Trabajo Fin de Máster
Máster en Ingeniería Aeronáutica

Implementación de un modelo de predicción de vida a fatiga en código libre

Autor: Alejandro Quirós Rodríguez

Tutor: Carlos Navarro Pintado

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

Sevilla, 2019

Trabajo Fin de Máster
Máster en Ingeniería Aeronáutica

Implementación de un modelo de predicción de vida a fatiga en código libre

Autor:
Alejandro Quirós Rodríguez

Tutor:
Carlos Navarro Pintado
Catedrático

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

Sevilla, 2019

Trabajo Fin de Máster: Implementación de un modelo de predicción de vida a fatiga en código libre

Autor: Alejandro Quirós Rodríguez
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

En primer lugar, agradecer a Carlos Navarro su ayuda para que este trabajo sea una realidad. También, agradecer a mi familia y amigos su apoyo durante estos dos años en Sevilla.

Resumen

La fatiga juega un papel relevante en muchos sectores industriales; especialmente en el aeronáutico, en el que ha sido causante de graves problemas cuando los diseños no han sido adecuados. Se produce en componentes sometidos a cargas fluctuantes en el tiempo, generándose grietas que pueden propagarse y llegar a producir el fallo de la pieza. Se puede ver perjudicada por la presencia de ciertos condicionantes que pueden facilitar su aparición y propagación, como pueden ser la entallas, que actúan como concentradores de tensiones.

La fatiga presenta una alta dependencia, sobre todo en la fase inicial de la grieta, de la microestructura del material. Esto conlleva que el fallo en dos piezas producidas de la misma forma y con el mismo material, sometidas a los mismo esfuerzos se produzca en momentos diferentes. Por lo tanto, resulta imposible predecir con exactitud cuando fallará una pieza ya que la propia física del problema lo impide. Pese a esta dispersión de los resultados, es de vital importancia desarrollar modelos que tengan la capacidad de predecir cuando se va a producir el fallo; lo que permite el diseño de componentes mejores.

En este trabajo se ha implementado un modelo de fatiga en Python que permite estimar la vida del componente, la duración de las distintas fases de vida de la grieta, cuando se produce la transición entre las fases y cual es la evolución de la longitud de la grieta en función de los ciclos de carga. Se ha organizado de la siguiente manera; en el Capítulo 2 se presenta el modelo de vida a fatiga en el que se ha basado este trabajo, mostrándose las distintas ecuaciones que serán necesarias para realizar los cálculos. En el Capítulo 3 se presentan los datos experimentales utilizados para realizar las simulaciones y validar el código, así como las propiedades del material empleado. En el Capítulo 4, en primer lugar, se detalla el funcionamiento general del código implementado; para posteriormente introducirse en las funciones de cada *script* para explicar su utilidad. Por último, en el Capítulo 5, se muestran los resultados obtenidos para la vida estimada y la evolución de la longitud de la grieta y se comparan con los resultados mostrados en el Capítulo 3.

Abstract

Fatigue plays an important role in several industrial sectors; specially in aeronautics, where it is the reason behind many design related issues. It is produced in components with fluctuating loads and it generates cracks that can propagate and end in a component failure. The effects of fatigue become larger in presence of some factors that may accelerate its apparition and propagation; such as notches, which produce stresses concentration.

Fatigue is highly dependent in the material microstructure, mainly during the initial phase of the crack. The failure in two components made using the same tools, with the same material and working under the same loads, will happen at different times. Therefore, it is impossible to exactly predict when the component is going to experiment a failure due to the physics behind the problem. Despite the scattering in the results, it is a main concern to develop models that can predict when a failure is going to happen allowing the design of better components.

In this work, a fatigue model has been implemented in Python; which can estimate the fatigue life of the component, the duration of each crack phase, when the transition of each phase is going to happen and how the crack grows as a function of the number of load cycles. The work is organized as follows; in Chapter 2, the fatigue life model is presented, showing all the required equations needed to make the different calculations. In Chapter 3, experimental data used to make the simulations and validate the code is presented; together with the properties from the material used. In Chapter 4, at first, the general working of the code is introduced; later, each function of the scripts are described. Finally, in Chapter 5, the results obtained with the code are shown and compared with the experimental data in Chapter 3.

Índice

Índice de Figuras

Índice de Tablas

Índice de Códigos

1 Introducción

La fatiga es un proceso de daño de material. Se produce cuando una pieza es sometida a una carga fluctuante en el tiempo y se caracteriza por la aparición de grietas que crecen desde la superficie hacia el interior. La importancia del estudio de la fatiga radica en el hecho de que la carga a la que se produce la rotura es inferior al límite de rotura del material.

La fatiga por *fretting* es un caso especial de fatiga, en el que la pieza es sometida a una carga de contacto. Estas cargas de contacto producen un efecto similar al que provocaría un concentrador de tensiones, lo que se traduce en una reducción de la vida de la pieza. En el *fretting*, las piezas están sometidas a una carga de contacto normal N y una carga de contacto tangencial Q. Aparte, en la fatiga por *fretting* la pieza está sometida a una tensión global σ fluctuante en el tiempo.

Durante la vida de una grieta se encuentran dos fases diferenciadas. En la primera, conocida como fase de iniciación, la grieta crece formando un ángulo con la superficie. Cuando la misma alcanza una determinada longitud, gira para crecer perpendicularmente. A esta segunda fase se le conoce como fase de propagación.

Para estudiar la iniciación de las grietas durante la fase de iniciación se puede hacer uso de la curva S-N, que se obtiene experimentalmente. Esta curva relaciona tensiones con número de ciclos para estados de carga uniaxiales. Sin embargo, la fatiga por *fretting* presenta un estado de tensiones multiaxial, además de un fuerte gradiente de tensiones cerca de la superficie; por lo que es necesario recurrir a criterios multiaxiales que tengan en cuenta estas características. Para estudiar la fase de propagación se recurre a la mecánica de la fractura, donde se hace uso del factor de intensidad de tensiones y una ley de crecimiento, como podría ser la Ley de Paris.

A la hora de predecir la vida a fatiga existen modelos que suponen que toda la vida es iniciación, que toda la vida es propagación o que combinan ambas fases. Los modelos basados en la iniciación funcionan bien cuando la fase de iniciación es mayor que la de propagación. Aunque son sencillos de aplicar, no tienen en cuenta lo que ocurre fuera de la zona en la que se evalúan las tensiones.

Los modelos basados en la propagación presentan la dificultad de tener que elegir una longitud a partir de la cual propagar la grieta. Esta longitud puede variar en gran medida dependiendo del tipo de ensayo o de la geometría, por lo que sería necesario basarse en algún parámetro para definirla correctamente.

En cuanto a los modelos que combinan tanto la iniciación como la propagación, la principal dificultad vuelve a ser la elección de la longitud de la grieta a partir de la cual se pasa de la fase de iniciación a la fase de propagación, conocida como longitud de iniciación. Existen modelos en los que esta longitud es fija, dependiendo esta longitud del criterio del autor y modelos en los que es variable.

El objetivo del presente trabajo es implementar en Python un modelo de predicción de vida a fatiga. En primer lugar se presentará el modelo utilizado, luego se realizará una descripción del código del

programa y finalmente se compararán los resultados obtenidos con resultados experimentales para valorar el funcionamiento del programa.

1.1 Python

Python es un lenguaje de programación creado a finales de los años ochenta. Tiene una licencia de código abierto administrada por la organización *Python Software Foundation*, llamada *Python Software Foundation License* [1] y compatible con la Licencia pública general de GNU. Esta licencia cubre todo lo añadido a Python desde la versión 2.0.1, mientras que algunos fragmentos de código de versiones previas están cubiertos por licencias anteriores. Existen dos versiones del lenguaje bajo soporte, Python 2 y Python 3. Python 2 fue desarrollado antes y ya no recibe actualizaciones de importancia, mientras que Python 3 sigue bajo desarrollo activo [2]. En este trabajo se utiliza Python 2.

Figura 1.1 Logo de Python [2].

Se trata de un lenguaje interpretado, orientado a objetos y tipado dinámicamente. Una de sus principales características es la existencia de multitud de módulos desarrollados por terceros que extienden sus capacidades. En este trabajo se hará uso de NumPy, SciPy, sklearn y matplotlib. NumPy introduce nuevas funciones y clases de datos que, por ejemplo, permiten realizar cálculos con vectores y matrices. SciPy incorpora rutinas que permiten realizar interpolación u optimización de funciones entre otros muchos aspectos. Sklearn incorpora funciones y clases para el análisis de datos, en este caso se utilizará para realizar regresiones lineales. Por último, matplotlib permite la realización de figuras. Todos los módulos utilizados se encuentran disponibles en la plataforma de código libre Anaconda [3].

2 Modelo de predicción de vida

El modelo utilizado en este trabajo; propuesto por C. Navarro [4], combina las fases de iniciación y propagación. La vida total será la suma de la obtenida en ambas fases, que se tratarán por separado. Será necesario definir la longitud de iniciación, que separará una de la otra. En este caso, el modelo utilizado emplea una longitud de iniciación variable.

Para cada caso de estudio se generará un vector de longitudes de iniciación. Para cada longitud de iniciación se calcularán los ciclos necesarios para iniciar la grieta hasta esa longitud y los ciclos necesarios para propagar dicha grieta desde la longitud de iniciación hasta el punto de rotura. Más adelante se describirán los modelos empleados para calcular los ciclos de ambas fases.

De esta forma se obtendrá una curva con la vida total, suma de la iniciación y la propagación, en función de la longitud de iniciación; el mínimo de dicha curva representa la vida de la pieza al tratarse del punto más desfavorable. En la Figura 2.1 se muestra un ejemplo de estas curvas para un caso concreto. Se puede ver como para longitudes de iniciación pequeñas, la fase de propagación es mucho más determinante que la fase de iniciación. Para longitudes de iniciación superiores la tendencia cambia y es la fase de iniciación la que cobra más relevancia.

