1 Tabla descriptiva de las datos de tensiones y deformaciones.

Columna	Símbolo	Descripción
Y	y	Dimensión en la dirección de profundidad de la grieta (m)
s_xx	σ_{xx}	Tensión principal de fatiga (MPa)
s_yy	σ_{yy}	Tensión en la dirección de la profundidad de la grieta (MPa)
s_zz	σ_{zz}	Tensión en la dirección normal a σ_{xx} y σ_{yy} (MPa)
s_xy	σ_{xy}	Tensión tangencial en el plano xy (MPa)
s_xz	σ_{xz}	Tensión tangencial en el plano xz (MPa)
s_yz	σ_{yz}	Tensión tangencial en el plano yz (MPa)
e_xx	ε_{xx}	Deformación en la dirección de la tensión principal de fatiga
e_yy	ε_{yy}	Deformación en la dirección de profundidad de la grieta
e_zz	ε_{zz}	Deformación en la dirección normal al plano xy
e_xy	ε_{xy}	Deformación tangencial en el plano xy
e_xz	ε_{xz}	Deformación tangencial en el plano xz
e_yz	ε_{yz}	Deformación tangencial en la dirección yz

En caso de que el usuario haya seleccionado una carpeta no válida al cargar los datos de tensiones y deformaciones. El usuario será informado a través de un mensaje de error como el que se muestra en la figura 3.35.

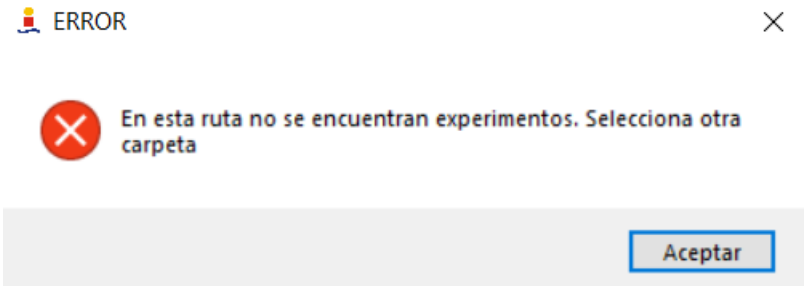

Figura 3.25 Mensaje de error de carga de datos.

3.5 Pestaña de Cálculo

Una vez introducidos todos los datos de tensiones y deformaciones en la pestaña **Datos** 3.4, estamos en disposición de calcular la estimación de vida a fatiga.
Esta pestaña está formada por cuatro cuadros que tendrán diferentes funciones. En la figura 3.26 se muestra una captura de esta pestaña.

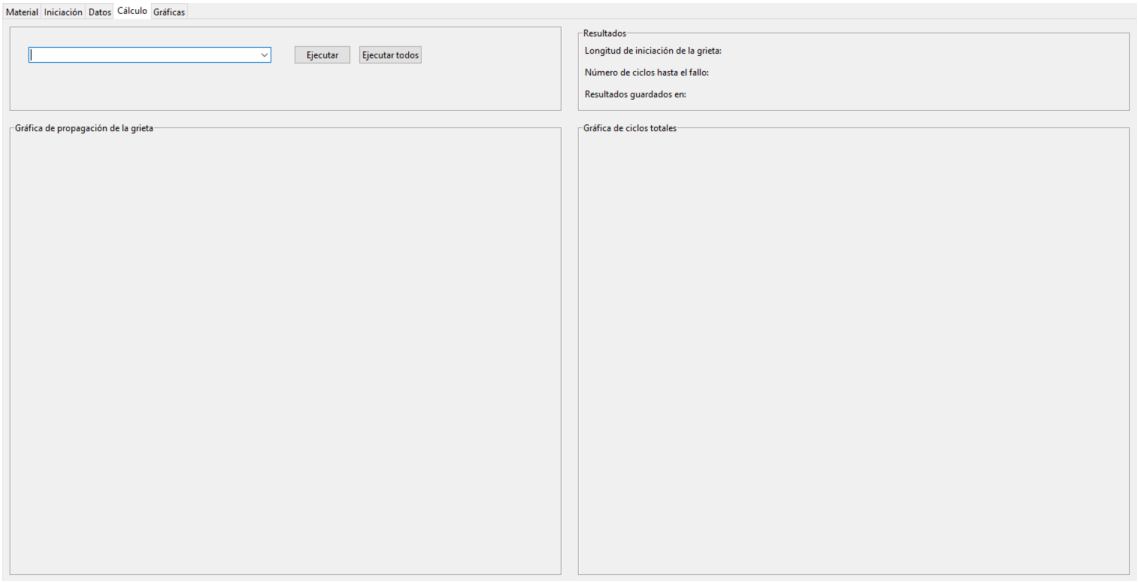

Figura 3.26 Captura de la pestaña Cálculo sin ejecutar.

El cuadro superior izquierdo está formado por una lista y dos botones. Esta lista contendrá el nombre de los datos de tensiones y deformaciones que añadimos anteriormente en la pestaña **Datos** si éstos han sido añadidos correctamente. La lista es rellenada automáticamente desde la pestaña **Datos** o desde el menú **Archivo**, pulsando en la opción **Abrir datos experimentales**. En caso contrario, esta lista estará vacía. En la figura 3.27 se muestra una captura de la lista con los nombres de los datos.

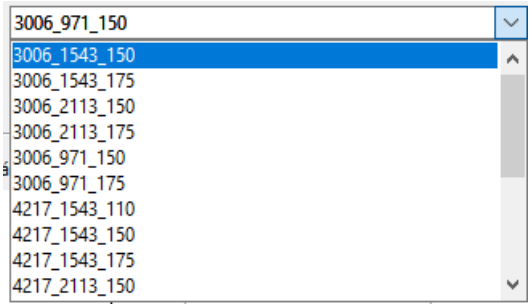

Figura 3.27 Lista de nombre de los datos.

Para realizar el cálculo de la estimación de vida a fatiga para unos datos específicos de la lista, el usuario debe presionar el botón **Ejecutar** situado en el mismo cuadro. El usuario debe tener en cuenta que el criterio de iniciación seleccionado en la pestaña **Iniciación** 3.3 y las propiedades del material seleccionadas en la pestaña **Material** 3.2 serán los utilizados para el cálculo. Los resultados y los parámetros de este cálculo serán añadidos al archivo llamado *resultados.xlsx* dentro de la carpeta *resultados_generales*, que tendrá una apariencia como la que se muestra en la figura 3.28.

El cálculo puede llevar varios minutos. Una vez realizado el cálculo, se producirán modificaciones en los demás cuadros.

	A	B	C	D	E	F	G	H
1	exp_id	param	N_t_min	N_i_min	N_p_min	N_i_perc	N_p_perc	a_inic(mm)
2	3006_1543_150	SWT	29882.1551	1402.001527	28480.15353	4.69176846358512%	95.30823153641	0.06
3	3006_1543_175	SWT	19428.1368	1207.035856	18221.10096	6.2128235317774%	93.78717646822	0.07
4	3006_971_150	SWT	47695.073	1127.954988	46567.118	2.364929787196029%	97.63507021280	0.05
5								
6								
7								
8								
9								

Figura 3.28 Excel de resultados.

Primero, el cuadro inferior izquierdo, denominado **Gráfica de propagación de la grieta** contendrá una figura que represente la propagación de la grieta en función del número de ciclos. Esta gráfica es la resultante de la suma de la contribución de los ciclos de iniciación y propagación a propagación total de la grieta. En la figura 3.29 se muestra una captura de este cuadro después del cálculo, donde se observa que hasta un cierto número de ciclos el crecimiento de la grieta estable, en este punto la grieta crece de forma más rápida hasta que se produce el fallo del material.

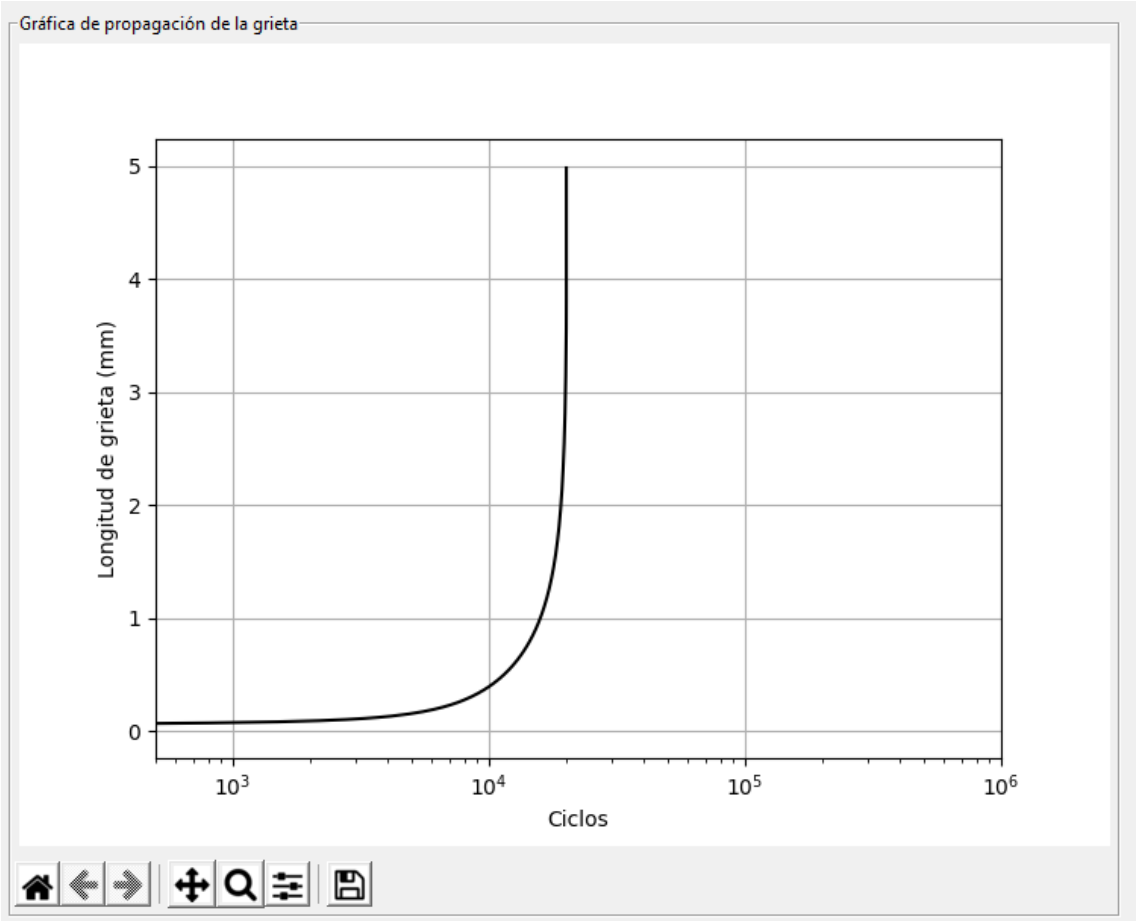

Figura 3.29 Cuadro de Gráfica de propagación de la grieta para unos datos específicos.

Segundo, el cuadro superior derecho, llamado **Resultados** contendrá los resultados del cálculo de forma numérica e indicará al usuario donde se guardaron los mismos. En la figura 3.30 se puede ver una captura de este cuadro tras la realización de los cálculos. Los resultados de la longitud de iniciación y el número de ciclos totales son añadidos al excel que se encuentra en la carpeta *resultados_generales* y que se denomina *resultados.xls*.

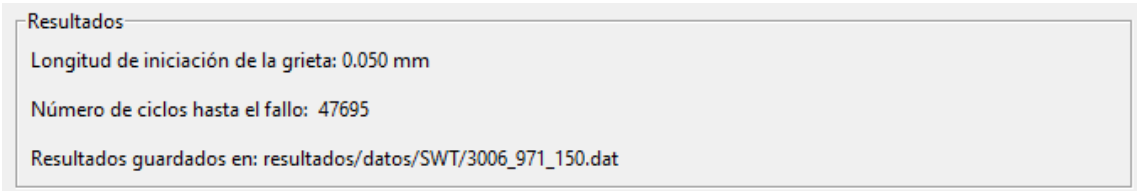

Figura 3.30 Cuadro de Resultados tras el cálculo de unos datos específicos.

Por último, el cuadro inferior derecho, denominado **Gráfica de ciclos totales**, contendrá los resultados gráficos de la estimación de vida a fatiga para unos determinados datos seleccionados en la lista. En la figura se muestra una captura de este cuadro tras la realización de los cálculos 3.31.

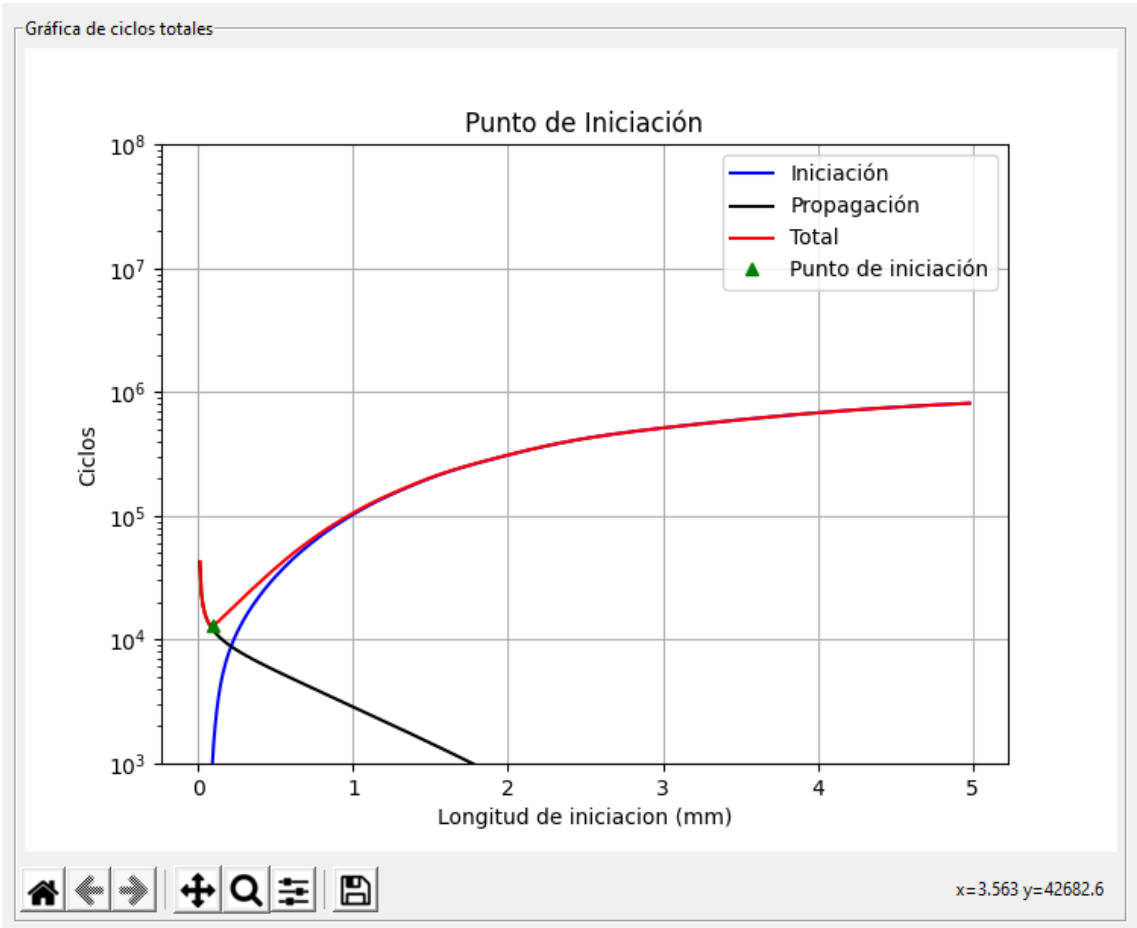

Figura 3.31 Cuadro de Gráfica de ciclos totales tras el cálculo de unos datos específicos.

Como ya hemos visto, el cuadro superior izquierdo tiene otro botón llamado **Ejecutar todos**. Este botón ejecuta el cálculo para todos los elementos de la lista de una vez para un criterio de

iniciación y un material específicos. Sin embargo, esta acción puede llevar horas en completarse si el número de elementos de la lista es elevado. Por ello, el usuario es advertido mediante un mensaje emergente advirtiendo del tiempo de ejecución que podría llevar la acción. El mensaje que aparecerá en pantalla, como el que se muestra en la figura 3.32, preguntará al usuario si está seguro de ejecutar los cálculos para todos los elementos.

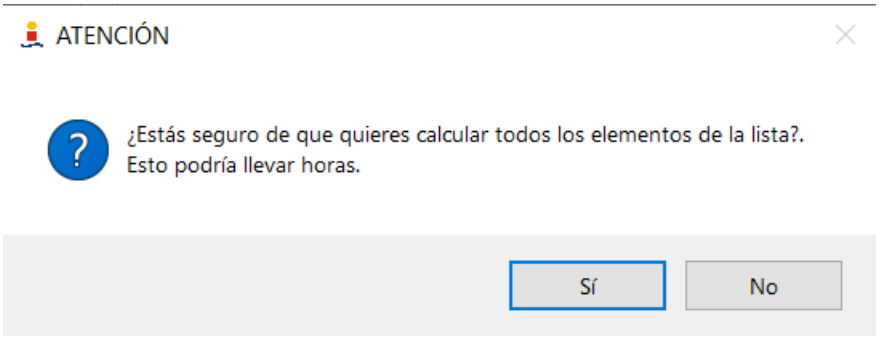

Figura 3.32 Mensaje de advertencia al ejecutar todos los cálculos.

Los resultados de ejecutar todos los cálculos son guardados automáticamente en las carpetas donde se encuentra el directorio principal. Para el caso de la gráfica de ciclos totales, las gráficas serán guardadas dentro de la carpeta *\resultados \grafs \CRITERIO* la carpeta y para el caso de los resultados numéricos, éstos serán guardados en la carpeta *\resultados \datos \CRITERIO* donde CRITERIO puede ser SWT para el criterio de Smith-Watson-Topper o FS para el criterio de Fatemi-Socie.

Tras pulsar este botón los resultados se pueden ir visualizando en las diferentes gráficas de la pestaña. En el caso de la **Gráfica de ciclos totales**, los resultados se van sustituyendo por los nuevos, de forma que sólo son visibles mientras se está el realizando un cálculo de un elemento de la lista. En cambio, en la **Gráfica de propagación de la grieta** los resultados se irán imprimiendo y podrán visualizarse todos a la vez. Un ejemplo de esta gráfica se muestra en figura 3.33.

Si el usuario ha introducido todos los datos y parámetros necesarios no se deberían producir errores en el cálculo de la estimación de vida a fatiga. En cambio se producirán errores si se dan diferentes circunstancias.

Por una parte, si el usuario ha olvidado especificar el material, éste será informado mediante un mensaje de error como el que se muestra en la figura 3.18 y el que cálculo no se podrá realizar hasta que no se especifique el material.

Por otra parte, si el usuario no ha cargado los resultados de las curvas de iniciación de la pestaña **Iniciación** o bien no ha comprobado que las curvas ya está realizadas con anterioridad, éste será informado con un mensaje de error como el que se muestra en la figura 3.34.

El último error que se puede producir, es que el usuario pulse el botón de **Ejecutar** sin haber cargado los datos de tensiones y deformaciones de correctamente. El usuario será advertido del error mediante un mensaje emergente como el que se muestra en la figura 3.35.

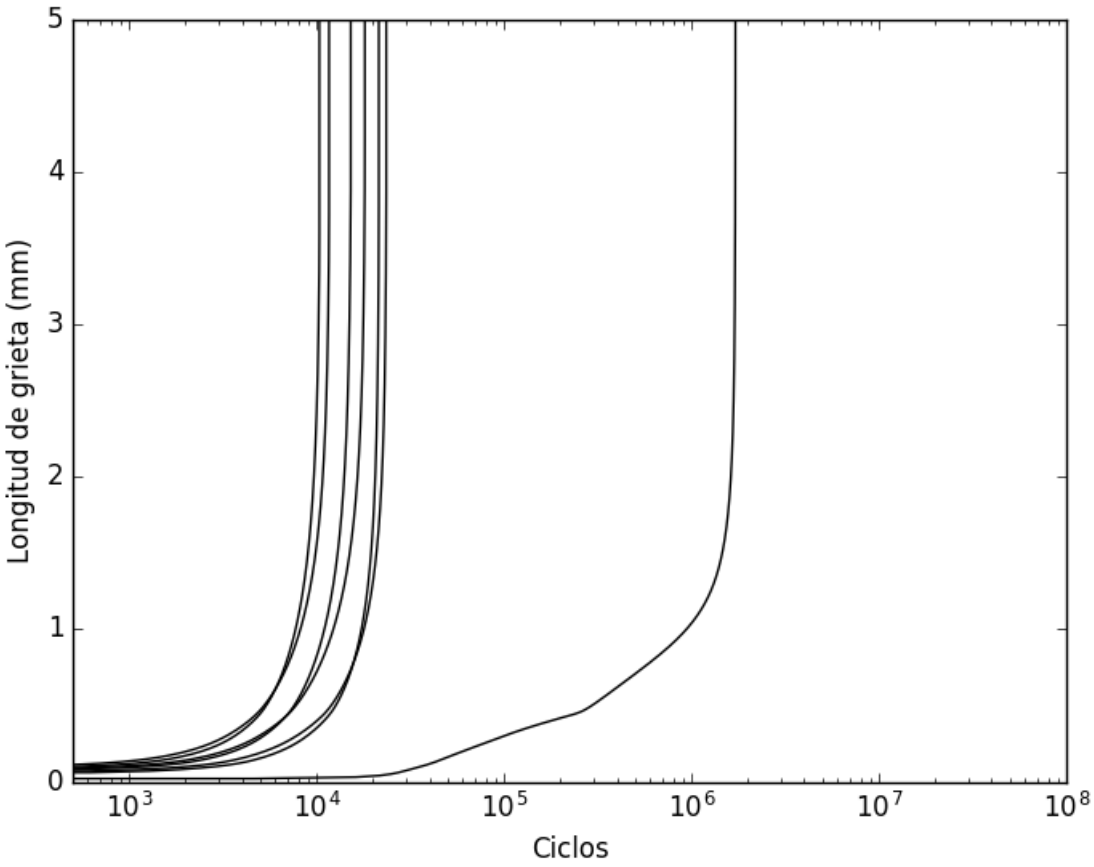

Figura 3.33 Ejemplo de curvas a-N_t con todos los datos de la lista[14].

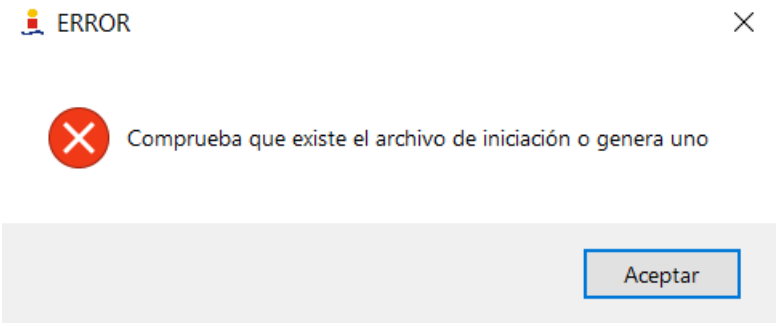

Figura 3.34 Mensaje de error de que no se ha especificado el archivo de iniciación.

3.6 Pestaña de Gráficas

En está última pestaña, se podrán visualizar diferentes gráficas interesantes tras obtener los resultados del cálculo de estimación de vida a fatiga. Esta pestaña sólo tiene dependencia con la pestaña **Iniciación**, ya que el usuario debe seleccionar el criterio de fatiga multiaxial (FS o SWT). A primera vista, esta pestaña tiene una apariencia como la que se muestra en la figura 3.36.

Como se observa en la figura 3.36 esta pestaña está formada por cuatro cuadros diferentes que muestran diferente información.

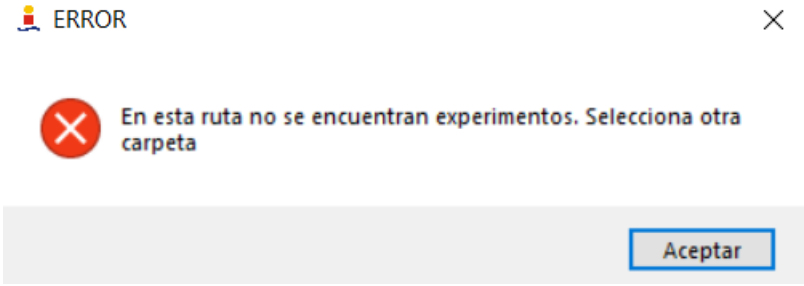

Figura 3.35 Mensaje de error en la carga de datos.

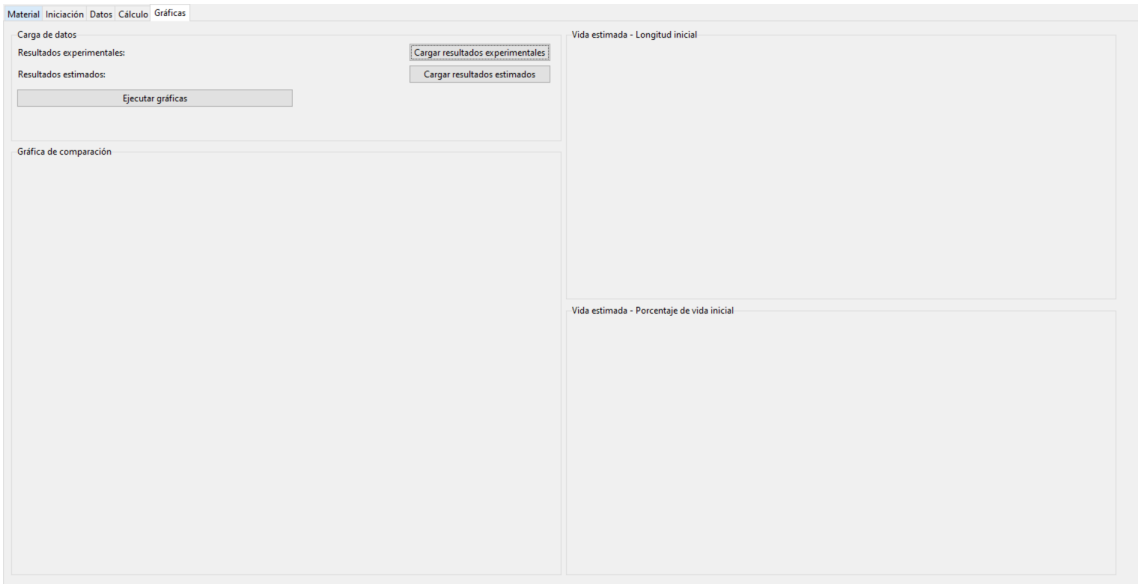

Figura 3.36 Captura de la pestaña Gráficas antes de realizar ninguna acción.

Por una parte, tenemos el cuadro superior izquierdo, denominado **Carga de datos**. Este cuadro se utiliza para pedir al usuario los datos experimentales y los estimados tras el cálculo. Los resultados experimentales se cargan pulsando en el botón **Cargar resultados experimentales** y los resultados estimados se cargan pulsando en el botón **Cargar resultados estimados**. El usuario tendrá que seleccionar los archivos en las ventanas de diálogo para los resultados experimentales y para los resultados estimados. Los archivos válidos para los resultados experimentales están en formato *.dat*, mientras que los resultados estimados pueden estar en los formatos *.dat* y *.xlsx*. En las figuras 3.37 y 3.38se muestran una captura de ambas ventanas de diálogo.

El usuario puede comprobar que se han cargado los archivos seleccionados en el presente cuadro, ya que se indica las rutas donde se encuentran los correspondientes archivos seleccionados, como se muestra en figura 3.39.

Si los archivos que ha seleccionado el usuario son correctos, el usuario puede mostrar las gráficas de los demás cuadros pulsando en el botón **Ejecutar gráficas**. Tras pulsar este botón, los cuadros restantes se modificarán.

El cuadro de mayor de tamaño, situado en la parte inferior izquierda y llamado **Gráfica de comparación**, contendrá una gráfica comparando los resultados experimentales con los resultados estimados. En el eje X se representa la vida estimada mientras que en el eje Y se representa la vida experimental. La recta roja representan los puntos de equivalencia entre la vida experimental y la

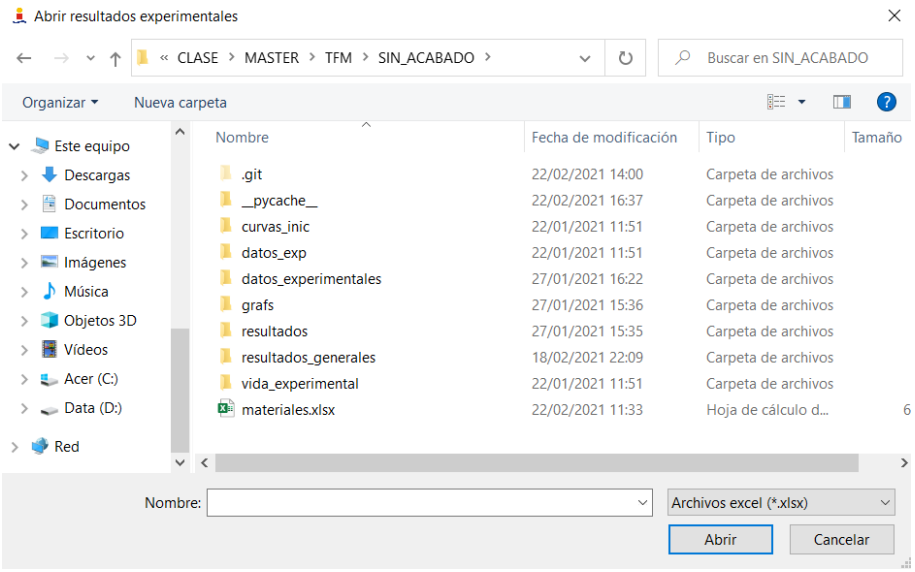

Figura 3.37 Captura de la ventana para abrir resultados experimentales.

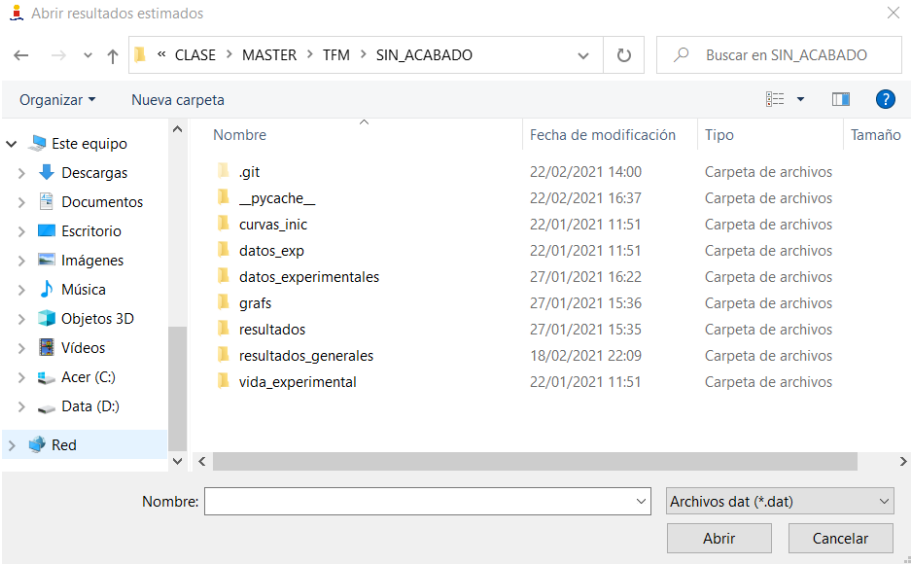

Figura 3.38 Captura de la ventana para abrir resultados estimados.

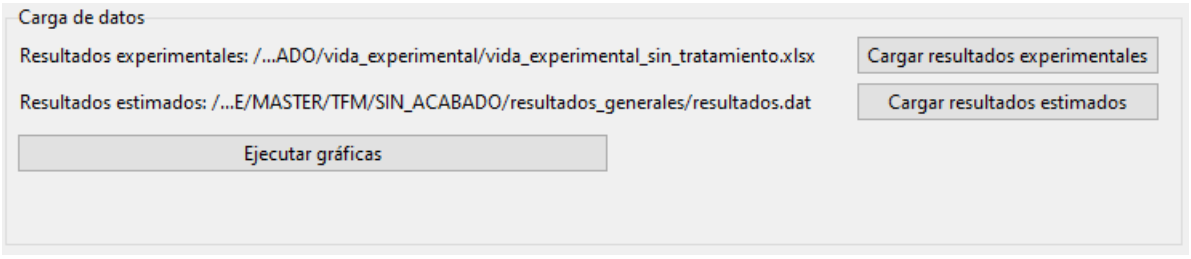

Figura 3.39 Cuadro de carga de datos tras seleccionar los archivos.

vida estimada, la recta verde representa la recta de regresión de los datos, y las rectas amarillas representa el doble y la mitad de la vida. Estas rectas nos permiten conocer si el modelo teórico de estimación de vida a fatiga es válido. En la figura 3.40, se muestra una captura de este cuadro para unos archivos seleccionados.

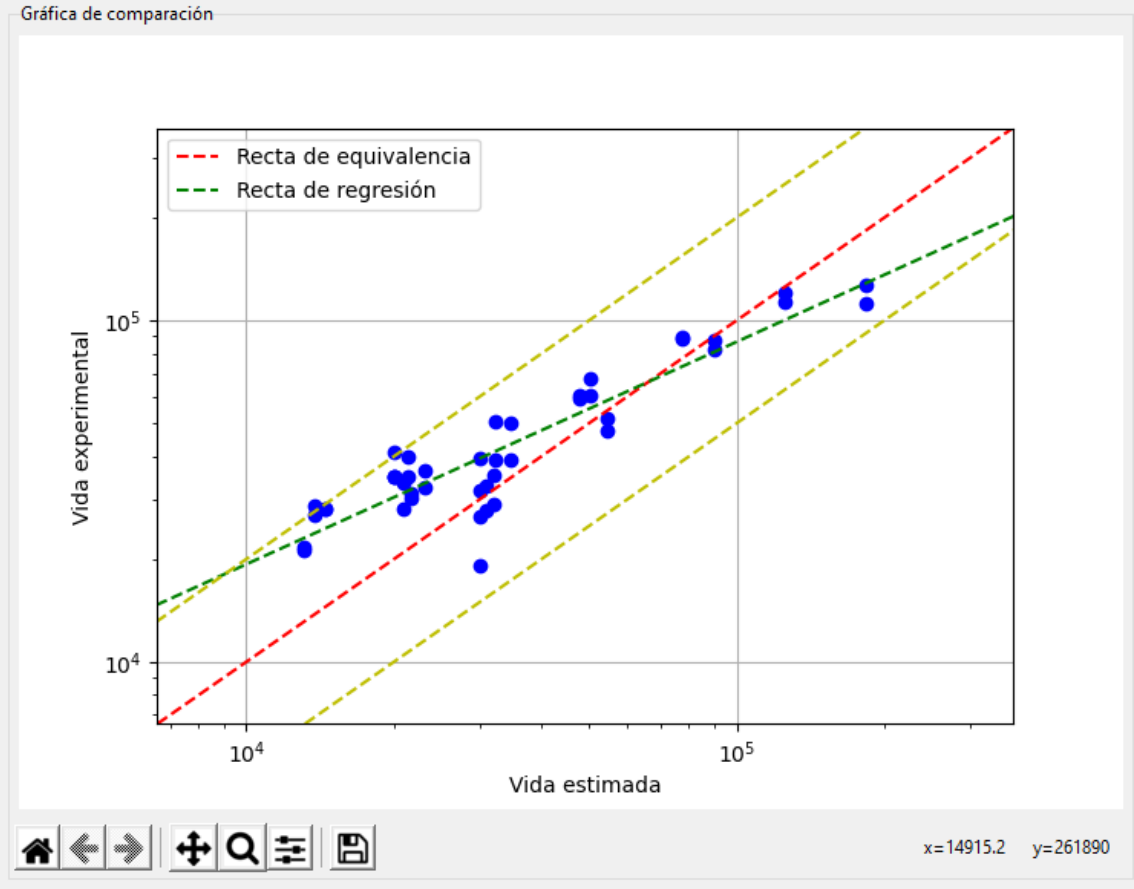

Figura 3.40 Cuadro de la Gráfica de comparación.