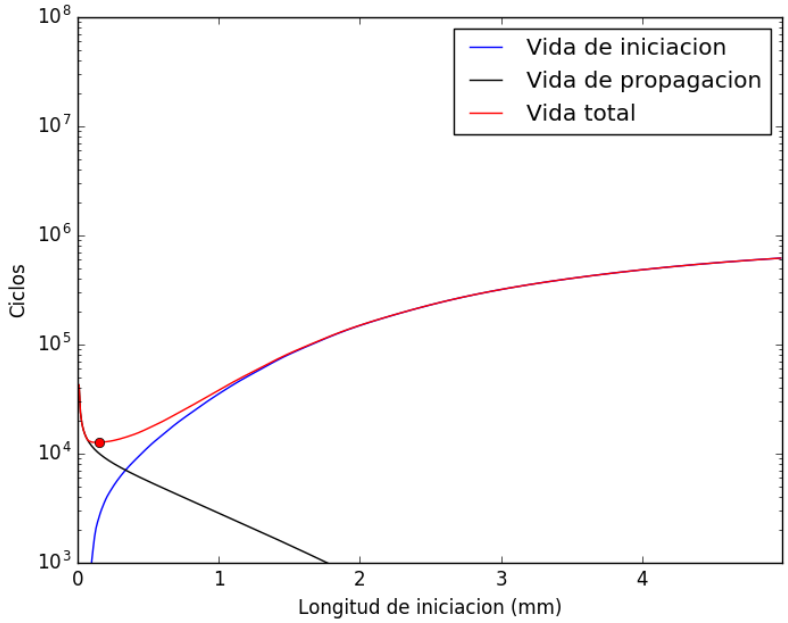


Figura 2.1 Ejemplos de curvas de vida.

2.1 Modelos de iniciación

Para la fase de iniciación se hará uso de dos parámetros de fatiga multiaxial que se definirán a continuación. Para cada uno de ellos se procederá de la siguiente forma. En primer lugar, se obtendrá la curva que relaciona el parámetro con la vida total en el caso de tensión uniaxial y ciclo de carga simétrico. Una vez obtenida esta curva, se corregirá la misma para diferentes longitudes de iniciación. Para ello, se restará a la vida total anterior los ciclos de propagación necesarios para propagar la grieta desde esa longitud de iniciación hasta el fallo. Para el cálculo de estos ciclos de propagación se considerará que la distribución de tensiones es constante. Estas curvas de iniciación obtenidas al corregir la curva que relaciona el parámetro con la vida total son independientes del experimento ya que solo dependen del criterio de fatiga multiaxial empleado y de las propiedades del material.

2.1.1 Fatemi-Socie

Fatemi y Socie [5] proponen un criterio que utiliza el incremento de deformaciones tangenciales en el plano en el que es máximo ($\Delta\gamma_{max}$), así como la tensión normal máxima perpendicular al plano en el que se produce el incremento de deformaciones tangenciales (σ_{max}). En la Ecuación 2.1 se define el parámetro.

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

En esta ecuación, k es la relación entre el límite elástico (σ_y) y el coeficiente de resistencia a fatiga (σ'_f) $k = \sigma_y/\sigma'_f$. En el caso de fatiga uniaxial de ciclo simétrico y utilizando un cálculo elástico de las tensiones, se puede hacer uso de la expresión mostrada 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 y b es el exponente de resistencia a fatiga. A partir de esta expresión es posible obtener la curva FS-N_t. Para cada longitud de iniciación concreta se corrige esta curva tal y como se comentó anteriormente. En la Figura 2.2 se muestran; a modo de ejemplo, tres curvas de iniciación correspondientes a tres longitudes de iniciación distintas, junto con la curva FS-N_t. Se puede comprobar como para ciclos muy elevados, no existen diferencias apreciables entre las distintas curvas de iniciación. Sin embargo, para ciclos bajos, las curvas empiezan a separarse progresivamente.

2.1.2 Smith-Watson-Topper

Smith et al. [6] proponen un criterio en el que intervienen el incremento de la deformación ($\Delta\varepsilon$) y la tensión normal máxima (σ_{max}). El parámetro se define para la orientación en la que el producto de ambos es máximo.

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

En el caso de fatiga uniaxial de ciclo simétrico y considerando que el cálculo de tensiones es elástico, se puede utilizar la Ecuación 2.4.

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

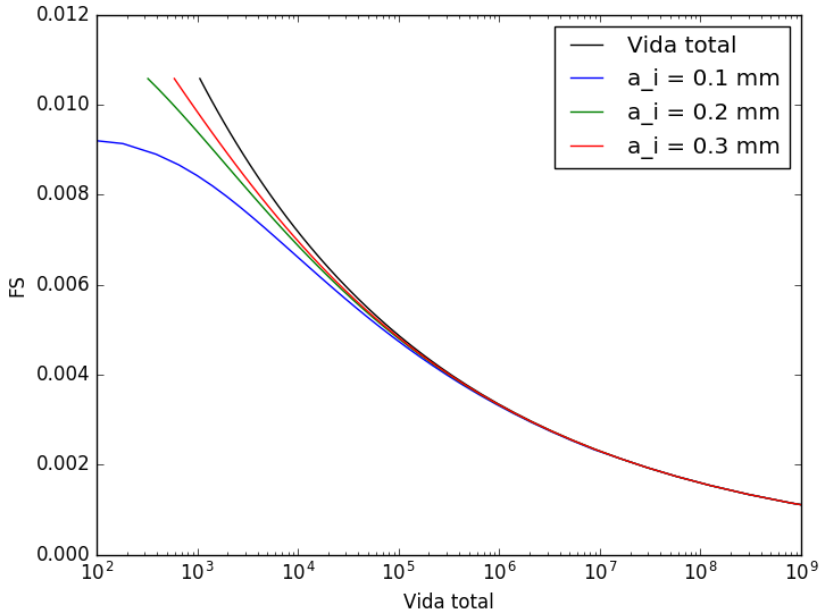


Figura 2.2 Curvas de iniciación FS-N_i.

A partir de la expresión anterior se obtiene la curva $SWT - N_t$. Corrigiendo esta curva para cada longitud de iniciación se obtienen las curvas de iniciación. En la Figura 2.3 se muestran tres ejemplos de curvas de iniciación para este parámetro junto con la curva $SWT - N_t$. Al igual que ocurría en el criterio de Fatemi-Socie, para ciclos altos las curvas obtenidas son iguales; mientras que para ciclos bajos se separan.

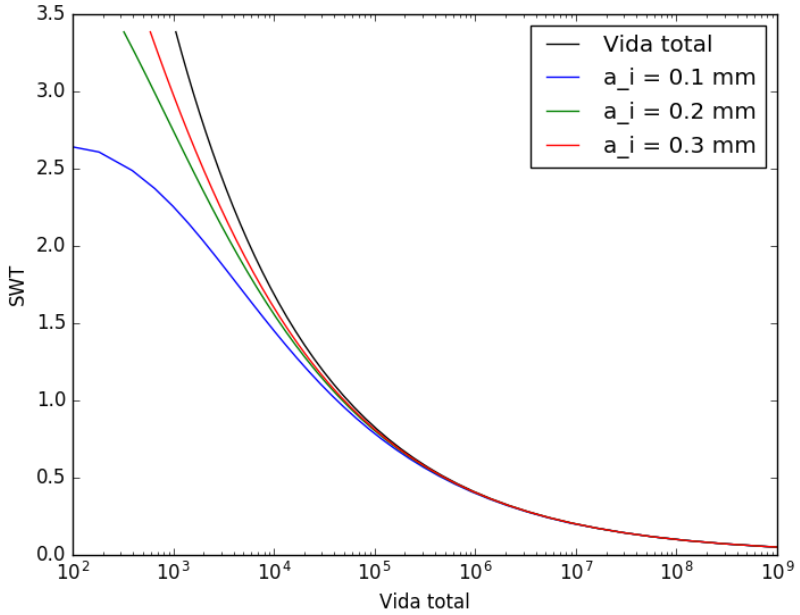


Figura 2.3 Curvas de iniciación SWT-N_i.

Obtenidas las curvas de iniciación que relacionan el valor de un cierto parámetro de fatiga multiaxial con el número de ciclos de iniciación, ya se puede obtener la curva de vida de iniciación que se representó en la Figura 2.1. La curva de vida de iniciación, a diferencia de las curvas de iniciación representadas en las Figuras 2.2 y 2.3, varía para cada experimento ya que para obtenerlas hay que utilizar los datos experimentales con las tensiones y deformaciones presentes en el espécimen.

Para obtener la curva de vida de iniciación de un experimento concreto, en primer lugar, para una profundidad determinada habrá que calcular el valor medio del parámetro desde la superficie hasta esa profundidad a partir de las tensiones y deformaciones. Esta profundidad equivaldrá a la longitud de iniciación a la cual se quiere calcular el número de ciclos de iniciación. Una vez obtenido el valor medio del parámetro, se puede acceder a la curva de iniciación que se corresponda con esa longitud de iniciación y se podrá obtener el número de ciclos de iniciación. En la Figura 2.4 se muestra un esquema del procedimiento a seguir para obtener los ciclos de iniciación correspondientes a dos longitudes de iniciación distintas, en este caso para Smith-Watson-Topper. Repitiendo este procedimiento para todo el rango de longitudes de iniciación se obtendrá la curva de vida de iniciación. Para crear este esquema explicativo se supone una distribución de tensiones y deformaciones tal que la distribución de SWT es lineal en el rango utilizado.