El cuadro situado en la parte superior derecha se denomina **Vida estimada- Longitud inicial**. En este cuadro se representa la gráfica que representa la longitud de grieta de iniciación con respecto a los ciclos. Este cuadro hace uso exclusivamente de los resultados estimados. En la figura 3.41 se muestra un ejemplo de esta gráfica con los resultados estimados seleccionados.

El cuadro situado en la parte inferior derecha se denomina **Vida estimada-Porcentaje de vida inicial**. En este cuadro se presenta la gráfica que relaciona el porcentaje que representa la fase de iniciación en la vida de la grieta con respecto a la vida estimada en ciclos. Al igual del cuadro anterior, este cuadro únicamente hace uso del archivo de resultados estimados. En la figura 3.41 se muestra un ejemplo de esta gráfica con los resultados estimados seleccionados.

En el caso de que los archivos no sean correctos, se producirán errores. Estos errores pueden deberse a dos razones fundamentalmente.

La primera de ellas es que el archivo no sea el correcto, ya sea por confusión o porque la extensión no sea la correcta.
La segunda razón es que los archivos que contienen los resultados experimentales no contengan la misma cantidad de elementos, por lo que no se pueden comparar.
La tercera razón, es que el formato de los datos almacenados no sea el correcto. El archivo de resultados estimados debe estar en el mismo formato que el se muestra en la figura 3.28.

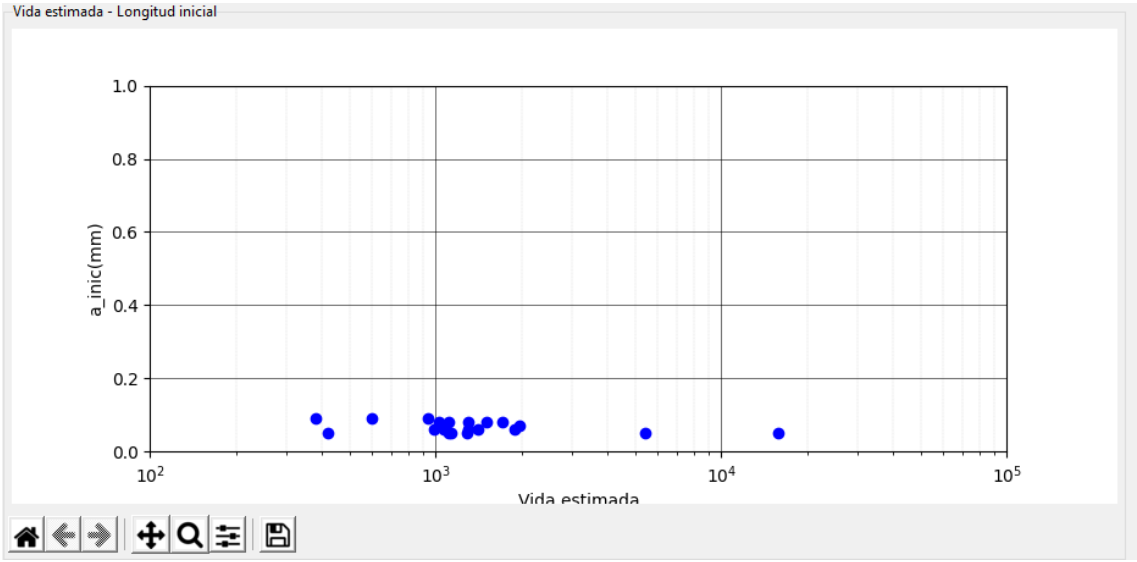

Figura 3.41 Cuadro de gráfica de longitud de iniciación y vida estimada.

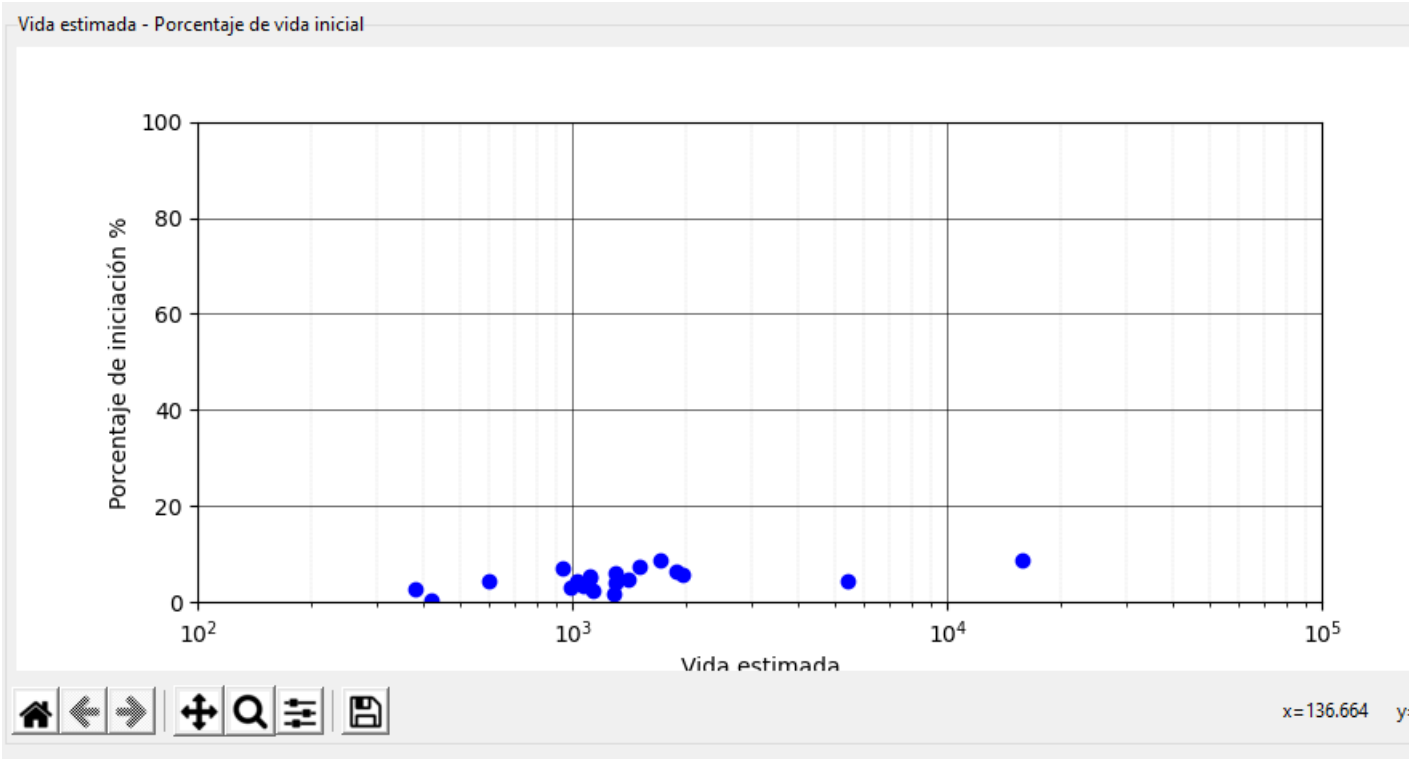

Figura 3.42 Cuadro de gráfica de porcentaje de vida de iniciación y vida estimada.

Estos errores se mostrarán en un mensaje emergente al usuario, como el que se muestra en la figura.

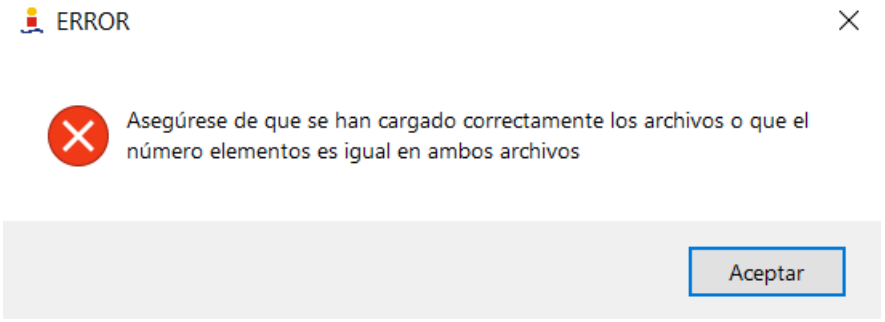

Figura 3.43 Mensaje de error en la carga de archivos experimentales y estimados.

4 Conclusiones

Se puede afirmar que el desarrollo de este software de interfaz gráfica de usuario se ha cumplido de forma satisfactoria. Teniendo en cuenta que el conocimiento sobre el desarrollo de software GUI en lenguaje Python era limitado al inicio del trabajo, se ha conseguido desarrollar un programa que ha cumplido con creces las expectativas que se tenían en el comienzo. Por tanto el trabajo ha supuesto un doble desafío personal. Por una parte, la necesidad de aprender y conocer los fundamentos teóricos de la fatiga y en concreto, la producida por esfuerzos multiaxiales. Por otra parte, la formación continua en el desarrollo de software de interfaz gráfica.

Como cualquier otro diseño de cualquier programa de ordenador, existen multitud de variantes e ideas con las que se pueden trabajar. Por tanto, cada desarrollador podría diseñar este mismo programa con una disposición y funciones diferentes pero cuyas funcionalidades fueran las mismas. Este programa, ha sido diseñado dentro de unas capacidades y conocimientos que yo poseía en el momento de su desarrollo. De esta forma, como trabajo futuro, podría modificarse, mejorarse e implementar nuevas funcionalidades que mejoren la experiencia de usuario. Este trabajo, podría ser un punto de inicio para un futuro trabajo de fin de grado o máster en el que se mejore, se traten los errores ocultos y se haga más profesional el software de forma que permita, incluso, su comercialización.

Desde el punto de vista personal, como he mencionado en el primer párrafo, la realización de este trabajo de fin de máster, ha supuesto para mí, el poder desarrollar mis habilidades en la programación en lenguaje Python, adquiridas de forma independiente al Máster de Ingeniería Aeronáutica. Además de lo anterior, también ha supuesto el aprendizaje de conceptos que no se ven con demasiada profundidad dentro de esta rama de la ingeniería como es la fatiga, presente en multitud de accidentes e incidentes aéreos.

Por último, quería destacar la necesidad de introducir en los estudios de ingeniería actuales asignaturas que fomenten y enseñen lenguajes de programación sea cual sea, no sólo en aquellos estudios relacionados con la informática. La necesidad de automatización de procesos y cálculos es bastante mayor hoy en día de lo que lo era hace unos años. Además el aprendizaje de un lenguaje de programación ayuda al alumno a comprender mejor los conceptos teóricos, ya que tiene que buscar la forma de crear un algoritmo que resuelva un problema en cuestión.

Apéndice A
Códigos utilizados para el desarrollo del programa

En este anexo se adjuntan todos los códigos escritos en lenguaje Python que han sido utilizados para el desarrollo del programa. En la figura A.1 se muestra un esquema en el que se ven las dependencias de unos códigos con otros y en qué pestañas se utilizan.

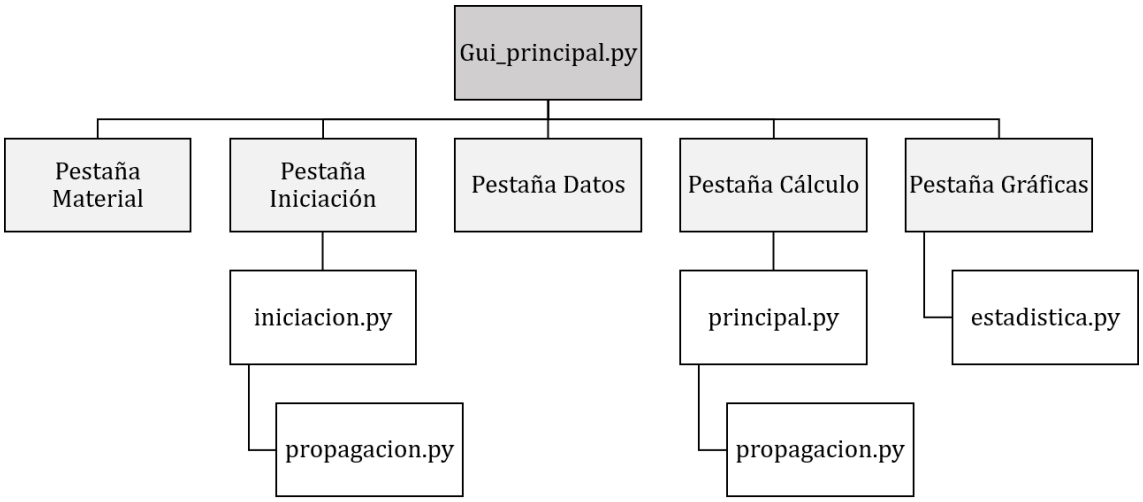

Figura A.1 Esquema de códigos del programa.

Como se observa el figura A.1, las pestañas Material y Datos no dependen de ningún código ya su función es únicamente la de suministrar los datos necesarios.

La pestaña Iniciación requiere hacer uso de las funciones del archivo *iniciacion.py* para obtener las curvas de iniciación, este archivo requiere de la funciones presentes en el archivo *propagacion.py* las cuales se utilizan para calcular la vida total hasta una cierta longitud de grieta.

La pestaña Cálculo requiere hacer uso de las funciones que se encuentran dentro del archivo *principal.py*, las cuales realizan el cálculo de la estimación de vida a fatiga teniendo en cuenta los datos de las curvas de iniciación generados previamente. Este archivo no requiere del archivo *iniciacion.py*, pero sí de las funciones dentro del archivo *propagacion.py*.

Por último, la pestaña Gráficas hace uso exclusivo de las funciones del archivo *estadistica.py* para realizar las diferentes figuras que se encuentran en esta pestaña.