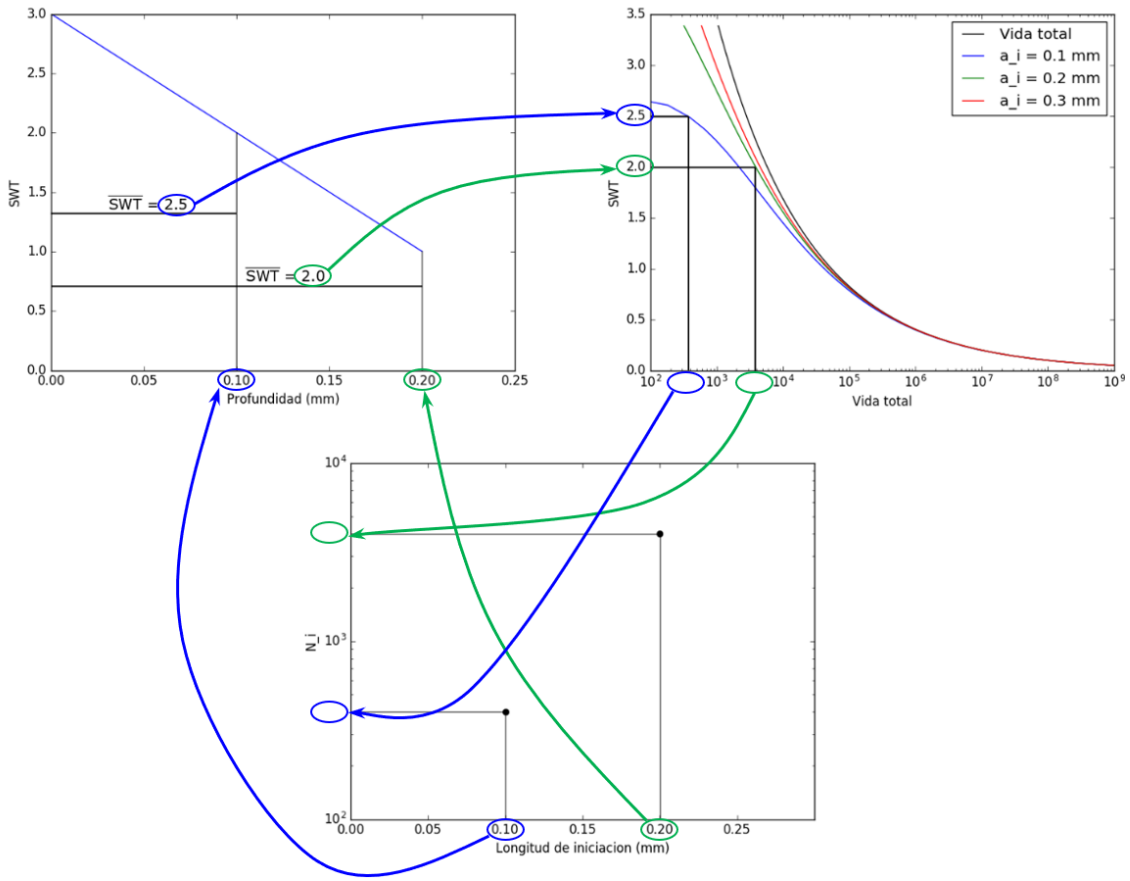


Figura 2.4 Esquema de obtención de las curvas de vida de iniciación.

2.2 Modelo de propagación

Para realizar el cálculo de los ciclos de propagación se hará uso de la mecánica de la fractura para propagar una grieta desde una longitud inicial hasta el fallo. En el modelo propuesto, esta longitud inicial será la longitud de iniciación descrita anteriormente. Por otro lado, se considerará que la grieta crece en línea recta. Esto no es cierto para toda la vida, pero sí lo es para la mayor parte de la misma, por lo que se puede considerar una buena aproximación. Además, se realiza la hipótesis de que el factor de intensidad de tensiones en modo II es despreciable frente al modo I.

2.2.1 Factor de intensidad de tensiones

El primer paso para calcular los ciclos de propagación es obtener el factor de intensidad de tensiones. Se empleará una función de pesos propuesta por Bueckner [7].

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

En está expresión, a representa la longitud de la grieta, s es una variable de integración y m_1 y m_2 son los dos parámetros mostrados a continuación.

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

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

Como se puede ver, m_1 y m_2 dependen del cociente a/W, donde W es la anchura del espécimen.

Una vez definida la función de pesos se puede calcular el factor de intensidad de tensiones en modo I utilizando la siguiente expresión.

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

En esta integral, σ_x es la tensión normal en la dirección perpendicular al plano de la grieta. La función de pesos descrita anteriormente es válida para grietas bidimensionales; por ello, será necesario corregirla por un factor para el caso de grietas tridimensionales como las que se producen en el *fretting*. Se hará uso del factor Φ calculado por Irwin [8].

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

Este factor depende de la relación a/c, que representa la relación de aspecto de la grieta. Considerando que la grieta tiene forma elíptica, a sería el semieje hacia el interior y c el semieje en la superficie. La definición de esta relación será clave a la hora de obtener los resultados. El factor de intensidad de tensiones es corregido dividiéndolo por el factor Φ.

2.2.2 Ley de propagación

Una vez conocido el modelo utilizado para calcular el factor de intensidad de tensiones, es necesario decidir un modelo para realizar la propagación de la grieta. En este trabajo se hará uso de una ley de crecimiento de grieta basada en la ley de Paris, en la que se utiliza un umbral de crecimiento modificado según la longitud de la grieta [4]. Se asume que en la propagación de la grieta solo

intervienen las tensiones máximas, ya que son estas las únicas que pueden provocar que la grieta se abra.

$$\frac{\mathrm{d}a}{\mathrm{d}N} = C\left(\Delta K^n - \left(\Delta K_{th\infty}\cdot\left(\frac{a^f}{a^f + a_0^f - l_0^f}\right)^{1/2f}\right)^n\right) \tag{2.10}$$

En esta ecuación C es el coeficiente de la ley de crecimiento; ΔK es el factor de intensidad de tensiones para las tensiones máximas, ya que como hemos mencionado anteriormente se va a considerar que la propagación solo depende de estas tensiones; n es el exponente de la ley de crecimiento; $\Delta K_{th\infty}$ es el umbral de crecimiento; f es un parámetro en aproximación al diagrama Kitagawa-Takahashi; a_0 es el parámetro de El Haddad y l_0 es la distancia a la primera barrera microestructural.

3 Datos experimentales

Para realizar la validación del código se compararán los resultados del modelo con los obtenidos experimentalmente por V. Martín [9], donde se analiza el tratamiento superficial de *shot peening* en fatiga por *fretting*. En concreto se compararán los resultados de vida a fatiga experimentales de tres grupos de ensayo distintos; un primer grupo de probetas sin tratamiento superficial, un segundo grupo de probetas sometido a *shot peening* y un tercero sometido a electroerosión. Los ensayos se han realizado para distintas combinaciones de tensiones axiales y cargas de contacto normales y tangenciales. A continuación, en la Tabla 3.1 se muestran los resultados obtenidos experimentalmente para los distintos grupos de ensayo. Se puede comprobar como las probetas sometidas a *shot peening* han aumentado en gran medida su vida a fatiga. Sin embargo, las sometidas a electroerosión la han visto reducida.

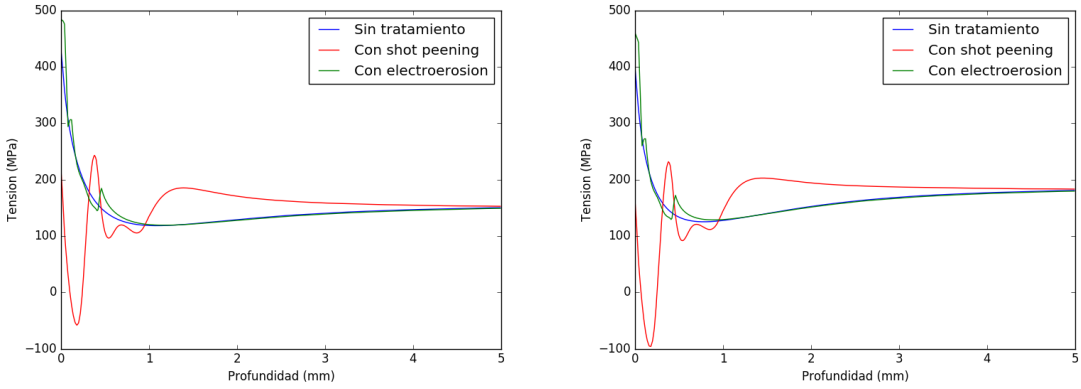
Para cada uno de estos experimentos se tiene el tensor de tensiones y deformaciones a lo largo de una línea perpendicular a la superficie. En la Figura 3.1 se muestran las tensiones σ_x, tensiones normales máximas perpendiculares al plano de la grieta, a lo largo de dicha línea para dos combinaciones de carga distintas. En todos los ensayos, para profundidades grandes se puede comprobar como los valores de las tensiones tienden al valor de la tensión axial aplicada. Las diferencias en las distintas curvas se encuentran cerca de la superficie. En el caso de los ensayos sin tratamiento se produce una disminución exponencial de las tensiones. En el caso de los ensayos con electroerosión se puede comprobar como se han introducido tensiones residuales de tracción, aunque el comportamiento es muy similar a los casos sin ningún tipo de tratamiento. Por último, en los ensayos sometidos a *shot peening* se puede comprobar como existen tensiones residuales elevadas que modifican por completo el comportamiento de las curvas.

A parte de los resultados experimentales de la vida a fatiga para las distintas combinaciones de carga y los tensores de tensiones y deformaciones en cada caso, se tienen longitudes de grieta experimentales para varios ensayos hechos para la combinación de cargas $\sigma = 110 \mid N = 971 \mid Q = 5429$ sin tratamiento superficial. En concreto, se tienen las longitudes de grietas para dos ensayos interrumpidos y para dos ensayos llevados hasta el fallo, en el que se ha medido la longitud de la grieta secundaria.

Tabla 3.1 Vidas a fatiga experimentales [9].

σ [MPa]	N [N]	Q [N]	Sin tratamiento		Con *shot peening*		Con electroerosión	
70	971	6629	165696	316603	5000000 [a]	5000000 [a]	148020	170073
110	971	5429	112165	126496	980678	1811104	-	-
110	1257	5429	113799	120663	1649736	1941545	-	-
110	1543	4217	88216	89376	1110174	1117513	-	-
110	1543	5429	82559	87481	810402	1008310	-	-
150	971	3006	59234	60040	231459	665167	-	-
150	971	4217	60288	67776	634259	678676	-	-
150	971	5429	47737	51574	631491	707514	-	-
150	1543	3006	19223	39408	275417	198884	-	-
150	1543	4217	50369	39001	234651	497260	-	-
150	1543	5429	50268	39202	290460	482862	-	-
150	2113	3006	34904	41002	140189	267873	14239	24875
150	2113	4217	34716	40004	166088	201105	-	-
150	2113	5429	32339	36431	66564	190763	16783	19005
175	971	3006	26587	31815	364153	366164	19267	23803
175	971	4217	27724	32843	308455	235467	-	-
175	971	5429	29100	35171	338774	413654	23591	25318
175	1543	3006	30154	31224	163474	164835	-	-
175	1543	4217	34748	34930	164384	169382	-	-
175	1543	5429	28005	33349	228379	231132	-	-
175	2113	3006	21207	21669	68801	83544	16743	19173
175	2113	4217	26989	28595	121642	127770	-	-
175	2113	5429	28112	28178	94741	146553	16543	16100

[a] Ensayos donde no se produjo rotura



(a) Tensiones para la combinación $\sigma = 150 \mid N = 2113 \mid Q = 5429$.

(b) Tensiones para la combinación $\sigma = 175 \mid N = 971 \mid Q = 5429$.

Figura 3.1 Ejemplos de la evolución de las tensiones σ_x.

3.1 Aluminio 7075-T651

Los ensayos se realizaron en probetas de aluminio 7075-T651, por lo que serán las características de este material las incorporadas al programa. A continuación, en la Tabla 3.3 se muestran las

Tabla 3.2 Longitudes de grieta experimentales [9].

Ensayos interrumpidos				Ensayos hasta rotura	
40000 ciclos		93746 ciclos		112165 ciclos	126496 ciclos
0.23300 mm	0.21190 mm	0.50000 mm	0.38288 mm	0.57134 mm	0.55667 mm

propiedades del material utilizadas, donde se distingue si las propiedades relacionadas con la fatiga son para $R = 0$ o $R = -1$, siendo $R = \sigma_{min}/\sigma_{max}$ el cociente entre las tensiones mínimas y máximas.

Tabla 3.3 Propiedades aluminio 7075-T651 [10].

Coeficiente de la ley de crecimiento ($R = 0$)	C	$8.83 \cdot 10^{-11}$
Exponente de la ley de crecimiento ($R = 0$)	n	3.322
Parámetro en aproximación al diagrama Kitagawa-Takahashi	f	2.5
Distancia de la superficie a la primera barrera microestructural	l_0	$2.5 \cdot 10^{-5}$ m
Umbral de crecimiento ($R = 0$)	$\Delta K_{th\infty}$	2.2 MPa$\sqrt{\text{m}}$
Tenacidad a fractura	K_{IC}	29 MPa$\sqrt{\text{m}}$
Módulo de Young	E	71000 MPa
Coeficiente de Poisson	ν	0.33
Límite elástico	σ_y	503 MPa
Exponente de resistencia a fatiga ($R = -1$)	b	-0.1553
Coeficiente de resistencia a fatiga ($R = -1$)	σ'_f	1610 MPa
Límite de fatiga ($R = -1$)	$\Delta\sigma_{fl}$	169 MPa

A parte de estas propiedades, será necesario conocer otras dos que se pueden obtener a partir de las anteriores. Por un lado se calculará el módulo de cizalladura, G. Este se puede obtener utilizando la Ecuación 3.1.

$$G = \frac{E}{2(1+\nu)} \tag{3.1}$$

Por otro lado será necesario calcular el parámetro de El Haddad, que en algunos modelos se utiliza para diferenciar la fase de iniciación de la fase de propagación. En la Ecuación 3.2 se muestra la definición del parámetro.

$$a_0 = \frac{1}{\pi}\left(\frac{\Delta K_{th\infty}}{\Delta\sigma_{fl}}\right)^2 \tag{3.2}$$

Debido a que $\Delta K_{th\infty}$ y $\Delta\sigma_{fl}$ se han especificado para dos R distintos, se va a realizar la suposición de que $\Delta\sigma_{fl}$ para $R = 0$ es el doble que para $R = -1$, por lo que en esta expresión $\Delta\sigma_{fl} = 2 \cdot 169 = 338$ MPa.

4 Descripción del código

El código se ha dividido en cuatro *scripts*, sirviendo cada uno a un propósito distinto. En primer lugar, en el *script propagacion.py* se han implementado las funciones necesarias para calcular los ciclos de propagación de una grieta, lo que será de utilidad tanto en la fase de propagación como en la fase de iniciación. En el *script iniciacion.py* están las funciones necesarias para obtener los ciclos de iniciación de una grieta. En el *script principal.py* se encuentran las funciones que permiten obtener las curvas de vida a partir de los datos experimentales necesarios. Por último, en el *script graficas.py* se incorpora lo necesario para realizar distintas gráficas que serán de utilidad para comprobar los resultados.

El cálculo de los ciclos de vida de un experimento concreto mediante el modelo aquí implementado se puede dividir en dos fases, como ya se ha explicado anteriormente. El cálculo de los ciclos de iniciación se ha abordado de la siguiente manera. En primer lugar se obtienen para un material concreto las curvas de iniciación mediante el *script iniciacion.py*. Como se ha visto (veáse la Sección 2.1), estas curvas de iniciación solo dependen del parámetro usado y de las propiedades del material, por lo que este cálculo será necesario realizarlo una única vez para cada material y parámetro. Los resultados de las curvas de iniciación se guardarán en un archivo de texto al que habrá que acceder posteriormente para cada experimento. En dicho archivo se tiene una matriz con los ciclos de iniciación en función de la longitud de iniciación y el valor del parámetro.

Una vez obtenidas las curvas de iniciación se pueden calcular las curvas de vida de iniciación, propagación y total para un experimento concreto. Teniéndose el tensor de tensiones y deformaciones a lo largo de una línea perpendicular a la superficie, se puede calcular el valor de los parámetros en cada uno de esos puntos. Si se calcula la media del parámetro desde la superficie hasta una longitud de iniciación, se pueden calcular los ciclos de iniciación interpolando en las curvas de iniciación. Aparte, con el tensor de tensiones se pueden calcular los ciclos de propagación propagando la grieta desde esa longitud de iniciación hasta el fallo utilizando el *script propagacion.py*. Todos estos cálculos se realizan dentro del *script principal.py*, llamando a las funciones requeridas del resto de *scripts*.

En las Figuras 4.1 y 4.2 se muestran diagramas con el funcionamiento del programa. Cada bloque representa un *script* y las flechas indican las llamadas que se realizan entre las distintas funciones. Por un lado, para obtener las curvas de iniciación de un material se ejecuta la función *curvas_iniciacion()* dentro de *iniciacion.py*. Por otro lado, para obtener las curvas de vida y la estimación de la misma se ejecuta la función *principal()* dentro *principal.py*. En las siguientes secciones se comentará el funcionamiento interno de las distintas funciones.

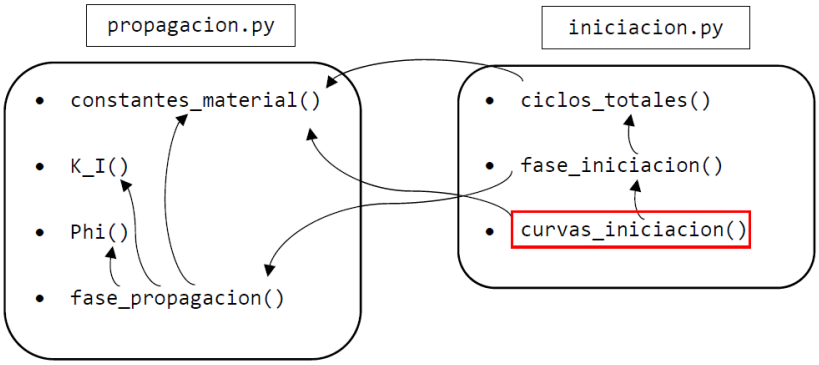


Figura 4.1 Diagrama de ejecución de las curvas de iniciación.

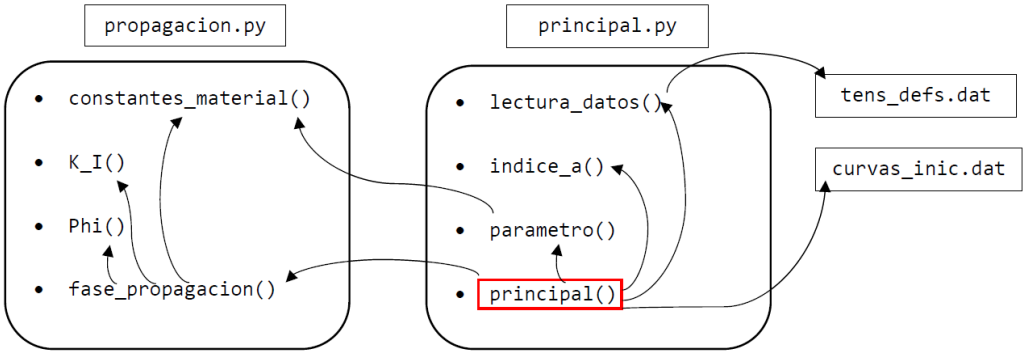


Figura 4.2 Diagrama de ejecución de las curvas de vida a fatiga.