A.1 Código de implementación de la aplicación GUI

Código A.1 Código de implementación del software de interfaz gráfica.

```
# -*- coding: utf-8 -*-
"""
Created on :19/02/2021

@author: David García Serrano
"""
###Importacion de módulos
import os
import tkinter as tk
from tkinter import ttk
import numpy as np
from iniciacion import curvas_iniciacion
from principal import *
from estadistica import*
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
    NavigationToolbar2Tk
import pandas as pd
import re

class programa(tk.Tk):
    def __init__(self):
        super().__init__()
        ###Configuración de inicio
        self.title("Cálculo de Fatiga")
        self.state("zoomed")
        self.geometry("800x600+10+10")
        self.iconbitmap("icon.ico")
        self.protocol("WM_DELETE_WINDOW", self.preguntar_salir)

        ### variables

        self.props = ["C","n","f","l_0","K_th","sigma_fl","a_0","K_IC","
            sigma_y","sigma_f","E","nu","b","G"]
        self.units = ["","","","m","MPa m^0.5","MPa","m","MPa m^0.5","
            MPa","MPa","MPa","","","MPa"]
        self.dict_prop = {}#Diccionario con los valores del material
        self.mat_values ={}
        self.material ="" #Material elegido en el combobox de material
        self.dict_units = dict(zip(self.props,self.units))

        self.N_i=[]    #vector con N_i
        self.n_a=1     #número de longitudes de grieta
        self.v_sigma=[] #vector de tensiones
        self.par =""   #parámetro
```

```
43          self.W =0.0    #anchura
44          self.da =0.0   #paso de grieta
45          self.ini_file ="" #Ubicación de archivo de iniciación
46          self.df =pd.DataFrame() #Dataframe de iniciación
47          self.df_materiales =pd.DataFrame() #Dataframe de materiales
48
49          self.dir_exp ="" #Directorio de datos de tensiones y
                deformaciones
50          self.files_exp =[]#Lista de datos de tensiones y deformaciones
51          self.file_path ="" #ubicación de los datos
52
53          self.lista_exp =[] #Lista de experimentos sin especificar
                traccion o compresión
54          self.df_datos =pd.DataFrame()
55
56          self.ubi_dat_exp ="" #ubicación de los datos experimentales para
                 pestaña gráficas
57          self.ubi_dat_est ="" #ubicación de los datos estimados para
                pestaña gráficas
58          self.ruta_principal = os.getcwd() #ruta de ejecución principal
59
60
61          for prop in self.props:
62              self.mat_values[prop] = tk.StringVar()
63
64          ### Menu
65          self.menu = tk.Menu(self)
66          self.file_menu = tk.Menu(self.menu, tearoff=0)
67          self.file_menu.add_command(label="Nuevo",command= self.
                abrir_nuevo)
68          self.file_menu.add_command(label="Seleccionar nueva carpeta",
                command= self.seleccionar__nueva_carpeta)
69          self.file_menu.add_command(label="Seleccionar carpeta existente",
                command= self.seleccionar_carpeta)
70          self.file_menu.add_separator()
71          self.file_menu.add_command(label="Abrir datos experimentales",
                command=self.carga_datos)
72          self.file_menu.add_command(label="Abrir resultados de iniciación
                ",command=self.abrir_resultados_iniciacion)
73          self.file_menu.add_separator()
74          self.file_menu.add_command(label="Salir", command=self.
                preguntar_salir)
75          self.menu.add_cascade(label="Archivo", menu=self.file_menu)
76          self.menu.add_command(label="Acerca de",command =self.
                mostrar_info)
77
78          self.config(menu=self.menu)
79
80          #Pestañas
```

```
81          self.pestanas = ["Material","Iniciación","Datos","Cálculo","Grá
                ficas"]
82          self.tabControl = ttk.Notebook(self)
83          self.tabs ={}
84          for pestana in self.pestanas:
85              self.tabs[pestana] = ttk.Frame(self.tabControl)
86
87              self.tabControl.add(self.tabs[pestana], text =pestana)
88
89          self.tabControl.pack(expand=1,fill= "both")
90
91          #Label Frame de propiedades del material
92          props_lf = ttk.Labelframe(self.tabs["Material"],text = "
                Propiedades")
93          props_lf.grid(column = 0,row =0,padx =20,pady=30)
94
95          # props_lf.place(width=180,height=500)
96          self.props_entries = {}
97          for prop in self.props:
98              self.props_entries[prop] = ttk.Entry(props_lf,textvariable =
                    self.mat_values[prop],width = 20,justify = "right",font
                    =("Arial",10))
99          props_labels = [ttk.Label(props_lf,text = f"{p}({self.dict_units
                [p]})",font =("Arial",10)) for p in self.props]
100         self.props_entries["C"].focus_set()
101         self.props_entries["G"].config(state=tk.DISABLED)
102         self.props_entries["a_0"].config(state=tk.DISABLED)
103
104         for i,prop in enumerate(self.props):
105             self.props_entries[prop].grid(column = 1, row = i, padx = 5,
                    pady = 5,sticky=tk.W)
106             props_labels[i].grid(column = 0, row = i, padx = 8,pady = 5,
                    sticky= tk.W)
107
108         #Botones de las propiedades
109         self.boton_borrar = ttk.Button(props_lf,width = 20,text="Borrar
                todo",command = self.borrar_campos)
110         self.boton_borrar.grid(column = 0,row =len(self.props),padx =5,
                pady =5,sticky= tk.W)
111         self.boton_guardar= ttk.Button(props_lf,text="Confirmar",width
                =20,command = lambda: self.guardar_campos(None))
112         self.boton_guardar.grid(column = 1,row =len(self.props),padx =5,
                 pady =5,sticky= tk.W)
113         self.tabs["Material"].bind("<Return>",self.guardar_campos)
114
115         #Guardar y borrar material
116         self.btn_guardar_material = ttk.Button(props_lf,width =20, text
                = "Guardar Material", command = self.guardar_material)
117         self.btn_borrar_material = ttk.Button(props_lf,width =20, text =
                 "Borrar Material",command = self.borrar_material)
```

```
118         self.btn_guardar_material.grid(column =1, row = len(self.props)
                +1, padx = 5, pady =5,sticky= tk.W)
119         self.btn_borrar_material.grid(column =0, row = len(self.props)+1,
                 padx = 5, pady =5,sticky= tk.W)
120
121         #combobox
122         self.comb_val = []
123         self.cargar_materiales()
124
125         self.combo_mat = ttk.Combobox(props_lf,width = 30,value =self.
                comb_val,font =("Arial",12,"bold"))
126         self.combo_mat.bind("<<ComboboxSelected>>",self.combosel)
127         self.combo_mat.grid(column = 0, row = len(self.props)+2,
                columnspan=2,padx = 5, pady = 8)
128
129         ### Label Frame del resumen
130         self.resum_lf =ttk.Labelframe(self.tabs["Material"],text = "
                Resumen")
131         self.resum_lf.grid(column = 1,row =0,padx =20,pady=30,sticky=tk.
                NW)
132
133         self.intro_resumen = ttk.Label(self.resum_lf, text = "Datos que
                se van a utilizar para la realización de los cálculos:",font
                = ("Arial",12))
134         self.intro_resumen.grid(column = 0, row =0,padx = 20)
135
136         self.resum_label ={}
137
138         for i,prop in enumerate(self.props):
139             self.resum_label[prop] = ttk.Label(self.resum_lf,text = prop+
                    ": ",font =("Arial",12,"bold"))
140             self.resum_label[prop].grid(column = 0,row =i+1, padx = 5,
                    pady = 5,sticky=tk.W)
141
142         ### Pestaña de Iniciación
143         self.vars_iniciacion_frame = ttk.Labelframe(self.tabs["Iniciació
                n"],text ="Variables para el cálculo de la iniciación",width
                =1200,height= 600)
144         self.vars_iniciacion_frame.place(relx= 0.01,rely =0.01,relwidth
                =0.975,relheight=0.25)
145
146         self.var_param = tk.StringVar()
147         self.var_param.set("SWT")
148
149         self.ac_param = tk.StringVar()
150         self.ac_param.set("eliptica")
151
152         self.CB_param_SWT =ttk.Radiobutton(self.vars_iniciacion_frame,
                text ="\tSWT",variable= self.var_param,value="SWT")
```

```
153         self.CB_param_SWT.grid(column = 0, row= 0 , columnspan= 2,pady =
                5, padx = 5,sticky = tk.W)
154         self.CB_param_FS =ttk.Radiobutton(self.vars_iniciacion_frame,
                text ="\tFS",variable = self.var_param,value="FS")
155         self.CB_param_FS.grid(column = 0, row= 1 , columnspan= 2,pady =
                5, padx = 5,sticky = tk.W)
156
157         self.W_entry = ttk.Entry(self.vars_iniciacion_frame,width = 6,
                justify=tk.RIGHT,font =("Arial",10))
158         self.W_entry.grid(column = 2, row =0, sticky= tk.W,padx=20,pady
                =5)
159         self.W_label = ttk.Label(self.vars_iniciacion_frame,text = "W(m)
                ",font =("Arial",10))
160         self.W_label.grid(column =3, row = 0, sticky= tk.W)
161         self.W_entry.insert(0,"10e-3")
162
163         self.da_entry = ttk.Entry(self.vars_iniciacion_frame,width = 6,
                justify=tk.RIGHT)
164         self.da_entry.grid(column = 2, row =1, sticky= tk.W,padx=20,pady
                =5)
165         self.da_label = ttk.Label(self.vars_iniciacion_frame,text = "da(
                m)")
166         self.da_label.grid(column =3, row = 1, sticky= tk.W)
167         self.da_entry.insert(0,"1e-5")
168
169         self.rb_plana = ttk.Radiobutton(self.vars_iniciacion_frame,text
                ="Grieta plana",variable =self.ac_param, value ="plana")
170         self.rb_plana.grid(column = 4, row = 0, columnspan =2, padx =20,
                 pady =5,sticky = tk.W)
171         self.rb_eliptica = ttk.Radiobutton(self.vars_iniciacion_frame,
                text ="Grieta semielíptica",variable =self.ac_param, value ="
                eliptica")
172         self.rb_eliptica.grid(column = 4, row = 1, columnspan =2, padx
                =20, pady =5,sticky = tk.W)
173
174         #boton de probar
175         self.ini_btn = ttk.Button(self.vars_iniciacion_frame,text = "
                Ejecutar iniciación",command =self.ejecutar_curvas)
176         self.ini_btn.grid(column = 0,row=2 ,columnspan=6 ,sticky=tk.W,
                padx = 5, pady = 5)
177
178         #TreeFrame
179         self.tree_frame = tk.LabelFrame(self.tabs["Iniciación"],text = "
                Datos de Iniciación")
180         self.tree_frame.place(relx =0.51,rely =0.27,relwidth=0.475,
                relheight=0.7)
181
182         self.tree_scrollbarx = ttk.Scrollbar(self.tree_frame,orient =tk.
                HORIZONTAL)
```

```
183         self.tree_scrollbary = ttk.Scrollbar(self.tree_frame,orient =tk.
                VERTICAL)
184 
185         self.tree_view =ttk.Treeview(self.tree_frame,xscrollcommand =
                self.tree_scrollbarx.set,yscrollcommand =self.tree_scrollbary.
                set)
186 
187         self.tree_scrollbarx.pack(side= tk.BOTTOM,fill =tk.X)
188         self.tree_scrollbary.pack(side= tk.RIGHT,fill =tk.Y)
189         self.tree_scrollbarx.config(command =self.tree_view.xview)
190         self.tree_scrollbary.config(command =self.tree_view.yview)
191 
192         #chart
193         self.canvas_chart = tk.LabelFrame(self.tabs["Iniciación"],text =
                 "gráfica de Iniciación")
194         self.canvas_chart.place(relx =0.01,rely =0.27,relwidth=0.475,
                relheight=0.7)
195 
196         ### Pestaña Datos
197 
198         #Label frame de datos experimentales
199         self.dat_exp_lf = ttk.LabelFrame(self.tabs["Datos"],text = "
                Datos de tensiones y deformaciones")
200         self.dat_exp_lf.place(relx = 0.01,rely = 0.01,relwidth=0.98,
                relheight=0.12)
201 
202         #combobox
203         self.combo_exp = ttk.Combobox(self.dat_exp_lf,width =50)
204         self.combo_exp.bind("<<ComboboxSelected>>",self.sel_exp)
205         self.combo_exp.grid(column = 0, row =0,padx = 5, pady = 5)
206 
207         #Boton Cargar datos
208         self.cargar_dat_btn = ttk.Button(self.dat_exp_lf, text = "Cargar
                 Datos",command= self.carga_datos)
209         self.cargar_dat_btn.grid(column = 2, row= 0, padx = 5, pady = 5)
210 
211         # Label Frame Gráficas
212         self.graf_lf = ttk.Labelframe(self.tabs["Datos"],text = "Grá
                ficas")
213         self.graf_lf.place(relx = 0.01, rely =0.15,relheight=0.85,
                relwidth= 0.48)
214 
215         self.graf_frame =tk.Frame(self.graf_lf)
216         self.graf_frame.place(x = 5,rely = 0.16, relheight=0.83,relwidth
                 =0.98)
217 
218         self.graf_opt_frame = tk.Frame(self.graf_lf,)
219         self.graf_opt_frame.place(x = 5, y = 5, relwidth =0.98,relheight
                =0.15)
220 
```

```
221         self.combo_eje_x = ttk.Combobox(self.graf_opt_frame,width =40)
222         self.combo_eje_y = ttk.Combobox(self.graf_opt_frame,width =40)
223         self.combo_eje_x.grid(column =1, row=0, pady = 5, padx = 5)
224         self.combo_eje_y.grid(column =1, row=1, pady = 5, padx = 5)
225
226         self.label_eje_x = ttk.Label(self.graf_opt_frame ,text = "Eje X")
227         self.label_eje_y =ttk.Label(self.graf_opt_frame ,text = "Eje Y")
228         self.label_eje_x.grid(column =0, row =0, padx = 5)
229         self.label_eje_y.grid(column =0, row = 1, padx = 5, pady =5)
230
231         self.btn_graf =ttk. Button(self.graf_opt_frame,text = "Actualizar
                ",command =self.actualizar_grafica)
232         self.btn_graf.grid(column = 2, row = 0, rowspan= 2, padx = 5,
                pady = 5)
233
234
235         # Label Frame Treeview
236         self.dat_tree_lf = ttk.Labelframe(self.tabs["Datos"],text = "
                Datos")
237         self.dat_tree_lf.place(relx = 0.5, rely =0.15,relheight=0.85,
                relwidth= 0.48)
238
239         self.dat_scrollbarx = ttk.Scrollbar(self.dat_tree_lf,orient=tk.
                HORIZONTAL)
240         self.dat_scrollbary = ttk.Scrollbar(self.dat_tree_lf,orient=tk.
                VERTICAL)
241
242         self.dat_tv =ttk.Treeview(self.dat_tree_lf,xscrollcommand = self.
                dat_scrollbarx.set,yscrollcommand =self.dat_scrollbary.set)
243
244         self.dat_scrollbarx.config(command =self.dat_tv.xview)
245         self.dat_scrollbary.config(command =self.dat_tv.yview)
246         self.dat_scrollbarx.pack(side= tk.BOTTOM,fill =tk.X)
247         self.dat_scrollbary.pack(side= tk.RIGHT,fill =tk.Y)
248
249         ### Pestaña Cálculo
250         self.ejec_lf = ttk.Frame(self.tabs["Cálculo"],relief = tk.GROOVE)
251         self.ejec_lf.place(x = 10, y = 10,relheight = 0.15,relwidth
                =0.48)
252
253         ## ejecucion
254         self.combo_ejec = ttk.Combobox(self.ejec_lf,width = 50)
255         self.combo_ejec.grid(column = 0, row = 0,padx =25,pady =25,sticky
                =tk.W)
256
257         self.btn_ejec = ttk.Button(self.ejec_lf,text = "Ejecutar",
                command = self.ejecutar_calculo)
```

```
258         self.btn_ejec.grid(column = 1, row =0, padx = 5,pady =25,sticky
                =tk.S)
259
260         self.btn_ejec_todo =ttk.Button(self.ejec_lf,text = "Ejecutar
                todos",command = self.ejecutar_calculo_todo)
261         self.btn_ejec_todo.grid(column = 2, row =0, padx = 5,pady =25,
                sticky =tk.S)
262
263         ## Resultados
264         self.result_ini_lf = tk.LabelFrame(self.tabs["Cálculo"],text = "
                Resultados")
265         self.result_ini_lf.place(relx = 0.5,y =10,relheight = 0.15,
                relwidth =0.48)
266
267         self.lbl_lon_ini = ttk.Label(self.result_ini_lf, text = "
                Longitud de iniciación de la grieta: ",justify =tk.LEFT)
268         self.lbl_lon_ini.pack(padx = 5, pady =5,anchor = tk.W)
269
270         self.lbl_cicl = ttk.Label(self.result_ini_lf, text = "Número de
                ciclos hasta el fallo: ",justify =tk.LEFT)
271         self.lbl_cicl.pack(padx = 5, pady =5,anchor = tk.W)
272
273         self.lbl_ubi = ttk.Label(self.result_ini_lf, text = "Resultados
                guardados en: ",justify =tk.LEFT)
274         self.lbl_ubi.pack(padx = 5, pady =5,anchor = tk.W)
275
276         self.graf_ini_lf = tk.LabelFrame(self.tabs["Cálculo"],text = "Gr
                áfica de propagación de la grieta")
277         self.graf_ini_lf.place(x =10 ,rely =0.18,relheight = 0.8,
                relwidth =0.48)
278
279         self.graf_cicl_lf = tk.LabelFrame(self.tabs["Cálculo"],text = "Gr
                áfica de ciclos totales")
280         self.graf_cicl_lf.place(relx=0.5 ,rely =0.18,relheight = 0.8,
                relwidth =0.48)
281
282         ### Pestaña Gráficas
283         self.carga_graf_lf = ttk.Labelframe(self.tabs["Gráficas"],text =
                 "Carga de datos")
284         self.carga_graf_lf.place(x = 10,y =10, relwidth =0.48,relheight
                =0.2)
285
286         self.reg_graf_lf = ttk.Labelframe(self.tabs["Gráficas"],text = "
                Gráfica de comparación")
287         self.reg_graf_lf.place(x = 10, rely =0.22,relwidth =0.48,
                relheight=0.76)
288
289         self.lon_vida_graf_lf =ttk.Labelframe(self.tabs["Gráficas"],text
                 = "Vida estimada - Longitud inicial")
```

```
290         self.lon_vida_graf_lf.place(relx =0.49,y = 10,relwidth =0.48,
                relheight = 0.48)
291
292         self.per_vida_graf_lf = ttk.Labelframe(self.tabs["Gráficas"],
                text = "Vida estimada - Porcentaje de vida inicial")
293         self.per_vida_graf_lf.place(relx =0.49,rely =0.50,relwidth =0.48,
                relheight = 0.48)
294
295         ## Carga de datos
296         self.lbl_dat_exp = ttk.Label(self.carga_graf_lf,text = "
                Resultados experimentales: ",width =85)
297         self.lbl_dat_exp.grid(column =0, row=0, padx =5, pady =2,sticky
                =tk.W)
298
299         self.lbl_dat_est =ttk.Label(self.carga_graf_lf,text = "
                Resultados estimados: ",width =85)
300         self.lbl_dat_est.grid(column =0, row =1,padx =5, pady =2,sticky
                =tk.W)
301
302         self.btn_carga_exp =ttk.Button(self.carga_graf_lf,text ="Cargar
                resultados experimentales",width =30,command =self.
                cargar_graf_dat_exp)
303         self.btn_carga_exp.grid(column=1,row =0,padx =5, pady =2,sticky
                =tk.E)
304
305         self.btn_carga_est =ttk.Button(self.carga_graf_lf,text ="Cargar
                resultados estimados",width=30, command = self.
                cargar_graf_dat_est)
306         self.btn_carga_est.grid(column=1,row =1,padx =5, pady =2,sticky
                =tk.E)
307
308         self.btn_ejecuta_graficas= ttk.Button(self.carga_graf_lf,text ="
                Ejecutar gráficas",width=60,command = self.ejecutar_graficas)
309         self.btn_ejecuta_graficas.grid(column =0, row =2, padx =5,pady
                =5,columnspan =2,sticky=tk.W)
310
311         self.mostrar_info()
312
313         ### Funciones
314     def combosel(self,event):
315         """Selecciona un valor de la lista y completa los campos con los
                valores asignados.
316         """
317         self.material = self.combo_mat.get()
318         cols = self.df_materiales.columns.values.tolist()[1:]
319
320         self.props_entries["a_0"].delete(0,tk.END)
321         self.props_entries["G"].delete(0,tk.END)
322
323         self.props_entries["a_0"].config(state=tk.DISABLED)
```

```
324         self.props_entries["G"].config(state=tk.DISABLED)
325 
326         for prop in cols:
327         #Cambiamos lo que hubiera en las entradas por las propiedas del
                material
328         #seleccionado
329             self.props_entries[prop].delete(0,tk.END)
330             self.props_entries[prop].insert(0,self.df_materiales[self.
                    df_materiales["Material"]==self.material][prop].values[0])
331 
332     def borrar_campos(self):
333         """Elimina los valores de los campos.
334         """
335         for prop in self.props:
336             self.props_entries[prop].delete(0,tk.END)
337 
338         self.props_entries["G"].config(state=tk.DISABLED)
339         self.props_entries["a_0"].config(state=tk.DISABLED)
340         self.dict_prop = {}
341         for prop in self.props:
342             self.resum_label[prop].config(text =prop+": ")
343 
344     def guardar_campos(self,event):
345         """Guarda los valores en un diccionario que posteriormente se
                utiliza para la realización
346         de los cálculos.
347         """
348         for prop in self.props:
349             #Rellenamos el diccionario con los datos de las entradas
350             try:
351                 self.dict_prop[prop] = float(self.mat_values[prop].get())
352             except ValueError:
353                 #sino son validos no los guardamos y los borramos de las
                        entradas
354                 self.props_entries[prop].delete( 0,tk.END)
355 
356         #Cambia la configuración de G y a_0 a normal
357         self.props_entries["G"].config(state=tk.NORMAL)
358         self.props_entries["a_0"].config(state=tk.NORMAL)
359 
360         #Calculamos los valores de G y a_0 y los insertamos en las
361         #entradas correspondientes.
362         self.dict_prop["G"]=self.dict_prop["E"]/(2.0*(1.0 + self.
                dict_prop["nu"]))
363         self.mat_values["G"].set(self.dict_prop["G"])
364         self.props_entries["G"].insert(0,self.mat_values["G"].get())
365 
366         self.dict_prop["a_0"]=1/np.pi*(self.dict_prop["K_th"]/(self.
                dict_prop["sigma_fl"]))**2.0
```

```
367         self.mat_values["a_0"].set(self.dict_prop["a_0"])
368         self.props_entries["a_0"].insert(0,self.mat_values["a_0"].get())
369 
370         if len(self.dict_prop)==len(self.props):
371             #Cambia las etiquetas del resumen de los datos si
372             # el diccionario está completo
373             for prop in self.props:
374                 self.resum_label[prop].config(text = "{}:\t{:.3e}\t{}".
                        format(prop,self.dict_prop[prop],self.dict_units[prop
                        ]))
375 
376     def cargar_materiales(self):
377         """Carga los materiales guardados en el excel materiales
378         al inicio del programa
379 
380         """
381         #Guardamos los materiales en el dataframe de materiales
382         self.df_materiales = pd.read_excel("materiales.xlsx",index_col=0)

383 
384         #Añadimos a la lista de materiales del combobox
385         self.comb_val = list(self.df_materiales["Material"])
386 
387 
388     def guardar_material(self):
389         """Guarda las propiedades de un material y lo agrega
390         al combobox de material
391 
392         """
393         #Obtenemos el material
394         self.material =self.combo_mat.get()
395 
396         #Añadimos el material a la lista de materiales
397         self.comb_val.append(self.material)
398         self.combo_mat.config(values=self.comb_val)
399 
400         #Obtenemos las columnas del dataframe de materiales
401         self.df_mat_cols = self.df_materiales.columns.values[1:]
402 
403         #Añadimos un diccionario con los datos del material
404         self.datos_materiales ={"Material":self.material}
405 
406         try:
407             for prop in self.df_mat_cols:
408                 self.datos_materiales[prop] = float(self.props_entries[
                        prop].get())
409 
410             #Añadimos la fila al dataframe y lo exportamos al archivo
                    excel
```

```
411             self.df_materiales= self.df_materiales.append(self.
                    datos_materiales,ignore_index =True)
412             self.df_materiales.to_excel("materiales.xlsx")
413 
414         except ValueError:
415             tk.messagebox.showerror("ERROR","Algún valor no es válido")
416             self.comb_val.pop()
417             self.combo_mat.config(values=self.comb_val)
418 
419 
420     def borrar_material(self):
421         """Borra un material específico del combobox de materiales
422 
423         """
424         #Obtiene el nombre del material
425         self.material =self.combo_mat.get()
426 
427         #Elimina del dataframe de materiales el material especificado
428         self.df_materiales = self.df_materiales.drop(self.df_materiales[
                self.df_materiales["Material"]==self.material].index)
429         self.df_materiales.to_excel("materiales.xlsx")
430 
431         #Lo eliminamos de la lista de materiales
432         self.comb_val.remove(self.material)
433         self.combo_mat.config(values=self.comb_val)
434 
435         #Borramos todos los campos
436         self.borrar_campos()
437 
438     def seleccionar__nueva_carpeta(self):
439         """Selecciona una carpeta nueva y crea las carpetas necerias
440         para guardar los archivos necesarios.
441 
442         """
443         #Preguntamos al usuario la carpeta y se cambia a ella
444         self.ruta_principal = tk.filedialog.askdirectory(title = "Abrir
                carpeta")
445         os.chdir(self.ruta_principal)
446 
447         acs_list = ["eliptica","plana"]
448         par_list =["SWT","FS"]
449         opt_list = ["datos","grafs"]
450 
451         ruta_curva_inic = os.path.join(self.ruta_principal,"curvas_inic")
452 
453         #lista con las rutas de las carpetas
454         lista_path =[]
455         lista_path.append("resultados_generales")
456 
```

```
457         for ac in acs_list:
458             lista_path.append("curvas_inic/{}".format(ac))
459             lista_path.append("grafs/{}".format(ac))
460 
461         for opt in opt_list:
462             for par in par_list:
463                 lista_path.append("resultados/{}/{}".format(opt,par))
464 
465         for path in lista_path:
466             if not os.path.exists(path):
467                 os.makedirs(path)
468         os.chdir(os.path.dirname(__file__))
469 
470     def seleccionar_carpeta(self):
471         """Selecciona la carpeta principal de ejecución del programa
472         y se cámbia a ella. Esta carpeta debe contener todas las
                carpetas
473         necesarias para el cálculo.
474         """
475         #Preguntamos al usuario la carpeta
476         self.ruta_principal = tk.filedialog.askdirectory(title = "Abrir
                carpeta")
477         # os.chdir(self.ruta_principal)
478 
479     def mostrar_info(self):
480         """Muestra la información del programa en una alerta.
481         """
482         text ="""Este programa ha sido desarrollado por David García
                Serrano
483                 para el Trabajo de Fin de Máster en Ingeniería Aeroná
                        utica. Año 2021.
484                 Para dudas acerca del programa consultar el documento del
                         TFM.
485                 Se trata de un programa en fase de desarrollo por lo que
                        pueden existir
486                 errores de implementación."""
487         tk.messagebox.showinfo("Información", text)
488 
489     def plot_iniciacion(self):
490         """Representa las curvas de iniciación.
491         Sólo las pinta en el caso de que el archivo de iniciación no
492         exista con anterioridad.
493 
494         """
495         #Borramos el contenido que hubiera con anterioridad
496         if len(self.canvas_chart.winfo_children())>=1:
497             for widget in self.canvas_chart.winfo_children():
498                 widget.destroy()
499 
500         #Cargamos una figura
```

```
501         self.figure = plt.figure(figsize =(6,4),dpi=100)
502         self.figure.add_subplot(111)
503 
504         #Representamos todas las curvas de iniciación
505         for i in range(self.n_a):
506             plt.plot(self.N_i[:,i],self.v_sigma)
507 
508         plt.grid()
509         plt.title(f"Curvas de iniciación para el parámetro {self.par}")
510         plt.xscale("log")
511         plt.xlabel("Ciclos")
512         plt.ylabel("$\sigma (MPa)$")
513         self.chart = FigureCanvasTkAgg(self.figure,self.canvas_chart)
514         self.chart.draw()
515         self.toolbar = NavigationToolbar2Tk(self.chart,self.canvas_chart)
516         self.toolbar.update()
517         self.chart.get_tk_widget().pack(side = tk.TOP,padx =5,pady =5,
                fill= tk.BOTH,expand = 1)
518 
519 
520     def ejecutar_curvas(self):
521         """Ejecuta las curvas de iniciación
522 
523         """
524 
525         #Obtenemos los datos del parámetro, paso de grieta y anchura.
526         self.par =self.var_param.get()
527         self.da =float(self.da_entry.get())
528         self.W= float(self.W_entry.get())
529         #Establecemos la ruta donde se encuentra el archivo de iniciació
                n
530         self.ini_file =self.ruta_principal+"/curvas_inic/{}/MAT_{}.dat".
                format(self.ac_param.get(),self.par)
531 
532         #Comprobamos que no existe
533 
534         if not os.path.isfile(self.ini_file):
535             if len(self.dict_prop)==len(self.props):
536             #Ejecutamos las curvas y las representamos
537                 self.N_i,self.n_a,self.v_sigma =curvas_iniciacion(par =
                        self.par, da=self.da,ac=self.ac_param.get(), W = self.
                        W, MAT=self.dict_prop,main_path=self.ruta_principal)
538                 self.plot_iniciacion()
539                 self.cargar_csv()
540             else:
541                 tk.messagebox.showerror("ERROR","No se ha especificado el
                         material. Por favor introduce las propiedades en la
                        pestaña Material.")
542         else:
```

```
543             #Si existe no realiza el cálculo y carga automáticamente los
                    datos
544             #de iniciación
545             tk.messagebox.showwarning("Atención","Ya se ha realizado la
                    ejecución de las curvas de iniciación con estos pará
                    metros, No es necesario ejecutar la iniciación")
546             self.cargar_csv()
547
548
549
550
551     def cargar_csv(self):
552         """Carga los datos de iniciación
553
554         """
555         try:
556             self.df =pd.read_table(self.ini_file,sep="\s+")
557             if len(self.df.columns.values)>1:
558                 self.tree_view.delete(*self.tree_view.get_children())
559
560                 self.tree_view["column"] = list(self.df.columns.values)
561                 self.tree_view["show"] = "headings"
562
563                 for column in self.tree_view["column"] :
564                     self.tree_view.heading(str(column), text= str(column))
565
566                 df_rows = self.df.to_numpy().tolist()
567
568                 for row in df_rows:
569                     self.tree_view.insert("","end",values = tuple(row))
570
571                 self.tree_view.pack(fill =tk.BOTH, expand=1)
572             else:
573                 tk.messagebox.showerror("ERROR","Archivo dañado, por
                        favor elimina el archivo y vuelve a ejecutar la
                        iniciación")
574         except ValueError:
575             tk.message.showerror("ERROR","Error al cargar el archivo")
576         except FileNotFoundError:
577             tk.message.showerror("ERROR","No se encuentra el archivo")
578
579
580
581     def carga_datos(self):
582         """Carga los datos de tensiones y deformaciones
583
584         """
585         #Preguntamos al usuario la ubicación de los datos
586         self.dir_exp= tk.filedialog.askdirectory(title= "Abrir carpeta",
                initialdir="/")
```

```
587
588         try:
589             #Añadimos los datos al combobox
590             self.files_exp = os.listdir(path=self.dir_exp)
591             self.combo_exp.config(value= self.files_exp)
592             self.combo_exp.set(self.files_exp[0])
593
594
595         except FileNotFoundError:
596             tk.messagebox.showerror("ERROR","En esta ruta no se
                    encuentran experimentos. Selecciona otra carpeta")
597         except IndexError:
598             tk.messagebox.showerror("ERROR","En esta ruta no se
                    encuentran experimentos. Selecciona otra carpeta")
599         self.nombre_experimentos()
600
601     def sel_exp(self,event):
602         """Selecciona los datos de tensiones y deformaciones del
                combobox
603         """
604         #Obtenemos la ruta completa de los datos
605         self.file_path = os.path.join(self.dir_exp,self.combo_exp.get())
606
607         #Cargamos los datos en un data frame y nos deshacemos de la fila
                %X
608         self.df_datos = pd.read_table(self.file_path,sep="\s+")
609         self.df_datos.Y = -self.df_datos.Y*1e-3-0.1
610         self.df_datos = self.df_datos.drop(r"%X",axis=1)
611
612         #Borramos los datos que hubiera anteriormente en el treeview
613         self.dat_tv.delete(*self.dat_tv.get_children())
614
615         #Cargamos en los combobox las columnas del dataframe y
                establecemos
616         #Y e s_xx por defecto
617         self.combo_eje_x.config(value=list(self.df_datos.columns.values))

618         self.combo_eje_y.config(value=list(self.df_datos.columns.values))

619         self.combo_eje_x.set("Y")
620         self.combo_eje_y.set("s_xx")
621
622         self.actualizar_grafica()
623
624         #Rellenamos el dataframe con los datos de tensión y deformación
625         self.dat_tv["column"] = list(self.df_datos.columns.values)
626         self.dat_tv["show"] = "headings"
627
628         for column in self.dat_tv["column"] :
629             self.dat_tv.heading(column, text= column)
```

```
630
631         df_rows = self.df_datos.to_numpy().tolist()
632
633         for row in df_rows:
634             self.dat_tv.insert("","end",values = tuple(row))
635
636
637         self.dat_tv.pack(fill =tk.BOTH, expand=1,padx = 5, pady = 5)
638
639
640     def actualizar_grafica(self):
641         """Genera una gráfica con los ejes especificados
642
643         """
644         #Borramos el contenido que hubiera anteriormente
645         if len(self.graf_lf.winfo_children())>=1:
646             for widget in self.graf_frame.winfo_children():
647                 widget.destroy()
648
649         #creamos la figura
650         self.dat_figure = plt.figure(figsize=(6,4),dpi = 100)
651         self.dat_figure.add_subplot(111)
652         plt.plot(self.df_datos[self.combo_eje_x.get()],self.df_datos[
                self.combo_eje_y.get()])
653         plt.grid()
654         plt.title(f"Gráfica de {self.combo_eje_y.get()} con respecto {
                self.combo_eje_x.get()}")
655         plt.xlabel(f"{self.combo_eje_x.get()}")
656         plt.ylabel(f"{self.combo_eje_y.get()}")
657         self.dat_chart= FigureCanvasTkAgg(self.dat_figure,self.
                graf_frame)
658         self.dat_chart.draw()
659         self.toolbar_dat = NavigationToolbar2Tk(self.dat_chart,self.
                graf_frame)
660         self.toolbar_dat.update()
661         self.dat_chart.get_tk_widget().pack(fill = tk.BOTH,expand=1)
662
663     def abrir_resultados_iniciacion(self):
664         """Abre directamente un archivo de iniciación ya existente sin
665         tener que volver a ejecutar la iniciación
666
667         """
668
669         #Borramos el contenido que hubiera en el Treeview
670         self.tree_view.delete(*self.tree_view.get_children())
671
672         try:
673             #Preguntamos al usuario el nombre del archivo y lo
                    almacenamos
674             #en el dataframe
```

```
675             self.ini_file =tk.filedialog.askopenfilename(initialdir="
                    curvas_inic/",title="Abrir archivo de iniciación",
                    filetypes=(("archivo DAT","*.dat"),("archivo csv","*.csv")
                    ,("Todos los archivos","*.*")))
676             self.df =pd.read_table(self.ini_file,sep="\s+")
677 
678         #Mostramos los posibles errores al cargar
679         except ValueError:
680             tk.messagebox.showerror("ERROR","El archivo no es válido")
681         except FileNotFoundError:
682             tk.messagebox.showerror("ERROR","No se encuentra el archivo")
683 
684         self.tree_view["column"] = list(self.df.columns.values)
685         self.tree_view["show"] = "headings"
686 
687         for column in self.tree_view["column"] :
688             self.tree_view.heading(str(column), text= str(column))
689         df_rows = self.df.to_numpy().tolist()
690 
691         for row in df_rows:
692             self.tree_view.insert("","end",values = tuple(row))
693 
694         self.tree_view.pack(fill =tk.BOTH, expand=1)
695 
696     def nombre_experimentos(self):
697         """
698         Obtiene los nombres de los experimentos a partir del nombre de
                los archivos de experimentos
699         y rellena el combobox de cálculo.
700         """
701         pattern = r"\d+_\d+_\d+"
702         self.lista_exp =[]
703         for f in self.files_exp:
704 
705             self.lista_exp.append(re.search(pattern,f).group())
706 
707         self.lista_exp =list(dict.fromkeys(self.lista_exp))
708         self.combo_ejec.config(value =self.lista_exp)
709         self.combo_ejec.set(self.lista_exp[0])
710 
711 
712     def ejecutar_calculo(self):
713         """Ejecuta el cálculo de los resultados para unos determinados
714         datos de tensiones y deformaciones.
715 
716         """
717         #Borramos el contenido que hubiera con anterioridad
718         if len(self.graf_ini_lf.winfo_children())>=1:
719             for widget in self.graf_ini_lf.winfo_children():
720                 widget.destroy()
```

```
721         if len(self.graf_cicl_lf.winfo_children())>=1:
722             for widget in self.graf_cicl_lf.winfo_children():
723                 widget.destroy()
724 
725         #Reiniciamos las etiquetas de la información del resultado
726         self.lbl_lon_ini.config(text ="Longitud de iniciación de la
                grieta: ")
727         self.lbl_cicl.config(text ="Número de ciclos hasta el fallo: ")
728         self.lbl_ubi.config(text ="Resultados guardados en: ")
729 
730         #Obtenemos los datos del parámetro y anchura
731         self.par =self.var_param.get()
732         self.W = float(self.W_entry.get())
733 
734         #Filtramos por los patrones para obtener el archivo
                correspondiente
735         pat_max =r'TENSOR_TRAC\S+_{}'.format(self.combo_ejec.get())
736         pat_min =r'TENSOR_COM\S+_{}'.format(self.combo_ejec.get())
737 
738         exp_max=   list(filter(lambda i: re.match(pat_max,i),self.
                files_exp))[0][:-4]
739         exp_min = list(filter(lambda i: re.match(pat_min,i),self.
                files_exp))[0][:-4]
740 
741         try:
742 
743             #Ejecutamos la función principal
744             a_inic,v_ai_mm, N_t_min,N_t,N_p, N_i, N_a = principal(self.
                    par,self.W,self.dict_prop,self.ac_param.get(),exp_max,
                    exp_min,self.dir_exp,ruta_curvas=self.ini_file,main_path=
                     self.ruta_principal)
745 
746             #Obtenemos la figuras y la incrustamos en el espacio
                    reservado
747             #para ella.
748             self.a_N_fig = pintar_grafica_a_N(N_a,v_ai_mm,self.par,self.
                    combo_ejec.get())
749             self.a_N_chart= FigureCanvasTkAgg(self.a_N_fig,self.
                    graf_ini_lf)
750             self.a_N_chart.draw()
751             self.a_N_TB = NavigationToolbar2Tk(self.a_N_chart,self.
                    graf_ini_lf)
752             self.a_N_TB.update()
753             self.a_N_chart.get_tk_widget().pack(fill = tk.BOTH,expand=1,
                    padx =5, pady = 5)
754 
755             self.cicl_fig = pintar_grafica_iniciacion(a_inic,v_ai_mm,
                    N_t_min,N_t,N_p, N_i,self.par, self.combo_ejec.get(),
                    main_path=self.ruta_principal)
```

```
756             self.cicl_chart= FigureCanvasTkAgg(self.cicl_fig,self.
                    graf_cicl_lf)
757             self.cicl_chart.draw()
758             self.cicl_TB = NavigationToolbar2Tk(self.cicl_chart,self.
                    graf_cicl_lf)
759             self.cicl_TB.update()
760             self.cicl_chart.get_tk_widget().pack(fill = tk.BOTH,expand=1,
                    padx =5, pady = 5)
761
762             #actualizamos la información de las etiquetas
763             self.lbl_lon_ini.config(text = self.lbl_lon_ini["text"] +"
                    {:.3f} mm".format(a_inic))
764             self.lbl_cicl.config(text = self.lbl_cicl["text"]+" {:.0f}".
                    format(N_t_min))
765             self.lbl_ubi.config(text = self.lbl_ubi["text"]+"resultados/
                    datos/{}/{}.dat".format(self.par,self.combo_ejec.get()))
766
767             #Mostramos información sobre donde está guardado el resultdo
768             tk.messagebox.showinfo("Atención","El resultado del cálculo
                    ha sido añadido al final del archivo /
                    resultados_generales/resultados.xlsx")
769
770         except KeyError:
771             #Mostramos error cuando existe un error en la carga
772             tk.messagebox.showerror("ERROR","No se ha especificado el
                    material. Por favor introduce las propiedades en la pesta
                    ña Material.")
773
774         except FileNotFoundError:
775             tk.messagebox.showerror("ERROR","No se han encotrado los
                    datos. Por favor selecciona un experimento de los que se
                    encuentran en la lista.")
776
777         except OSError:
778             tk.messagebox.showerror("ERROR","Comprueba que existe el
                    archivo de iniciación o genera uno")
779
780         except:
781             tk.messagebox.showerror("ERROR","No se han cargado los datos"
                    )
782
783     def ejecutar_calculo_todo(self):
784         """Ejecuta todos los datos cargados en el combobox de una vez
                para
785         obtener las gráficas y los resultados de la vida a fatiga
786
787         """
788         resp= tk.messagebox.askyesno("ATENCIÓN","¿Estás seguro de que
                quieres calcular todos los elementos de la lista?. Esto podrí
                a llevar horas.")
```

```
789
790         if resp:
791         #Borramos el contenido que hubiera anteriormente en los frames
792             if len(self.graf_ini_lf.winfo_children())>=1:
793                 for widget in self.graf_ini_lf.winfo_children():
794                     widget.destroy()
795             #Reiniciamos las etiquetas de las pestañas
796             self.lbl_lon_ini.config(text ="Longitud de iniciación de la
                    grieta: ")
797             self.lbl_cicl.config(text ="Número de ciclos hasta el fallo:
                    ")
798             self.lbl_ubi.config(text ="Resultados guardados en: ")
799             #Obtenemos los datos del parámetro y anchura.
800             self.par =self.var_param.get()
801             self.W = float(self.W_entry.get())
802
803             #Creamos una figura
804             self.a_N_fig =plt.figure()
805             try:
806                 for exp in self.lista_exp:
807
808                     #Obtenemos los experimentos que empiecen por estos
                            patronees
809                     pat_max =r'TENSOR_TRAC\S+_{}'.format(exp)
810                     pat_min =r'TENSOR_COM\S+_{}'.format(exp)
811
812                     exp_max=   list(filter(lambda i: re.match(pat_max,i),
                            self.files_exp))[0][:-4]
813                     exp_min = list(filter(lambda i: re.match(pat_min,i),
                            self.files_exp))[0][:-4]
814
815                     #ejecutamos la función principal para obtener los
                            resultados
816                     a_inic,v_ai_mm, N_t_min,N_t,N_p, N_i, N_a = principal(
                            self.par,self.W,self.dict_prop,self.ac_param.get(),
                            exp_max,exp_min,self.dir_exp,ruta_curvas = self.
                            ini_file,main_path=self.ruta_principal)
817
818                     pintar_grafica_a_N_todas(N_a,v_ai_mm)
819
820                     #Cargamos los datos en un chart y ponemos la barra de
                            navegación
821                     #para interactuar con la gráfica
822                     self.a_N_chart= FigureCanvasTkAgg(self.a_N_fig,self.
                            graf_ini_lf)
823                     self.a_N_chart.draw()
824                     self.a_N_TB = NavigationToolbar2Tk(self.a_N_chart,self
                            .graf_ini_lf)
825                     self.a_N_TB.update()
```

```
826                     self.a_N_chart.get_tk_widget().pack(fill = tk.BOTH,
                            expand=1,padx =5, pady = 5)
827                     self.graf_ini_lf.update()
828
829                     if exp != self.lista_exp[-1]:
830                         for widget in self.graf_ini_lf.winfo_children():
831                             widget.destroy()
832             except:
833                 tk.messagebox.showerror("ERROR","No hay elementos válidos
                        en la lista")
834
835
836
837     def cargar_graf_dat_exp(self):
838         """Carga los datos de tensiones y deformaciones
839
840         """
841
842         self.lbl_dat_exp.config(text ="Resultados experimentales: ")
843         self.ubi_dat_exp = tk.filedialog.askopenfilename(title ="Abrir
                resultados experimentales",initialdir="/",filetypes=(("
                Archivos excel","*.xlsx"),("Todos los archivos","*.*")))
844         self.lbl_dat_exp.config(text =self.lbl_dat_exp["text"]+"/..."+
                self.ubi_dat_exp[-60:])
845
846     def cargar_graf_dat_est(self):
847         """Carga los datos estimados
848         """
849         self.lbl_dat_est.config(text ="Resultados estimados: ")
850
851         self.ubi_dat_est = tk.filedialog.askopenfilename(title ="Abrir
                resultados estimados",initialdir="/",filetypes=(("Archivos
                dat","*.dat"),("Archivos excel","*.xlsx"),("Todos los
                archivos","*.*")))
852         self.lbl_dat_est.config(text =self.lbl_dat_est["text"]+"/..."+
                self.ubi_dat_est[-60:])
853
854
855     def ejecutar_graficas(self):
856         """Imprime los resultados en las graficas en la pestaña Gráfica
857
858         """
859         #Borramos lo que haya dentro del widjet para no sobreponer el
860         #siguiente cálculo
861         if len(self.reg_graf_lf.winfo_children())>=1:
862             for widget in self.reg_graf_lf.winfo_children():
863                 widget.destroy()
864         if len(self.lon_vida_graf_lf.winfo_children())>=1:
865             for widget in self.lon_vida_graf_lf.winfo_children():
866                 widget.destroy()
```

```
867         if len(self.per_vida_graf_lf.winfo_children())>=1:
868             for widget in self.per_vida_graf_lf.winfo_children():
869                 widget.destroy()
870         try:
871             # Gráfica de regresión
872             self.par = self.var_param.get()
873             self.fig_reg = plt.figure(figsize =(5,5))
874             regresion(self.par,self.ubi_dat_est,self.ubi_dat_exp)
875             self.reg_chart= FigureCanvasTkAgg(self.fig_reg,self.
                    reg_graf_lf)
876             self.reg_chart.draw()
877             self.reg_TB = NavigationToolbar2Tk(self.reg_chart,self.
                    reg_graf_lf)
878             self.reg_TB.update()
879             self.reg_chart.get_tk_widget().pack(fill = tk.BOTH,expand=1,
                    padx =5, pady = 5)
880 
881             #Gráfica de longitud - número de ciclos
882             self.fig_lon_vida = grafica_lon_vida(self.par,self.
                    ubi_dat_est)
883             self.lon_vida_chart = FigureCanvasTkAgg(self.fig_lon_vida,
                    self.lon_vida_graf_lf)
884             self.lon_vida_chart.draw()
885             self.lon_vida_TB = NavigationToolbar2Tk(self.lon_vida_chart,
                    self.lon_vida_graf_lf)
886             self.lon_vida_TB.update()
887             self.lon_vida_chart.get_tk_widget().pack(fill = tk.BOTH,
                    expand=1,padx =5, pady = 5)
888 
889             #Gráfica de porcentaje de iniciación- número de ciclos
890             self.fig_per_vida = grafica_per_vida(self.par,self.
                    ubi_dat_est)
891             self.per_vida_chart = FigureCanvasTkAgg(self.fig_per_vida,
                    self.per_vida_graf_lf)
892             self.per_vida_chart.draw()
893             self.per_vida_TB = NavigationToolbar2Tk(self.per_vida_chart,
                    self.per_vida_graf_lf)
894             self.per_vida_TB.update()
895             self.per_vida_chart.get_tk_widget().pack(fill = tk.BOTH,
                    expand=1,padx =5, pady = 5)
896 
897         except:
898             tk.messagebox.showerror("ERROR", "Asegúrese de que se han
                    cargado correctamente los archivos o que el número
                    elementos es igual en ambos archivos")
899 
900     def abrir_nuevo(self):
901         """Reinicia el programa
902         """
903         os.chdir(os.path.dirname(__file__))
```

```
904         self.destroy()
905         self.__init__()
906 
907 
908     def preguntar_salir(self):
909         """Pregunta al usuario si quiere salir del programa.
910         """
911         resp = tk.messagebox.askokcancel("Atención","¿Estás seguro de
                querer salir?")
912         if resp:
913             self.destroy()
914             os.chdir(os.path.dirname(__file__))
915 
916 if __name__ =="__main__":
917     app = programa()
918     app.mainloop()
```