4.1 Descripción del *script propagacion.py*

Como ya se ha mencionado, dentro del *script propagación.py* se encuentran todas las funciones necesarias para realizar la integración de la propagación de una grieta. En la cabecera del mismo se importan los módulos *math* y *numpy*, tal y como se muestra en el Código 4.1, que serán de utilidad a la hora de realizar los diferentes cálculos.

Código 4.1 Módulos importados en *propagacion.py*.

```
import math
import numpy as np
```

En el Código 4.2 se muestra el código de la función *constantes_material()*. En esta función se introducen y calculan todas las propiedades necesarias del material estudiado que se mostraron en la Sección 3.1.

La función admite como entrada la variable "MAT", que identifica al material que se quiere utilizar. Actualmente solo está el Aluminio 7075-T651, que utiliza el índice 0. Si se añadiera un nuevo material utilizaría el valor "MAT" = 1 y así sucesivamente con cada nuevo material añadido. La salida de la función es una lista con las propiedades necesarias del material.

Código 4.2 Código de la función *constantes_material()*.

```
def constantes_material(MAT = 0):
    """Devuelve las propiedades del material.

    INPUT:  MAT        = indice asignado al material
                       = 0 por defecto

    OUTPUTS: mat_props[C = coeficiente de la ley de crecimiento
            n          = exponente de la ley de crecimiento
            f          = parametro en aproximacion al diagrama
            Kitagawa-Takahashi
            l_0        = (m) distancia de la superficie a la primera
                barrera
            microestructural
            K_th       = (MPa m^0.5) umbral de creciemiento de la grieta
            a_0        = (m) parametro de El Haddad
            K_IC       = (MPa m^0.5) tenacidad a fractura
            sigma_y    = (MPa) limite elastico
            sigma_f    = (MPa) coeficiente de resistencia a fatiga
            E          = (MPa) modulo de Young
            nu         = coeficiente de Poisson
            b          = exponente de resistencia a fatiga
            G          = (MPa) modulo de cizalladura]"""

    #Constantes de los materiales
    C       = []
    n       = []
    f       = []
    l_0     = []
    sigma_fl = []
    K_th    = []
    a_0     = []
    K_IC    = []
    sigma_y = []
    sigma_f = []
    E       = []
    nu      = []
    b       = []
    G       = []

    #Material 0: Aluminio 7075-T651
    C.append(8.83e-11)
    n.append(3.322)
    f.append(2.5)
    l_0.append(25e-6)
    K_th.append(2.2)
    sigma_fl.append(169.0)
    K_IC.append(29.0)
    sigma_y.append(503.0)
    sigma_f.append(1610.0)
```

```
    E.append(71000.0)
    nu.append(0.33)
    b.append(-0.1553)
    G.append(E[MAT]/(2.0*(1.0 + nu[MAT])))

    #Material 1:
#    C.append()
#    n.append()
#    f.append()
#    l_0.append()
#    K_th.append()
#    sigma_fl.append()
#    K_IC.append()
#    sigma_y.append()
#    sigma_f.append()
#    E.append()
#    nu.append()
#    b.append()
#    G.append(E[MAT]/(2.0*(1.0 + nu[MAT])))

    #Calculo del parametro de El Haddad y salida de las constantes
    a_0.append(1/math.pi*(K_th[MAT]/(2*sigma_fl[MAT]))**2.0)

    mat_props = [C[MAT], n[MAT], f[MAT], l_0[MAT], K_th[MAT], a_0[MAT],
                 K_IC[MAT], sigma_y[MAT], sigma_f[MAT], E[MAT], nu[MAT],
                 b[MAT], G[MAT]]
    return mat_props
```

En el Código 4.3 se puede ver el código de la función *K_I()*, que se utiliza para calcular el factor de intensidad de tensiones de una grieta determinada. En esta función se han introducido las Ecuaciones 2.5 - 2.8.

Como entrada recibe cuatro variables; “sigma”, la tensión perpendicular al plano de la grieta; “a”, la longitud de la grieta; “ds”, el paso de integración y “W”, la anchura del espécimen. La integral de la Ecuación 2.8 se ha dividido según el cálculo de la misma sea para la fase de iniciación o de propagación. En la fase de iniciación, la tensión será constante; mientras que en la fase de propagación la tensión variará con la profundidad; por lo que la variable “sigma” puede ser de tipo *float* o de tipo *list*. La salida de la función es el valor del factor de intensidad de tensiones “K_I”.

Código 4.3 Código de la función *K_I()*.

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

    INPUTS: sigma = (MPa) tension 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 especimen

    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

    def integr_KI(s, sx):
        """Realiza el cálculo de la integral del FIT"""

        res = sx/s**0.5*(1.0 + m1*s/a + m2*(s/a)**2.0)

        return res

    #Si sigma es de tipo float, el cálculo es para la fase de iniciación
    if type(sigma) is not list:

        integral = 0.0
        s        = ds/2.0
        N        = int(round(a/ds, 8))

        for i in range(N - 1):
            integral += integr_KI(s, sigma)*ds
            s        += ds

        K_I      = (2.0/math.pi)**0.5*integral

    #Si sigma es de tipo list, el cálculo es para la fase de propagación
    else:
        integral = 0.0
        s        = ds/2.0

        for i in sigma[:-1]:
            j = sigma[sigma.index(i)+1]
            integral += integr_KI(s, (i+j)/2.0)*ds
            s        += ds

        K_I = (2.0/math.pi)**0.5*integral

    return K_I
```

En el Código 4.4 se muestra la función *Phi()* que permite calcular el factor Φ para una grieta elíptica, mostrado anteriormente en la Ecuación 2.9. Como entrada solo necesita la variable "ac" que representa la relación de aspecto de la grieta. Como salida devuelve la variable "Phi".

Código 4.4 Código de la función *Phi()*.

```
def Phi(ac = 0.5):
```

```
    """Devuelve el factor Phi calculado por Irwin para el caso de una
       grieta
    elíptica.

    INPUT: ac  = a/c | relacion entre los semiejes

    OUTPUT: Phi = factor de la grieta eliptica"""

    #Calculo del factor para grietas elípticas
    Phi  = 0.0
    N    = 100.0
    dphi = math.pi/2.0/N
    phi  = dphi/2.0

    while(phi <= math.pi/2.0):
        Phi += math.sqrt(1.0 - (1.0 - (ac)**2.0)*(math.sin(phi))**2.0)*
           dphi
        phi += dphi

    return Phi
```

En el Código 4.5 se puede ver la función *fase_propagacion()*, en la que se realiza la integración de la propagación de la grieta y se calcula el número de ciclos de propagación desde una longitud inicial hasta el fallo; utilizándose la Ecuación 2.10.

Admite como entrada seis variables; “sigma”, la tensión máxima perpendicular al plano de la grieta; “ind_a”, índice que asocia la longitud de una grieta con su posición en la lista de longitudes de grieta; “a_i”, longitud inicial de la grieta desde la que se iniciará la propagación; “da”, paso de integración; “W”, anchura del espécimen y “MAT”, índice del material utilizado.

Internamente se ha definido una nueva función en la que se ha incorporado la Ecuación 2.10 con la ley de crecimiento. Esta función presenta una asíntota vertical, por lo que se hace diverger el resultado cuando la grieta no debe propagarse. La integración de dicha función se ha vuelto a dividir según sea la fase de iniciación o de propagación, ya que la variable “sigma” puede volver a ser de tipo *float* o de tipo *list* al igual que ocurría en la función *K_I()*.

Como salida devuelve la variable “N_p” que representa el número de ciclos necesarios para propagar la grieta desde la longitud inicial hasta que se produce el fallo.

Código 4.5 Código de la función *fase_propagacion()*.

```
def fase_propagacion(sigma, ind_a, a_i, da, W, MAT = 0):
    """Devuelve los ciclos de propagacion de la grieta.

    INPUTS: sigma  = (MPa) tensión maxima perpendicular al plano de la
       grieta
                     (float) --> fase de iniciación
                     (list) --> fase de propagación
            ind_a  = indice asociado a la longitud de grieta
            a_i    = (m) longitud inicial de la grieta
            da     = (m) paso de longitudes de grietas
            W      = (m) anchura del especimen
            MAT    = indice asignado al material
```

```
    OUTPUT: N_p    = ciclos de la fase de propagacion"""

    #Obtenemos las constantes del material
    mat_props = constantes_material(MAT)

    C    = mat_props[0]
    n    = mat_props[1]
    f    = mat_props[2]
    l_0  = mat_props[3]
    K_th = mat_props[4]
    a_0  = mat_props[5]
    K_IC = mat_props[6]

    def integr_prop(x, s, ac):
        """Realiza el cálculo de la integral de los ciclos de propagació
            n"""
        phi = Phi(ac)
        ki = K_I(s, x, da, W)/phi

        if ki < K_th*(x**f/(x**f + a_0**f - l_0**f))**(0.5*f):
            res = 1e20
        else:
            res = 1.0/(C*(ki**n
                        - (K_th*(x**f/(x**f + a_0**f - l_0**f))**(0.5*f)
                            )**n))

        return ki, res

    #Si sigma es de tipo float, el cálculo es para la fase de iniciación
    if type(sigma) is not list:
        ac  = 0.5
        N_p = 0.0
        a   = a_i
        ki  = 0.0

        while ki < K_IC:
            N_p += integr_prop(a, sigma, ac)[1]*da
            ki   = integr_prop(a, sigma, ac)[0]
            a   += da

    #Si sigma es de tipo list, el cálculo es para la fase de propagación.