A.2 Código de implementación de la fase de iniciación

Código A.2 Codigo de implementación la fase de iniciación.

```
# -*- coding: utf-8 -*-
"""
Created on:19/02/2021

@author: David García y Alejandro Quirós
"""

import os
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
from propagacion import fase_propagacion
import numpy as np
import time


def ciclos_totales(param, crit, MAT):
    """Devuelve los ciclos totales en función del criterio usado,
       utilizando
    un fsolve().

    INPUTS: param = Fatemi-Socie o Smith-Watson-Topper en un punto
            crit  = parámetro a utilizar
            MAT   = índice asignado al material

    OUTPUT: N_t  = ciclos totales"""

    #Constantes del material necesarias
    sigma_y = MAT["sigma_y"]
    sigma_f = MAT["sigma_f"]
    k       = sigma_y/sigma_f
    E       = MAT["E"]
    nu      = MAT["nu"]
    b       = MAT["b"]

    def fsolve_FS(x):
        """Funcion que utiliza el fsolve para calcular los ciclos de
           vida
        cuando se utiliza el Fatemi-Socie."""
        expr = param - ((1+nu)*sigma_f/E*(2.0*x)**b + k/2.0*(1.0
                    + nu)*sigma_f**2.0/(E*sigma_y)*(2.0*x)**(2.0*b))

        return expr

    def fsolve_SWT(x):
```

```
43          """Funcion que utiliza el fsolve para calcular los ciclos de
                vida
44          cuando se utiliza el Smith-Watson-Topper."""
45          expr = param - (sigma_f**2.0/E*(2*x)**(2.0*b))
46
47          return expr
48
49      if crit == 'FS':
50          N_t = fsolve(func=fsolve_FS, x0=10, xtol=1e-10)[0]
51
52      elif crit == 'SWT':
53          N_t = fsolve(func=fsolve_SWT, x0=100, xtol=1e-10)[0]
54
55      return N_t
56
57  ##########################################################################
58
59  def fase_iniciacion(param, sigma, crit, a_i,ac, da, W, MAT):
60      """Devuelve los ciclos de la fase de iniciación de una grieta,
            conocida
61      la tensión media desde la superficie hasta la punta de la misma.
62
63      INPUTS: param = Fatemi-Socie o Smith-Watson-Topper en un punto
64              sigma = tensión x en ese punto
65              crit  = parámetro a utilizar
66              a_i   = (m) tamaño de la grieta de iniciación
67              ac    = plana o elíptica (0 o 0.5)
68              da    = paso para realizar los cálculos
69              W     = (m) anchura del espécimen
70              MAT   = índice asignado del material
71
72      OUTPUT: N_i  = ciclos de la fase de iniciacion"""
73
74      #Calcula los ciclos totales hasta el fallo
75      N_t = ciclos_totales(param, crit, MAT)
76
77      #ind_a se utiliza para la propia fase de propagación por lo
78      #que en la fase de iniciación no es una variable relevante y puede
            tomar
79      #cualquier valor
80      ind_a = 0
81
82      #Cálculos los ciclos que necesita la grieta para propagarse
83      N_p = fase_propagacion(sigma, ind_a, a_i,ac, da, W, MAT)
84
85      #Calculamos los ciclos de iniciación
86      N_i = N_t - N_p
87
88      #Si el numero es negativo devuelve 0
89      if N_i < 0:
```

```
90          N_i = 0.0
91
92      #Para ciclos muy altos solo se tiene en cuenta la iniciación para
            evitar
93      #errores que se producen en la integración de la propagación
94      if N_t > 1.5e7:
95          N_i = N_t
96
97      return N_i
98
99  ##########################################################################
100
101 def curvas_iniciacion(par, da,ac, W, MAT,main_path=""):
102     """Escribe un archivo de texto con las curvas de
103     iniciación para distintas longitudes de grieta para un material.
104
105     INPUT:  par   = parámetro para el modelo de iniciación
106             ac      =plana o elíptica (0 o 0.5)
107             da    = paso para realizar los cálculos
108             W     = (m) anchura del espécimen
109             MAT   = índice asignado al material
110
111     OUTPUTS: MATX_par.dat = archivo con las curvas de iniciación
112             figura.png  = imagen con las curvas de iniciación"""
113
114     print(('\nCurvas de iniciación del material '
115           +'utilizando el parámetro {}\n').format(par))
116
117     #Abrimos el archivo donde se van a escribir los datos,
118     #escribimos la cabecera del mismo y creamos los vectores con las
            tensiones
119     #y los tamaños de grieta para crear las curvas de iniciación
120
121     ruta    = main_path + '/curvas_inic/{}/'.format(ac)
122     ruta_fig = main_path + '/grafs/{}/'.format(ac)
123
124
125     archivo = open('{}/MAT_{}.dat'.format(ruta, par), 'w')
126
127     archivo.write('{:.3e} '.format(0.0000))
128
129     v_sigma    = []       #Vector de tensiones
130     v_param    = []       #Vector de Fatemi-Socie o Smith-Watson-Topper
131     n_sigma    = 45          #Discretizaciones de la curva de iniciación
132
133     sigma_max  = 500.0    #(MPa) tension maxima
134     sigma_min  = 50.0     #(MPa) tension minima
135
136     delta_sigma = 10.0     #Paso de tensiones
137
```

```
138     #Cargamos propiedades del material
139     sigma_y = MAT["sigma_y"]
140     sigma_f = MAT["sigma_f"]
141     k       = sigma_y/sigma_f
142     E       = MAT["E"]
143     G       = MAT["G"]
144
145     #Generamos el vector de Fatemi-Socie o Smith-Watson-Topper
146
147     v_sigma = np.arange(sigma_min,sigma_max,delta_sigma)
148     n_sigma = len(v_sigma)
149     v_def = v_sigma/E
150     v_gamma = v_sigma/2/G
151     v_param = v_gamma*(1.0+k*v_sigma/2.0/sigma_y) if par ==’FS’ else
            v_sigma*v_def
152
153     n_a   = 100       #Número de curvas de iniciacion por material
154     a_min = 5e-5          #Tamaño más pequeño grieta
155     ex    = 1.2     #Variable para controlar como crece la diferencia
156                 #entre longitudes de grieta
157
158     v_a = [a_min*(i+1.0)**ex for i in range(n_a)] #Vector de tamaños de
            grietas
159     for a in v_a:
160         archivo.write(’{:.3e} ’.format(a))
161
162     N_i = np.zeros((n_sigma,n_a)) #inicializamos la matriz de ciclos de
            iniciación
163
164     t1 = time.time()
165     for i in range(len(v_param)):
166         archivo.write(’\n{:.3e} ’.format(v_param[i]))
167         for j,a in enumerate(v_a):
168             N_i[i,j] = fase_iniciacion(v_param[i], v_sigma[i], par, a,ac,
                    da, W, MAT)
169             archivo.write(’{:.3e} ’.format(N_i[i,j]))
170         #Pintamos en la consola el porcentaje realizado
171         print(’\r{:.2%} completado’.format((i+1.0)/len(v_param)), end = ’
                ’)
172
173     #Cerramos el archivo
174
175     archivo.close()
176     t2= time.time()
177     print("\nSe han requerido {:.2f}s".format(t2-t1))
178
179     #Pintamos las curvas de iniciación
180
181     plt.figure()
182     for i in range(n_a):
```

```
183         plt.plot(N_i[:,i],v_sigma)
184     plt.grid()
185     plt.title(f"Curvas de iniciación para el parámetro {par}")
186     plt.xscale("log")
187     plt.xlabel("Ciclos")
188     plt.ylabel("$\sigma (MPa)$")
189 
190     # #Guardamos la figura y la cerramos
191     plt.savefig(ruta_fig+f'curvas_inic_{par}.png')
192 
193     return N_i,n_a,v_sigma
194 
195 def plot_N_i(par,N_i,v_sigma,n_a):
196     plt.figure()
197     for i in range(n_a):
198         plt.plot(N_i[:,i],v_sigma)
199     plt.grid()
200     plt.title(f"Curvas de iniciación para el parámetro {par}")
201     plt.xscale("log")
202     plt.xlabel("Ciclos")
203     plt.ylabel("$\sigma (MPa)$")
204     plt.show()
```

A.3 Código de implementación de la fase de propagación

Código A.3 Código de implementación de fase de propagación.

```
# -*- coding: utf-8 -*-
"""
Created on:19/02/2021

@author: David García y Alejandro Quirós
"""

import numpy as np
from scipy.integrate import quad

MAT = {"C"    :8.83e-11,
    "n"       : 3.322,
    "f"       : 2.5,
    "l_0"     : 25e-6,
    "sigma_fl" : 169.0,
    "K_th"    : 2.2,
    "K_IC"    : 29.0,
    "sigma_y" : 503.0,
    "sigma_f" : 1610.0,
    "E"       : 71000.0,
    "nu"      : 0.33,
    "b"       : -.1553}
MAT["G"]= MAT["E"]/(2.0*(1.0 + MAT["nu"]))
MAT["a_0"] =1/np.pi*(MAT["K_th"]/(MAT["sigma_fl"]))**2.0

#########################################################################

def K_I(sigma, a, ds, W):
    """Devuelve el factor de intensidad de tensiones para un tamaño de
       grieta
    determinado utilizando una función de peso propuesta por Bueckner.

    INPUTS: sigma = (MPa) tensión perpendicular al plano de la grieta
                    (float) --> fase de iniciación
                    (list) --> fase de propagación
            a     = (m) longitud de la grieta
            ds    = (m) paso de longitudes de grieta
            W     = (m) espesor del espécimen

    OUTPUT: K_I  = (MPa m^0.5) factor de intensidad de tensiones"""

    #Funciones de peso
    m1 = 0.6147 + 17.1844*(a/W)**2.0 + 8.7822*(a/W)**6.0
    m2 = 0.2502 + 3.2889*(a/W)**2.0 + 70.0444*(a/W)**6.0

```

```
45      def integr_KI(s, sx):
46          """Realiza el cálculo de la integral del FIT"""
47
48          res = sx/s**0.5*(1.0 + m1*s/a + m2*(s/a)**2.0)
49
50          return res
51
52      #Si sigma es de tipo float, el cálculo es para la fase de iniciación
53      if type(sigma) is not list:
54
55          integral = 0.0
56          s       = ds/2.0
57          N       = int(round(a/ds, 8))
58
59          for i in range(N - 1):
60              integral += integr_KI(s, sigma)*ds
61              s       += ds
62
63          K_I     = (2.0/np.pi)**0.5*integral
64
65      #Si sigma es de tipo list, el cálculo es para la fase de propagación
66      else:
67          integral = 0.0
68          s       = ds/2.0
69
70          for i in sigma[:-1]:
71              j = sigma[sigma.index(i)+1]
72              integral += integr_KI(s, (i+j)/2.0)*ds
73              s       += ds
74
75          K_I = (2.0/np.pi)**0.5*integral
76
77      return K_I
78
79  ##########################################################################
80
81  def Phi(ac = 0.5):
82      """Devuelve el factor Phi calculado por Irwin para el caso de una
            grieta
83      elíptica.
84
85      INPUT: ac  = a/c | relación entre los semiejes
86
87      OUTPUT: Phi = factor de la grieta elíptica"""
88
89      #Calculo del factor para grietas elíptica
90      int_phi = lambda phi: np.sqrt(1.0 - (1.0 - (ac)**2.0)*(np.sin(phi))
            **2.0)
91      phi = quad(int_phi,0,np.pi/2)[0]
92      return phi
```

```
93
94
95
96 ##########################################################################
97
98 def fase_propagacion(sigma, ind_a, a_i, ac,da, W, MAT):
99     """Devuelve los ciclos de propagacion de la grieta.
100
101     INPUTS: sigma  = (MPa) tensión máxima perpendicular al plano de la
            grieta
102                      (float) --> fase de iniciación
103                      (list) --> fase de propagación
104             ind_a  = índice asociado a la longitud de grieta
105             a_i    = (m) longitud inicial de la grieta
106             ac     = plana o elíptica (0 o 0.5)
107             da     = (m) paso de longitudes de grietas
108             W      = (m) anchura del espécimen
109             MAT    = índice asignado al material
110
111     OUTPUT: N_p    = ciclos de la fase de propagacion"""
112
113     #Obtenemos las constantes del material
114     # mat_props = constantes_material(MAT)
115
116     C    = MAT["C"]
117     n    = MAT["n"]
118     f    = MAT["f"]
119     l_0  = MAT["l_0"]
120     K_th = MAT["K_th"]
121     a_0  = MAT["a_0"]
122     K_IC = MAT["K_IC"]
123
124
125     def integr_prop(x, s, ac):
126         """Realiza el cálculo de la integral de los ciclos de propagació
                n"""
127         phi = Phi(ac)
128         ki = K_I(s, x, da, W)/phi
129
130         if ki < K_th*(x**f/(x**f + a_0**f - l_0**f))**(0.5*f):
131             res = 1e20
132         else:
133             res = 1.0/(C*(ki**n
134                           - (K_th*(x**f/(x**f + a_0**f - l_0**f))**(0.5*f)
                                  )**n))
135
136         return ki, res
137
138     #Si sigma es de tipo float, el cálculo es para la fase de iniciación
139     # ac = 0.5
```

```
140
141     if ac == "plana":
142         ac = 0.0
143     elif ac == "eliptica":
144         ac =0.5
145
146     N_p = 0.0
147     a   = a_i
148     ki  = 0.0
149     if type(sigma) is not list:
150         while ki < K_IC:
151             N_p += integr_prop(a, sigma, ac)[1]*da
152             ki   = integr_prop(a, sigma, ac)[0]
153             a   += da
154
155     #Si sigma es de tipo list, el cálculo es para la fase de propagación.

156     #Se empieza la integral en la longitud de iniciación requerida y se
            va
157     #aumentando la longitud, utilizando la variable i, de forma que en
            cada
158     #vuelta del bucle aumenta en 1 el tamaño del vector de tensiones y
            la
159     #longitud de grieta consecuentemente con el paso.
160     else:
161         i =0
162         while ki < K_IC:
163             #Invertimos los vectores de tensiones, ya que para el calculo
                    de
164             #K_IC la integral se inicia al fondo de la grieta acabando en
165             #la superficie. Seleccionamos del vector completo de
                    tensiones, las
166             #componentes del mismo que van desde la superficie hasta el
                    tamaño
167             #de grieta asociado al indice ind_a
168             sxx_max = np.flipud(sigma[:ind_a + 1 + i]).tolist()
169             N_p    += integr_prop(a, sxx_max, ac)[1]*da
170             ki      = integr_prop(a, sxx_max, ac)[0]
171             a      += da
172             i      += 1
173
174     return N_p
```

A.4 Código de implementación del cálculo de vida

Código A.4 Código de implementación del cálculo de vida.

```
# -*- coding: utf-8 -*-
"""
Created on:19/02/2021

@author: David García y Alejandro Quirós
"""

import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.interpolate import interp2d, interp1d
from propagacion import fase_propagacion
import pandas as pd
from time import time
import re

def lectura_datos(ruta, exp_max, exp_min):

    """Carga los datos experimentales.

    INPUT:  ruta    = ruta para cargar los datos experimentales
            exp_max = nombre del archivo con la tensiones y defs maximas
            exp_min = nombre del archivo con la tensiones y defs minimas

    OUTPUTS: x      = (m) vector de distancias a la superficie de la
        grieta
             sxx_max = (MPa) vector de tensiones máximas en la direccion
                 x
             s_max   = (MPa) vector de tensiones máximas
             e_max   = vector de deformaciones máximas
             e_min   = vector de deformaciones minimas"""

    cols = ["X", "Y", "s_xx", "s_yy","s_zz", "s_xy", "s_xz","s_yz", "
        e_xx", "e_yy", "e_zz", "e_xy", "e_xz", "e_yz"]

    #Cargamos los datos de distancias. Cambiamos el vector para
    #que sea positivo y empiece en 0
    datos_max = pd.read_csv(f"{ruta}/{exp_max}.dat",skiprows=1,sep=' ',
        names=cols) #todos los datos para las tracciones
    datos_min = pd.read_csv(f"{ruta}/{exp_min}.dat",skiprows=1,sep =' ',
        names=cols) #todos los datos para las compresiones

    x = datos_max.Y.to_numpy()*(-1e-3)-0.1

```

```
41      #Cargamos el resto de datos
42      sxx_max = datos_max.s_xx.to_numpy().tolist()
43
44      s_max = datos_max[["s_xx", "s_yy","s_zz", "s_xy", "s_xz","s_yz"]].
            to_numpy()
45
46      e_max = datos_max[["e_xx", "e_yy", "e_zz", "e_xy", "e_xz", "e_yz"]].
            to_numpy()
47
48      e_min = datos_min[["e_xx", "e_yy", "e_zz", "e_xy", "e_xz", "e_yz"]].
            to_numpy()
49      return x, sxx_max, s_max, e_max, e_min
50
51  #
        ##############################################################################

52  #
        ##############################################################################

53
54  def indice_a(a, x):
55      """Devuelve el índice asociado a una longitud de grieta.
56
57      INPUT:  a     = longitud de grieta de iniciación
58              x     = vector de distancias a la superficie
59
60      OUTPUTS: ind_a = índice asociado a la longitud de grieta de iniciaci
                ón"""
61      a = round(a,8)
62      #Calculamos el índice para el cual tenemos esa longitud de grieta
63      for i,x0 in enumerate(x):
64          if round(x0,8) >= a:
65              ind_a = i
66              break
67
68      return ind_a
69
70  #
        ##############################################################################

71  #
        ##############################################################################

72  def rotar_matriz(alfa,matriz):
73          """Devuelve una matriz rotada según los ángulos almacenados en
                alfa.
74
75          -alfa: array de 3 componentes con los ángulos
76          -matriz: matriz de 3x3"""
77          matriz = np.array(matriz)
```

```
78          R_x  = np.array([[1.0,      0.0,                0.0],
79                           [0.0, np.cos(alfa[0]), -np.sin(alfa[0])],
80                           [0.0, np.sin(alfa[0]), np.cos(alfa[0])]])
81
82          R_y  = np.array([[np.cos(alfa[1]), 0.0, -np.sin(alfa[1])],
83                           [0.0,           1.0,           0.0],
84                           [np.sin(alfa[1]), 0.0, np.cos(alfa[1])]])
85
86          R_z  = np.array([[np.cos(alfa[2]), -np.sin(alfa[2]), 0.0],
87                           [np.sin(alfa[2]), np.cos(alfa[2]), 0.0],
88                           [0.0,            0.0,           1.0]])
89          R = R_x @ R_y @ R_z
90
91          return (R.T @ matriz @ R)
92
93  def hacer_matriz(vector,i=0):
94      """Devuelve una matriz a partir del vector de tensiones o
            deformaciones.
95
96      INPUT:
97      - vector: array de seis componentes
98      - i: fila del vector
99
100     OUTPUT:
101     - m: matriz de componentes del vector
102     """
103     vector = np.array(vector)
104     m = np.array([[vector[i,0], vector[i,3], vector[i,4]],
105                           [vector[i,3], vector[i,1], vector[i,5]],
106                           [vector[i,4], vector[i,5], vector[i,2]]])
107     return m
108
109
110 def parametro(par, MAT, x, s_max, e_max, e_min):
111     """Calcula el vector para el modelo de iniciación asociado
112     a un experimento.
113
114     INPUT:  par   = parámetro para el modelo de iniciación
115             MAT   = indice asignado al material
116             x     = (m) vector de distancias a la superficie de la
                        grieta
117             s_max = (MPa) vector de tensiones máximas
118             e_max = vector de deformaciones máximas
119             e_min = vector de deformaciones minimas
120
121     OUTPUTS: FS/SWT = vector con los Fatemi-Socie o Smith-Watson-Topper
            en cada
122              punto"""
123
124     def func_FS(alfa, j):
```

```
125         """Devuelve delta_gamma_max/2 en un punto concreto. Se utiliza
                en el
126         calculo del Fatemi-Socie."""
127         E_xyz_max = hacer_matriz(e_max,j)
128         E_xyz_min = hacer_matriz(e_min,j)
129         E_max = rotar_matriz(alfa,E_xyz_max)
130         E_min = rotar_matriz(alfa,E_xyz_min)
131
132
133         #El signo negativo se debe a que la función minimiza en vez de
134         #maximizar.
135         return -(E_max[0,1] - E_min[0,1])
136
137
138     def func_SWT(alfa, j):
139         """Devuelve el Smith-Watson-Topper asociado a un punto."""
140         E_xyz_max = hacer_matriz(e_max,j)
141         E_xyz_min = hacer_matriz(e_min,j)
142         E_max = rotar_matriz(alfa,E_xyz_max)
143         E_min = rotar_matriz(alfa,E_xyz_min)
144         S_xyz_max = hacer_matriz(s_max,j)
145         S_max = rotar_matriz(alfa,S_xyz_max)
146
147         #El signo negativo se debe a que la función minimiza en vez de
148         #maximizar.
149         return -(S_max[0,0]*(E_max[0,0]-E_min[0,0])/2.0)
150
151     #Inicializamos variables
152     alfa0 = [0.0, 0.0, 0.0] #Ángulo para primera iteración de la función
             de
153                             #minimización
154     #Limites para el angulo
155     bnds = ((-np.pi, np.pi), (-np.pi, np.pi), (-np.pi, np.pi))
156
157     sigma_y = MAT["sigma_y"]
158     sigma_f = MAT["sigma_f"]
159     k       = sigma_y/sigma_f
160
161     alfa          = np.zeros((len(x),3))
162     delta_gamma_max = np.zeros_like(x)
163     s_norm        = np.zeros_like(x)
164     FS            = np.zeros_like(x)
165     SWT           = np.zeros_like(x)
166
167     #Calculamos en cada punto el Fatemi-Socie o el Smith-Watson-Topper
            asociado
168     if par == 'FS':
169         for j in range(len(x)):
170             fs = minimize(func_FS, alfa0, bounds = bnds, args=(j),
171                           options={'disp': False})
```

```
172             delta_gamma_max[j]=-fs.fun*2.0
173             S_xyz_max = hacer_matriz(s_max,j)
174             S_max = rotar_matriz(alfa[j,:],S_xyz_max)
175             s_norm[j]=S_max[2,2]
176             FS[j]= delta_gamma_max[j]/2.0*(1.0 + k*s_norm[j]/sigma_y)
177         return FS
178
179     elif par == 'SWT':
180         for j in range(len(x)):
181             swt  = minimize(func_SWT, alfa0, bounds = bnds, args=(j),
182                             options={'disp': False})
183             alfa[j,:]=swt.x
184             SWT[j]=-swt.fun
185         return SWT
186
187 #
    ##############################################################################

188 #
    ##############################################################################

189
190 def principal(par, W, MAT,ac,exp_max, exp_min,ruta_exp,ruta_curvas=None,
    main_path=""):
191     """Estima la vida a fatiga.
192
193     INPUTS: par     = parámetro para el modelo de iniciación
194             W       = (m) anchura del espécimen
195             MAT     = indice asignado al material
196             ac      = propagación plana o elíptica
197             exp_max = nombre del archivo con la tensiones y defs máximas
198             exp_min = nombre del archivo con la tensiones y defs mínimas
199             ruta_curvas = ubicación de las curvas de iniciación
200
201     OUTPUTS: resultados.dat = actualiza el archivo de resultados con la
202             longitud de iniciacion y los ciclos de iniciacion,
                propagacion y
203             total para que se produzca el fallo
204             a_inic        = longitud de grieta de iniciación
205             v_ai_mm       = vector de longitudes de grietas en mm
206             N_t_min       = Ciclos de iniciación
207             N_t           = Vector de ciclos totales
208             N_p           = Vector de ciclos de propagación
209             N_i           = Vector de ciclos de iniciación
210             N_a           = Vector de ciclos de propagación a partir de
                la iniciación
211             exp_id.dat    = archivo con los datos de las curvas de vida
                """
212
213
```

```
214     print('Datos Experimentales:\n {}.dat\n  {}.dat\n'.format(exp_max,
215                                                             exp_min))
216     #exp_id obtiene a partir del nombre del archivo con los datos un
217     #identificador del experimento. Dependiendo del nombre del archivo
            debe
218     #modificarse.
219     pattern =r"\d+_\d+_\d+"
220     exp_id     = re.search(pattern,exp_max).group()
221
222     #Obtenemos las rutas a las carpetas necesarias para los calculos
223     # cwd        = os.getcwd()
224     # ruta_exp   = cwd + '/datos_experimentales'
225     ruta_datos = main_path+ '/resultados/datos/{}'.format(par)
226     if ruta_curvas is None:
227         ruta_curvas = main_path + '/curvas_inic/{}'.format(ac)
228         data_interp = np.loadtxt("{}/MAT_{}.dat".format(ruta_curvas, par)
                )
229     else:
230         data_interp = np.loadtxt(ruta_curvas)
231
232
233     #Cargamos los datos de las curvas de iniciación del material
234
235
236     #Separamos las curvas en las variables necesarias
237     x_interp = data_interp[0,1:] #Eje x de la matriz de interpolación
238     y_interp = data_interp[1:,0]       #Eje y de la matriz de interpolaci
            ón
239     m_N_i    = data_interp[1:,1:]       #Matriz con los ciclos de
            iniciación
240
241
242
243     #Creamos la función de interpolación
244     function_interp = interp2d(x_interp, y_interp, m_N_i, kind='quintic',
245                                bounds_error=False)
246
247     #Cargamos los datos experimentales y generamos el vector de FS o SWT
248     x, sxx_max, s_max, e_max, e_min = lectura_datos(ruta_exp, exp_max,
249                                                     exp_min)
250     param = parametro(par, MAT, x, s_max, e_max, e_min)
251
252     a_i_min = round(x[1], 8) #Tamaño mínimo de longitud de grieta de
253                              #iniciacion. Redondeamos para evitar errores
254                              #numericos.
255     a_i_max = round(x[-1], 8) #Tamaño máximo de longitud de grieta de
            iniciacion
256     da      = a_i_min     #Paso entre longitudes de grietas
257
```

```
258     #Creamos el vector de longitudes de grieta de iniciacion
259
260     v_ai = np.arange(a_i_min,a_i_max, da)#Vector de longitudes de grieta
            en m
261     v_ai_mm = v_ai*1e3       #Vector de longitudes de grieta en mm
262
263     #Calculamos los ciclos de iniciación, de propagación y totales para
            cada
264     #longitud de grieta de iniciacion
265     #Inicializamos los ciclos
266     N_i= np.zeros_like(v_ai)
267     N_p= np.zeros_like(v_ai)
268     N_t= np.zeros_like(v_ai)
269
270     for i,a in enumerate(v_ai):
271         ind_a    = indice_a(a, x) #Índice asociado a esa longitud de
                grieta
272         param_med = np.mean(param[:ind_a + 1]) #Valor medio del
                parametro para la interpolacion
273
274         #Realizamos la interpolacion para calcular los ciclos de
                iniciación
275         N_i[i] = function_interp(a, param_med)[0]
276
277         #La interpolación puede dar valores menores que 0 para ciclos
                muy bajos
278         if N_i[i] < 0:
279             N_i[i] = 0
280
281         #Calculamos los ciclos de propagación
282         N_p[i] = fase_propagacion(sxx_max, ind_a, a,ac, da, W, MAT)
283
284         #Ciclos totales
285         N_t[i] = N_i[i]+N_p[i]
286
287
288     #Calculamos el numero de ciclos hasta el fallo y la longitud de
            iniciación
289     #de la grieta, que se producen en el mínimo de la curva de ciclos
            totales
290     N_t_min  = np.min(N_t)
291     i_N_t_min = np.argmin(N_t)
292     N_i_min  = N_i[i_N_t_min]
293     N_p_min  = N_p[i_N_t_min]
294     a_inic   = v_ai_mm[i_N_t_min]
295
296
297     #Pintamos la figura con la evolución de la longitud de grieta y
            guardamos
298     #en un archivo los datos
```

```
299     ciclos = open(ruta_datos + '/{}.dat'.format(exp_id), 'w')
300     ciclos.write('{:<5}\t{:<12}\t{:<12}\t{:<12}\t{:<12}'.format('a_i', '
            N_t',
301                                                             'N_i', 'N_p',
302                                                             'N_a'))
303
304     N_a = []            #Vector de ciclos con la evolución de la grieta
305
306     for i,ai in enumerate(v_ai):
307         #Hasta la longitud de iniciación crece como los ciclos de
                iniciación
308         if ai <= a_inic*1e-3:
309             n_a = N_i[i]
310             N_a.append(n_a)
311         #A partir de la longitud de iniciación crece de acuerdo con los
                ciclos
312         #de propagación
313         else:
314             n_a = N_a[-1] + N_p[i-1] - N_p[i]
315             N_a.append(n_a)
316         n_i = N_i[i]
317         n_p = N_p[i]
318
319         ciclos.write('\n{:.3f}\t{:.6e}\t{:.6e}\t{:.6e}\t{:.6e}'.format(
                ai*1e3,
320                      n_i+n_p, n_i, n_p, n_a))
321     ciclos.close()
322
323     cols = ['exp_id','param', 'N_t_min', 'N_i_min', 'N_p_min', 'N_i_perc
            ', 'N_p_perc', 'a_inic(mm)']
324     values =[exp_id, par, N_t_min, N_i_min, N_p_min, str(float(N_i_min)/
            N_t_min*100)+"%", str(float(N_p_min)/N_t_min*100)+"%", a_inic]
325
326     result_dict = dict(zip(cols,values))
327     if os.path.isfile(main_path+"/resultados_generales/resultados.xlsx"):

328         df = pd.read_excel(main_path+"/resultados_generales/resultados.
                xlsx")
329         if len(df[df["exp_id"] ==exp_id][df["param"]==par])>0:
330             df = df.drop(df[df["exp_id"] ==exp_id][df["param"]==par].
                    index[0])
331
332         df =df.append(result_dict,ignore_index=True)
333
334     else:
335         df= pd.DataFrame([result_dict])
336
337     df.to_excel(main_path+"/resultados_generales/resultados.xlsx",index=
            False)
338
```

```
339
340
341     print('Longitud de iniciación de la grieta: {} mm'.format(a_inic))
342     print('Numero de ciclos hasta el fallo: {}\n'.format(N_t_min))
343
344     return a_inic,v_ai_mm, N_t_min,N_t,N_p, N_i, N_a
345
346 def pintar_grafica_a_N(N_a, v_ai_mm,par,exp_id):
347     fig = plt.figure('Longitud de grieta_{}_{}'.format(par,exp_id))
348     plt.xscale('log')
349     plt.xlabel('Ciclos')
350     plt.ylabel('Longitud de grieta (mm)')
351     plt.xlim([5e2,1e6])
352     plt.grid()
353     plt.plot(N_a, v_ai_mm, 'k')
354     return fig
355
356 def pintar_grafica_a_N_todas(N_a, v_ai_mm):
357     # fig = plt.figure('Longitud de grieta')
358     plt.xscale('log')
359     plt.xlabel('Ciclos')
360     plt.ylabel('Longitud de grieta (mm)')
361     plt.xlim([5e2,1e6])
362     plt.grid()
363     plt.plot(N_a, v_ai_mm, "k")
364
365
366
367 def pintar_grafica_iniciacion(a_inic,v_ai_mm, N_t_min,N_t,N_p, N_i,par,
    exp_id,main_path="" ):
368
369     fig = plt.figure(f"{exp_id}_{par}")
370     plt.title("Punto de Iniciación")
371     plt.xlabel('Longitud de iniciacion (mm)')
372     plt.ylabel('Ciclos')
373     plt.yscale("log")
374     plt.grid()
375     plt.plot(v_ai_mm, N_i, 'b')
376     plt.plot(v_ai_mm, N_p, 'k')
377     plt.plot(v_ai_mm, N_t, 'r')
378     plt.plot(a_inic, N_t_min, 'g^')
379     plt.ylim([1e3,1e8])
380     plt.legend(["Iniciación","Propagación" , "Total","Punto de iniciació
        n"])
381     plt.annotate(text="a_inic: {:.3f} mm\nN_inic: {:.0f}".format(a_inic,
        np.floor(N_t_min)),
382                 xy =(a_inic,N_t_min),
383                 xytext =(1,1e7),
384                 arrowprops=dict(facecolor ="blue",width=0.1,headwidth
                    =0.2))
```

```
385     plt.savefig(main_path+"/resultados/grafs/{}/{}.png".format(par,
            exp_id))
386     return fig
```