    #Se empieza la integral en la longitud de iniciacion requerida y se
        va
    #aumentando la longitud, utilizando la variable i, de forma que en
        cada
    #vuelta del bucle aumenta en 1 el tamaño del vector de tensiones y
        la
    #longitud de grieta consecuentemente con el paso.
```

```
    else:
        ac  = 0.5
        N_p = 0.0
        i   = 0
        a   = a_i
        ki  = 0.0

        while ki < K_IC:
            #Invertimos los vectores de tensiones, ya que para el calculo
                de
            #K_IC la integral se inicia al fondo de la grieta acabando en
            #la superficie. Seleccionamos del vector completo de
                tensiones, las
            #componentes del mismo que van desde la superficie hasta el
                tamaño
            #de grieta asociado al indice ind_a
            sxx_max = np.flipud(sigma[:ind_a + 1 + i]).tolist()
            N_p    += integr_prop(a, sxx_max, ac)[1]*da
            ki      = integr_prop(a, sxx_max, ac)[0]
            a      += da
            i      += 1

    return N_p
```

4.2 Descripción del *script iniciacion.py*

Dentro del *script iniciacion.py* se encuentran las funciones necesarias para obtener las curvas de iniciación de un material. Este *script* hace uso de algunas funciones del *script propagacion.py*, tal y como se mostró en la Figura 4.1, por lo que en la cabecera del mismo es necesario importarlas tal y como se puede comprobar en el Código 4.6. Además de estas funciones se hace uso del módulo *os* para poder trabajar con las rutas de los archivos, el módulo *matplotlib.pyplot* para realizar gráficas y la función *fsolve()* del módulo *scipy.optimize* para la resolución de ecuaciones.

Código 4.6 Módulos importados en *iniciacion.py*.

```
import os
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
from propagacion import fase_propagacion
from propagacion import constantes_material
```

En el Código 4.7 se muestra la función *ciclos_totales()*, en la que se realiza el cálculo de los ciclos totales hasta el fallo para un criterio concreto. En esta función se hace uso de las Ecuaciones 2.2 y 2.4.

Como entrada recibe tres variables; “param”, el valor del parámetro; “crit”, nombre del criterio utilizado para distinguir la ecuación que hay que utilizar y “MAT”, índice del material utilizado. La variable “crit” es de tipo *string* y puede recibir dos valores distintos, ‘FS’ o ‘SWT’, según se requiera. Para realizar los cálculos se hace uso de la función *fsolve()*, importada en la cabecera del

script. Como salida devuelve la variable "N_t", que representa el número de ciclos totales hasta el fallo.

Código 4.7 Código de la función *ciclos_totales()*.

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

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

    OUTPUT: N_t  = ciclos totales"""

    #Constantes del material necesarias
    sigma_y = constantes_material(MAT)[7]
    sigma_f = constantes_material(MAT)[8]
    k       = sigma_y/sigma_f
    E       = constantes_material(MAT)[9]
    nu      = constantes_material(MAT)[10]
    b       = constantes_material(MAT)[11]

    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):
        """Funcion que utiliza el fsolve para calcular los ciclos de
           vida
        cuando se utiliza el Smith-Watson-Topper."""
        expr = param - (sigma_f**2.0/E*(2*x)**(2.0*b))

        return expr

    if crit == 'FS':
        N_t = fsolve(func=fsolve_FS, x0=10, xtol=1e-10)[0]

    elif crit == 'SWT':
        N_t = fsolve(func=fsolve_SWT, x0=100, xtol=1e-10)[0]

    return N_t
```

En el Código 4.8 se muestra la función *fase_iniciacion()*, en la que se calculan los ciclos de iniciación para una longitud de iniciación concreta. En esta función se corrigen las curvas de vida

totales de un parámetro para una longitud concreta.

Recibe como entrada siete variables; “param”, valor del parámetro en el que se quiere realizar la corrección; “sigma”, tensión perpendicular al plano de la grieta correspondiente al valor de la variable “param”; “crit”, nombre del criterio utilizado; “a_i”, longitud de iniciación; “da”, paso de integración; “W”, anchura del espécimen y “MAT”, índice del material asignado. La variable “crit” vuelve a recibir dos posibles valores distintos, ‘FS’ o ‘SWT’.

Dentro de esta función, por un lado se llama a la función *ciclos_totales()* y por otro a la función *fase_propagación()* importada en la cabecera. De esta forma se puede corregir lo obtenido por la primera función con la segunda. Como salida devuelve la variable “N_i” que representa los ciclos necesarios para iniciar la grieta hasta la longitud de iniciación. Debido a discrepancias entre el modelo utilizado para la propagación y los criterios de fatiga multiaxiales empleados, se pueden obtener ciclos de iniciación negativos. Para ello se introducen dos correcciones distintas de la variable “N_i”. En primer lugar, si el resultado es negativo se iguala a 0. En segundo lugar, para ciclos totales muy elevados se asume que la propagación es despreciable frente a la iniciación.

Código 4.8 Código de la función *fase_iniciacion()*.

```
def fase_iniciacion(param, sigma, crit, a_i, da, W, MAT):
    """Devuelve los ciclos de la fase de iniciacion de una grieta,
       conocida
    la tension media desde la superficie hasta la punta de la misma.

    INPUTS: param = Fatemi-Socie o Smith-Watson-Topper en un punto
            sigma = tension x en ese punto
            crit  = parametro a utilizar
            a_i   = (m) tamaño de la grieta de iniciacion
            da    = paso para realizar los calculos
            W     = (m) anchura del especimen
            MAT   = indice asignado del material

    OUTPUT: N_i  = ciclos de la fase de iniciacion"""

    #Calcula los ciclos totales hasta el fallo
    N_t = ciclos_totales(param, crit, MAT)

    #ind_a se utiliza para la propia fase de propagacion por lo
    #que en la fase de iniciacion no es una variable relevante y puede
       tomar
    #cualquier valor
    ind_a = 0

    #Calculos los ciclos que necesita la grieta para propagarse
    N_p = fase_propagacion(sigma, ind_a, a_i, da, W, MAT)

    #Calculamos los ciclos de iniciacion
    N_i = N_t - N_p

    #Si el numero es negativo devuelve 0
    if N_i < 0:
        N_i = 0.0
```

```
    #Para ciclos muy altos solo se tiene en cuenta la iniciación para
        evitar
    #errores que se producen en la integración de la propagación
    if N_t > 1.5e7:
        N_i = N_t

    return N_i
```

En el Código 4.9 se puede ver la función *curvas_iniciación()*, en el que se coordina la realización de las curvas de iniciación. Como entrada recibe cuatro variables; "par", nombre del criterio utilizado; "da", paso de integración; "W", anchura del espécimen y "MAT", índice del material utilizado. La variable "par" nuevamente puede tomar el valor 'FS' o 'SWT'. Como salida se obtiene un archivo de texto en formato *.dat* con las curvas de iniciación.

Dentro de la función, en primer lugar, se elige el rango de tensiones para el que se quieren calcular las curvas de iniciación; así como el número de discretizaciones de las mismas. Posteriormente se calcula, para cada valor de tensión, el valor del parámetro correspondiente utilizando las Ecuaciones 2.1 y 2.3 según corresponda. Luego se elige el número de curvas a utilizar, así como la separación entre las mismas. Una vez definidas todas estas variables se puede proceder a realizar el cálculo de las curvas de iniciación y escribir el archivo de texto, utilizando para ello la función *fase_iniciacion()*. Por último, se genera una figura con las curvas.

Código 4.9 Código de la función *curvas_iniciacion()*.

```
def curvas_iniciacion(par, da, W, MAT):
    """Escribe un archivo de texto con las curvas de
    iniciación para distintas longitudes de grieta para un material.

    INPUT:  par   = parametro para el modelo de iniciacion
            da    = paso para realizar los calculos
            W     = (m) anchura del especimen
            MAT   = indice asignado al material

    OUTPUTS: MATX_par.dat = archivo con las curvas de iniciacion
            figura.png  = imagen con las curvas de iniciacion"""

    print ('Curvas de iniciacion del material {} '
          +'utilizando el parametro {}\n').format(MAT, par)

    #Abrimos el archivo donde se van a escribir los datos,
    #escribimos la cabecera del mismo y creamos los vectores con las
        tensiones
    #y los tamaños de grieta para crear las curvas de iniciación
    cwd     = os.getcwd()
    ruta    = cwd + '/curvas_inic'
    ruta_fig = cwd + '/grafs/' + par
    archivo = open('{}/MAT{}_{}.dat'.format(ruta, MAT, par), 'w')

    archivo.write('0    ')
```

```
v_sigma    = []      #Vector de tensiones
v_param    = []      #Vector de Fatemi-Socie o Smith-Watson-Topper
n_sigma    = 45      #Discretizaciones de la curva de iniciacion

sigma_max  = 500.0   #(MPa) tension maxima
sigma_min  = 50.0    #(MPa) tension minima

delta_sigma = (sigma_max - sigma_min)/n_sigma #Paso de tensiones

#Cargamos propiedades del material
sigma_y = constantes_material(MAT)[7]
sigma_f = constantes_material(MAT)[8]
k       = sigma_y/sigma_f
E       = constantes_material(MAT)[9]
G       = constantes_material(MAT)[12]

#Generamos el vector de Fatemi-Socie o Smith-Watson-Topper
for i in range(n_sigma+1):
    sigma_i = sigma_min+i*delta_sigma
    def_i  = sigma_i/E
    gamma_i = sigma_i/2.0/G
    v_sigma.append(sigma_i)
    if par == 'FS':
        v_param.append(gamma_i*(1.0+k*sigma_i/2.0/sigma_y))
    elif par == 'SWT':
        v_param.append(sigma_i*def_i)

m_N_i = [[], []]     #Matriz para guardar ciclos y tensiones para
                     #facilitar la generacion de las graficas
v_a   = []           #Vector de tamaños de grietas
n_a   = 120          #Número de curvas de iniciacion por material
a_min = 5e-6         #Tamaño más pequeño grieta
ex    = 1.45      #Variable para controlar como crece la diferencia
            #entre longitudes de grieta

for i in range(n_a):
    v_a.append(a_min*(i+1.0)**ex)
    archivo.write('{:.2e} '.format(v_a[-1]))

#Creamos las curvas de iniciación en el archivo
for i in range(len(v_param)):
    archivo.write('\n{:+.2e}'.format(v_param[i]))

    for j in v_a:
        N_i  = fase_iniciacion(v_param[i], v_sigma[i], par, j, da, W,
            MAT)
        m_N_i[0].append(N_i)
        m_N_i[1].append(v_sigma[i])

        archivo.write(' {:+.6e}'.format(N_i))
```

```
        #Pintamos en la consola el porcentaje realizado
        print '\r{:.2%} completado'.format((i+1.0)/len(v_param)),

    #Cerramos el archivo
    archivo.close()

    v_N_i = []
    v_sigma2 = []

    #Utilizando la matriz auxiliar m_N_i, creamos los vectores de ciclos
         de
    #iniciacion y tensiones en el formato adecuado para dibujar las
        curvas.
    for i in range(n_a):
        v_N_i.append([])
        v_sigma2.append([])

    for i in range(n_sigma):
        for j in range(n_a):
            v_N_i[j].append(m_N_i[0][j+i*n_a])
            v_sigma2[j].append(m_N_i[1][j+i*n_a])

    #Pintamos las curvas de iniciación
    plt.close('MAT_' + str(MAT) + '|crit_' + par)
    plt.figure('MAT_' + str(MAT) + '|crit_' + par)
    for i in range(n_a):
        plt.plot(v_N_i[i], v_sigma2[i])

    plt.xlabel('Ciclos de iniciacion')
    plt.xscale('log')
    plt.show()

    #Guardamos la figura y la cerramos
    plt.savefig(ruta_fig + '/curvas_inic.png')
    plt.close('MAT_' + str(MAT) + '|crit_' + par)
```

4.3 Descripción del *script principal.py*

En el *script principal.py* se coordina la generación de las curvas de vida de iniciación, de propagación y total, a partir de las cuales se puede obtener la vida a fatiga estimada, la longitud de iniciación de la grieta, la duración de las fases de iniciación y propagación o la evolución de la longitud de la grieta. En la cabecera, mostrada en el Código 4.10 se importan varios módulos ya mencionados anteriormente; *os, math, numpy, matplotlib.pyplot* y las funciones del *script propagacion.py fase_propagacion() y constantes_material()*. Además de estas, se importan las funciones *minimize()* e *interp2d* del módulo *scipy*, que permiten optimizar una función e interpolar en una matriz de datos respectivamente.

Código 4.10 Módulos importados en *principal.py*.

```
import os
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.interpolate import interp2d
from propagacion import fase_propagacion
from propagacion import constantes_material
```

La función *lectura_datos()* del Código 4.11 permite cargar los datos experimentales con el tensor de tensiones y el tensor de deformaciones en cada punto. Recibe tres variables como entrada; "ruta", la ruta en la que se encuentran los archivos; "exp_max", el nombre del archivo con tensiones y deformaciones máximas y "exp_min", el nombre del archivo con las tensiones y deformaciones mínimas. La salida de la función devuelve una lista con las distancias a la superficie a la que se tienen el resto de datos, las tensiones máximas perpendiculares al plano de la grieta, el tensor de tensiones máximas, el tensor de deformaciones máximas y el tensor de deformaciones mínimas.

Código 4.11 Código de la función *lectura_datos()*.

```
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"""

    #Cargamos los datos de distancias. Cambiamos el vector para
    #que sea positivo y empiece en 0
    x = np.loadtxt("{}/{}.dat".format(ruta, exp_max), skiprows = 1,
                  usecols = (1,))*-1e-3
    x = x.tolist()

    for i in x:
        x[x.index(i)] = i - 0.1

    #Cargamos el resto de datos
    sxx_max = np.loadtxt("{}/{}.dat".format(ruta, exp_max), skiprows = 1,

                       usecols = (2,)).tolist()

    s_max = np.loadtxt("{}/{}.dat".format(ruta, exp_max), skiprows = 1,
                     usecols = (2, 3, 4, 5, 6, 7)).tolist()
```

```
    e_max = np.loadtxt("{}/{}.dat".format(ruta, exp_max), skiprows = 1,
                    usecols = (8, 9, 10, 11, 12, 13)).tolist()

    e_min = np.loadtxt("{}/{}.dat".format(ruta, exp_min), skiprows = 1,
                    usecols = (8, 9, 10, 11, 12, 13)).tolist()

    return x, sxx_max, s_max, e_max, e_min
```

En el Código 4.12 se muestra la función *indice_a()*. Esta función relaciona una distancia con su índice correspondiente en el vector de distancias a la superficie. Como entrada recibe "a", la distancia y "x", el vector con distancias a la superficie. La salida de la función "ind_a" es el índice asociado.

Código 4.12 Código de la función *indice_a()*.

```
def indice_a(a, x):
    """Devuelve el indice asociado a una longitud de grieta.

    INPUT:  a     = longitud de grieta de iniciacion
            x     = vector de distancias a la superficie

    OUTPUTS: ind_a = indice asociado a la longitud de grieta de
        iniciacion"""

    #Redondeamos a la 8 cifra para evitar errores numéricos
    a = round(a, 8)

    #Calculamos el índice para el cual tenemos esa longitud de grieta
    for i in x:
        i_round = round(i, 8)
        if i_round >= a:
            ind_a = x.index(i)
            break

    return ind_a
```

La función *parametro()* del Código 4.13 calcula en cada punto el parámetro a partir del tensor de tensiones y deformaciones. Como entrada recibe seis variables; "par", el nombre del criterio utilizado; "MAT", el índice del material utilizado; "x", el vector con distancias a la superficie; "s_max", el tensor de tensiones máximas en cada punto; "e_max", el tensor de deformaciones máximas en cada punto y "e_min", el tensor de tensiones mínimas en cada punto. La salida de la función es una lista con el Fatemi-Socie o el Smith-Watson-Topper en cada punto.

Internamente se utilizan dos funciones distintas, mediante las cuales se calcula la variable a optimizar en cada parámetro. En el caso de Fatemi-Socie es el incremento de deformaciones tangenciales, mientras que en el caso de Smith-Watson-Topper es el propio parámetro tal y como se puede ver en la definición de los mismos en las Ecuaciones 2.1 y 2.3. Tras definir estas dos funciones se recorre el vector de distancias a la superficie para calcular en cada punto el parámetro asociado. Para ello, se optimiza en cada punto la función interna descrita anteriormente.