A.5 Código de implementación de la estadística

Código A.5 Código de implementación de la estadística.

```
# -*- coding: utf-8 -*-
"""
Created on :19/02/2021

@author: David García Serrano
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

def regresion(par,vida_estimada, vida_experimental):
    """Cálculo de la recta de regresion entre la vida estimada y la vida
        experimental
    INPUTS:
        par: parámetro de los cálculos (SWT o FS)

        vida_estimada: ubicación del archivo.dat con los resultados de
            todos los experimentos

        vida_experimental: ubicación del archivo.xlsx con los datos de
            todos los experimentos de

    OUTPUTS:
        figura: Gráfica con la representación de los puntos, la recta de
            regresión y los
        datos de los coeficientes.

    """

    data_exp = pd.read_excel(vida_experimental,index_col=0) #datos de
        vida experimental

    #podemos cargar los datos en formato .dat o .xlsx
    try:
        data_est =pd.read_table(vida_estimada,sep=r"\s+",skiprows =1,
            names =["exp_id", "param","N_t_min","N_i_min", "N_p_min","
            N_i_perc","N_p_perc","a_inic" ])

    except:
        data_est = pd.read_excel(vida_experimental)

    data_est =data_est[data_est.param==par] #Filtramos solo para ese par
        ámetro
```

```
38      dict_est = {} #Diccionario con las datos estimados
39
40
41      for j,exp in enumerate(data_est.exp_id): #Rellenamos los
            diccionarios con el valor N_t_min
42          dict_est[exp]=data_est.iloc[j].N_t_min
43
44      exps = data_exp.columns.values[1:] #vector de columnas de data_exp
45
46      #Cargamos el Dataframe con todos los datos.
47      Df_est_exp = pd.DataFrame()
48      Df_est_exp["exp_id"]=exps
49      Df_est_exp["vida_estimada"]=[dict_est[i] for i in exps]
50      Df_est_exp["vida_experimental1"]=[data_exp[i].values[0] for i in
            exps]
51      Df_est_exp["vida_experimental2"]=[data_exp[i].values[1] for i in
            exps]
52      Df_est_exp["vida_est_log"] =np.log10(Df_est_exp.vida_estimada)
53      Df_est_exp["vida_exp_log1"] =np.log10(Df_est_exp.vida_experimental1)
54      Df_est_exp["vida_exp_log2"] =np.log10(Df_est_exp.vida_experimental2)
55
56
57
58      #Hacemos la regresion con los logaritmos
59      x =np.concatenate((Df_est_exp.vida_est_log.values,Df_est_exp.
            vida_est_log.values))
60      y =np.concatenate((Df_est_exp.vida_exp_log1.values,Df_est_exp.
            vida_exp_log2.values))
61
62      #Modelo de regresión lineal
63      Lm= LinearRegression(fit_intercept=True)
64      try:
65          Lm.fit(x.reshape(-1,1),y.reshape(-1,1))
66      except:
67          Lm.fit(x.reshape(-1,1),y.reshape(-1,1))
68
69      a= Lm.coef_[0] #Pendiente de la recta de regresión
70      b = Lm.intercept_ #Residual de la recta de regresión
71      r =Lm.score(x.reshape(-1,1),y.reshape(-1,1)) # Coeficiente de
            correlación
72      xs = np.array([0.0,10.0])
73      ys = a*xs+b #Recta de regresión
74
75      #Deshacemos logaritmos
76      xs = 10**xs
77      ys =10**ys
78
79      x = 10**x
80      y =10**y
81
```

```
82      # plt.figure(figsize =(5,5))
83      plt.plot(x,y,"ob")
84      plt.xscale("log")
85      plt.yscale("log")
86      plt.xlim([np.min(x)*0.5,np.max(x)*2])
87      plt.ylim([np.min(x)*0.5,np.max(x)*2])
88      plt.xlabel('Vida estimada')
89      plt.ylabel('Vida experimental')
90      plt.plot([1e1,1e10],[1e1,1e10],'--r',label ="Recta de equivalencia")
            #recta
91      plt.plot(xs,ys,"--g",label ="Recta de regresión")
92      plt.plot([1e1,1e10],[0.5e1,0.5e10], "--y")
93      plt.plot([1e1,1e10],[2e1,2e10], "--y")
94      plt.legend()
95      plt.grid()
96
97  def grafica_lon_vida(par,vida_estimada):
98      """Función que representa la vida estimada en una gráfica la
            longitud de iniciación
99      con respecto a los ciclos
100     INPUTS:
101         -par: criterio FS o SWT
102         -vida estimada: datos con los resultados de cálculos de vida a
                fatiga
103
104     OUTPUTS:
105         -fig: Figura
106     """
107
108     #Carga de datos
109     try:
110
111         data_est =pd.read_table(vida_estimada,sep=r"\s+",skiprows =1,
                names =["exp_id", "param","N_t_min","N_i_min", "N_p_min","
                N_i_perc","N_p_perc","a_inic" ])
112     except:
113         data_est = pd.read_excel(vida_estimada)
114
115     data_est = data_est[data_est.param == par]
116
117     x = data_est.N_i_min.values
118     y = data_est.a_inic.values
119
120     fig = plt.figure(figsize=(5,3))
121
122     plt.plot(x,y,'ob')
123     plt.xscale('log')
124     plt.ylim([0,1])
125     plt.xlim([1e2,1e5])
126     plt.xlabel('Vida estimada')
```

```
127     plt.ylabel('a_inic(mm)')
128     plt.grid(which = 'major', color = 'k', linestyle='-', linewidth=0.4)
129     plt.grid(which = 'minor', color = 'gray', linestyle=':', linewidth
            =0.2)
130
131     return fig
132
133
134 def grafica_per_vida(par,vida_estimada):
135     """Función que representa la vida estimada en una gráfica el
            porcentaje
136     de vida que se lleva la iniciación.
137     INPUTS:
138         -par: criterio FS o SWT
139         -vida estimada: datos con los resultados de cálculos de vida a
                fatiga
140
141     OUTPUTS:
142         -fig: Figura
143
144     """
145
146     try:
147         data_est =pd.read_table(vida_estimada,sep=r"\s+",skiprows =1,
                names =["exp_id", "param","N_t_min","N_i_min", "N_p_min","
                N_i_perc","N_p_perc","a_inic" ])
148     except:
149         data_est = pd.read_excel(vida_estimada)
150     data_est = data_est[data_est.param == par]
151
152     x = data_est.N_i_min.values
153     y = data_est["N_i_perc"].values
154     y =np.array([float(i[:-1]) for i in y])
155
156     fig = plt.figure(figsize=(5,3))
157
158     plt.plot(x,y,'ob')
159     plt.xscale('log')
160     plt.ylim([0,100])
161     plt.xlim([1e2,1e5])
162     plt.xlabel('Vida estimada')
163     plt.ylabel('Porcentaje de iniciación %')
164     plt.grid(which = 'major', color = 'k', linestyle='-', linewidth=0.4)
165     plt.grid(which = 'minor', color = 'gray', linestyle=':', linewidth
            =0.2)
166
167     return fig
```

Apéndice B
Manual de instalación del programa

En este anexo se describen los pasos para poder instalar el programa. En primer lugar es necesario comentar que este software sólo se ha desarrollo en un sistema operativo Microsoft Windows ®, por lo que puede que en otros sistemas operativos no funcione o lo haga de forma defectuosa. A continuación se describen los pasos a seguir para instalar correctamente el software de predicción de fatiga multiaxial.

1. Descomprimir el archivo *.rar* donde se encuentran los archivos de ejecución del programa en una carpeta especifica.
2. Comprobar que Python 3.7 ó 3.8 está instalado en el ordenador. Si no lo está, es necesario dirigirse a la página *www.python.org* y descargar e instalar cualquiera de estas versiones
3. Tener instalados los módulos necesarios. Al instalar Python, normalmente se instalan los módulos **Numpy**, **Matplolib**, **re** y **os**. Por otra parte, los módulos que no se encuentran instalados por defecto son **Sklearn**, **Scipy**, **Pandas** y **Openpyxl**
4. En caso de que no estén instalados, es necesario instalar aquellos que falten. Para ello hay que dirigirse a la ventana de comandos de Windows (CMD). A continuación se describe el proceso para instalar cada módulo.

   a) Para instalar **Numpy**. Escribir en el CMD: `pip install numpy`

   b) Para instalar **Matplotlib**. Escribir en el CMD: `pip install matplotlib`

   c) Para instalar **re**. Escribir en el CMD: `pip install re`

   d) Para instalar **os**. Escribir en el CMD: `pip install os`

   e) Para instalar **Sklearn**. Escribir en el CMD: `pip install sklearn`

   f) Para instalar **Scipy**. Escribir en el CMD: `pip install scipy`

   g) Para instalar **Pandas**. Escribir en el CMD: `pip install pandas`

   h) Para instalar **Openpyxl**. Escribir en el CMD: `pip install openpyxl`

   Una vez instalado todos los módulos necesarios se podrá ejecutar el programa. El script principal se denomina *GUI_principal.py* y haciendo doble click sobre este archivo automáticamente se abrirá.

   Por favor, no elimine ningún archivo ni carpeta dentro de la carpeta donde descomprimió el archivo *.rar* o de lo contrario el programa no funcionará.

Bibliografía

[1] *Numpy*, *https://numpy.org/*, 2021.

[2] *Pandas*, *https://pandas.pydata.org/*, 2021.

[3] *Regular expressions*, *https://docs.python.org/3/library/re.html*, 2021.

[4] *Scikit-learn*, *https://scikit-learn.org/stable/index.html*, 2021.

[5] *Scipy*, *https://www.scipy.org/*, 2021.

[6] *Visualization with python*, *https://matplotlib.org/stable/index.html*, 2021.

[7] Ali Fatemi and Darrell F. Socie, *A critical plane approach to multiaxial fatigue damage including out-of-phase loading*, Fatigue Fracture of Engineering Materials and Structures **11** (1988), no. 3, 149–165.

[8] L. Grueter, W. Huget, and H. Kylla, *Weight functions and stress intensity magnification factors for elliptical and semi-elliptical cracks under variable normal stress - part i*, Materialwissenschaft und Werkstofftechnik **15** (1984), no. 1, 10–17.

[9] G. R. Irwin, *Crack-extension force for a part-through crack in a plate*, Journal of Applied Mechanics **29** (1962).

[10] Burkhard Meier, *Python gui programming cookbook: Develop functional and responsive user interfaces with tkinter and pyqt5*, 3 ed., Packt Publishing, 2019.

[11] C Navarro, S Munoz, and J Dominguez, *On the use of multiaxial fatigue criteria for fretting fatigue life assessment*, International Journal of Fatigue **30** (2008), no. 1, 32–44.

[12] C Navarro Pintado, *Iniciación y crecimiento de grietas en fatiga por fretting*, Ph.D. thesis, PhD thesis, Universidad de Sevilla, 2005.

[13] Python, *Welcome to python.org*, *https://www.python.org*, 2021.

[14] Alejandro Quirós Rodríguez, *Implementación de un modelo de predicción de vida a fatiga en código libre*, Universidad de Sevilla (2019).

[15] Kaveh Samadian and Wim De Waele, *Fatigue crack growth model incorporating surface waviness for wire+ arc additively manufactured components*, Procedia Structural Integrity **28** (2020), 1846–1855.

[16] G. C. Sih, *Methods of analysis and solutions of crack problems*, Noordhoff International, 1973.

[17] KNua Smith, *A stress-strain function for the fatigue of metals*, Journal of materials **5** (1970), 767–778.