Código 4.13 Código de la función *parametro()*.

```
def parametro(par, MAT, x, s_max, e_max, e_min):
    """Calcula el vector para el modelo de iniciacion asociado
    a un experimento.

    INPUT:  par   = parametro para el modelo de iniciacion
            MAT   = indice asignado al material
            x     = (m) vector de distancias a la superficie de la
                grieta
            s_max = (MPa) vector de tensiones máximas
            e_max = vector de deformaciones máximas
            e_min = vector de deformaciones minimas

    OUTPUTS: FS/SWT = vector con los Fatemi-Socie o Smith-Watson-Topper
        en cada
             punto"""

    def func_FS(alfa, j):
        """Devuelve delta_gamma_max/2 en un punto concreto. Se utiliza
            en el
        calculo del Fatemi-Socie."""
        R_x   = np.array([[1.0, 0.0, 0.0],
                          [0.0, math.cos(alfa[0]), -math.sin(alfa[0])],
                          [0.0, math.sin(alfa[0]), math.cos(alfa[0])]])
        R_x_T = np.transpose(R_x)
        R_y   = np.array([[math.cos(alfa[1]), 0.0, -math.sin(alfa[1])],
                          [0.0, 1.0, 0.0],
                          [math.sin(alfa[1]), 0.0, math.cos(alfa[1])]])
        R_y_T = np.transpose(R_y)
        R_z   = np.array([[math.cos(alfa[2]), -math.sin(alfa[2]), 0.0],
                          [math.sin(alfa[2]), math.cos(alfa[2]), 0.0],
                          [0.0, 0.0, 1.0]])
        R_z_T = np.transpose(R_z)

        E_xyz_max = np.array([[e_max[j][0], e_max[j][3], e_max[j][4]],
                              [e_max[j][3], e_max[j][1], e_max[j][5]],
                              [e_max[j][4], e_max[j][5], e_max[j][2]]])

        E_xyz_min = np.array([[e_min[j][0], e_min[j][3], e_min[j][4]],
                              [e_min[j][3], e_min[j][1], e_min[j][5]],
                              [e_min[j][4], e_min[j][5], e_min[j][2]]])

        E_max = np.dot(np.dot(np.dot(np.dot(R_x_T, np.dot(R_y_T, np.dot(
            R_z_T,
                       E_xyz_max))), R_z), R_y), R_x).tolist()
        E_min = np.dot(np.dot(np.dot(np.dot(R_x_T, np.dot(R_y_T, np.dot(
            R_z_T,
                       E_xyz_min))), R_z), R_y), R_x).tolist()

        #El signo negativo se debe a que la función minimiza en vez de
```

```
    #maximizar.
    return -(E_max[0][1] - E_min[0][1])

def func_SWT(alfa, j):
    """Devuelve el Smith-Watson-Topper asociado a un punto."""
    R_x   = np.array([[1.0, 0.0, 0.0],
                      [0.0, math.cos(alfa[0]), -math.sin(alfa[0])],
                      [0.0, math.sin(alfa[0]), math.cos(alfa[0])]])
    R_x_T = np.transpose(R_x)
    R_y   = np.array([[math.cos(alfa[1]), 0.0, -math.sin(alfa[1])],
                      [0.0, 1.0, 0.0],
                      [math.sin(alfa[1]), 0.0, math.cos(alfa[1])]])
    R_y_T = np.transpose(R_y)
    R_z   = np.array([[math.cos(alfa[2]), -math.sin(alfa[2]), 0.0],
                      [math.sin(alfa[2]), math.cos(alfa[2]), 0.0],
                      [0.0, 0.0, 1.0]])
    R_z_T = np.transpose(R_z)

    E_xyz_max = np.array([[e_max[j][0], e_max[j][3], e_max[j][4]],
                          [e_max[j][3], e_max[j][1], e_max[j][5]],
                          [e_max[j][4], e_max[j][5], e_max[j][2]]])

    E_xyz_min = np.array([[e_min[j][0], e_min[j][3], e_min[j][4]],
                          [e_min[j][3], e_min[j][1], e_min[j][5]],
                          [e_min[j][4], e_min[j][5], e_min[j][2]]])

    S_xyz_max = np.array([[s_max[j][0], s_max[j][3], s_max[j][4]],
                          [s_max[j][3], s_max[j][1], s_max[j][5]],
                          [s_max[j][4], s_max[j][5], s_max[j][2]]])

    E_max = np.dot(np.dot(np.dot(np.dot(R_x_T, np.dot(R_y_T, np.dot(
        R_z_T,
                    E_xyz_max))), R_z), R_y), R_x).tolist()
    E_min = np.dot(np.dot(np.dot(np.dot(R_x_T, np.dot(R_y_T, np.dot(
        R_z_T,
                    E_xyz_min))), R_z), R_y), R_x).tolist()
    S_max = np.dot(np.dot(np.dot(np.dot(R_x_T, np.dot(R_y_T, np.dot(
        R_z_T,
                    S_xyz_max))), R_z), R_y), R_x).tolist()

    #El signo negativo se debe a que la función minimiza en vez de
    #maximizar.
    return -(S_max[0][0]*(E_max[0][0]-E_min[0][0])/2.0)

#Inicializamos variables
alfa0 = [0.0, 0.0, 0.0] #Ángulo para primera iteracion de la funcion
     de
                        #minimizacion
#Limites para el angulo
```

```
    bnds = ((-math.pi, math.pi), (-math.pi, math.pi), (-math.pi, math.pi
        ))

    sigma_y = constantes_material(MAT)[7]
    sigma_f = constantes_material(MAT)[8]
    k       = sigma_y/sigma_f

    alfa           = []
    delta_gamma_max = []
    s_norm         = []
    FS             = []
    SWT            = []

    #Calculamos en cada punto el Fatemi-Socie o el Smith-Watson-Topper
        asociado
    for j in range(len(x)):
        if par == 'FS':
            fs = minimize(func_FS, alfa0, bounds = bnds, args=(j),
                          options={'disp': False})
            alfa.append(fs.x.tolist())
            delta_gamma_max.append(-func_FS(alfa[j], j)*2.0)

            R_x   = np.array([[1.0, 0.0, 0.0],
                              [0.0, math.cos(alfa[j][0]), -math.sin(alfa[j
                                  ][0])],
                              [0.0, math.sin(alfa[j][0]), math.cos(alfa[j
                                  ][0])]])
            R_x_T = np.transpose(R_x)
            R_y   = np.array([[math.cos(alfa[j][1]), 0.0, -math.sin(alfa[
                j][1])],
                              [0.0, 1.0, 0.0],
                              [math.sin(alfa[j][1]), 0.0, math.cos(alfa[j
                                  ][1])]])
            R_y_T = np.transpose(R_y)
            R_z   = np.array([[math.cos(alfa[j][2]), -math.sin(alfa[j][2])
                , 0.0],
                              [math.sin(alfa[j][2]), math.cos(alfa[j][2]),
                                  0.0],
                              [0.0, 0.0, 1.0]])
            R_z_T = np.transpose(R_z)

            S_xyz_max = np.array([[s_max[j][0], s_max[j][3], s_max[j
                ][4]],
                                  [s_max[j][3], s_max[j][1], s_max[j][5]],
                                  [s_max[j][4], s_max[j][5], s_max[j][2]]])

            S_max = np.dot(np.dot(np.dot(np.dot(R_x_T, np.dot(R_y_T,
                          np.dot(R_z_T, S_xyz_max))), R_z), R_y), R_x).
                              tolist()
            s_norm.append(S_max[2][2])
```

```
            FS.append(delta_gamma_max[j]/2.0*(1.0 + k*s_norm[j]/sigma_y))

        elif par == 'SWT':
            swt  = minimize(func_SWT, alfa0, bounds = bnds, args=(j),
                            options={'disp': False})
            alfa.append(swt.x.tolist())
            SWT.append(-func_SWT(alfa[j], j))

    if par == 'FS':
        return FS

    elif par == 'SWT':
        return SWT
```

La función *principal()* mostrada en el Código 4.14 es la función desde la cual se coordina el resto de funciones. Como entrada recibe cinco variables; "par", el nombre del criterio utilizado; "W", la anchura del espécimen; "MAT", el índice asignado al material; "exp_max", el nombre del archivo con las tensiones y deformaciones máximas y "exp_min", el nombre del archivo con las tensiones y deformaciones mínimas.

Dentro de esta función, tras declarar las rutas de las distintas carpetas, se genera la función de interpolación para calcular los ciclos de la fase de iniciación. Después, se cargan los datos experimentales mediante la función *lectura_datos()* y se genera el vector del parámetro seleccionado utilizando la función *parametro()*. Tras inicializar variables, se recorre un vector de longitudes de iniciación calculándose en cada punto los ciclos de la fase de iniciación para iniciar la grieta hasta dicha longitud de iniciación y los ciclos de la fase de propagación para propagar la grieta desde la longitud de iniciación hasta el fallo. Los ciclos de vida totales son las suma de ambas fases.

Por otro lado, tras obtener las curvas de vida se genera la curva con la evolución de la longitud de la grieta en función del número de ciclos. La obtención de esta curva se divide en dos partes. Hasta la longitud de iniciación, la grieta crece tal y como lo hace el número de ciclos de iniciación. A partir de esta longitud de grieta, la grieta crece según los ciclos de propagación de forma que para dos longitudes sucesivas en esta fase, el incremento en el número de ciclos de la curva es igual a la diferencia entre los ciclos de propagación de ambas longitudes.

Como salida esta función genera archivos de texto con las curvas de vida a fatiga, la evolución de la longitud de la grieta y datos sobre el punto en el que se produce el mínimo de la curva de vida total, que se corresponde con la vida de la pieza.

Código 4.14 Código de la función *principal()*.

```
def principal(par, W, MAT, exp_max, exp_min):
    """Estima la vida a fatiga.

    INPUTS: par    = parametro para el modelo de iniciacion
            W      = (m) anchura del especimen
            MAT    = indice asignado al material
            exp_max = nombre del archivo con la tensiones y defs maximas
            exp_min = nombre del archivo con la tensiones y defs minimas

    OUTPUTS: resultados.dat = actualiza el archivo de resultados con la
```

```
        longitud de iniciacion y los ciclos de iniciacion,
            propagacion y
        total para que se produzca el fallo
        exp_id.dat    = archivo con los datos de las curvas de vida
            """

print 'Datos Experimentales:\n {}.dat\n  {}.dat\n'.format(exp_max,
                                                        exp_min)
#exp_id obtiene a partir del nombre del archivo con los datos un
#identificador del experimento. Dependiendo del nombre del archivo
    debe
#modificarse.
exp_id     = exp_max[16:]

#Obtenemos las rutas a las carpetas necesarias para los calculos
cwd        = os.getcwd()
ruta_exp   = cwd + '/datos_exp'
ruta_curvas = cwd + '/curvas_inic2'
ruta_fig   = cwd + '/resultados/grafs/' + par
ruta_datos = cwd + '/resultados/datos/' + par

#Cargamos los datos de las curvas de iniciación del material
data_interp = np.loadtxt("{}/MAT{}_{}.dat".format(ruta_curvas, MAT,
    par))

#Separamos las curvas en las variables necesarias
x_interp = data_interp[0][1:] #Eje x de la matriz de interpolación
y_interp = []        #Eje y de la matriz de interpolación
m_N_i   = []         #Matriz con los ciclos de iniciación

for i in data_interp[1:]:
    y_interp.append(i[0])
    m_N_i.append(i[1:])

y_interp = np.array(y_interp)

#Creamos la función de interpolación
function_interp = interp2d(x_interp, y_interp, m_N_i, kind='quintic',

                           bounds_error=False)

#Cargamos los datos experimentales y generamos el vector de FS o SWT
x, sxx_max, s_max, e_max, sxx_min, e_min = lectura_datos(ruta_exp,
    exp_max,
                                                    exp_min)
param = parametro(par, MAT, x, s_max, e_max, e_min)

#Iniciamos variables
v_ai   = []          #Vector de longitudes de grieta en m
v_ai_mm = []         #Vector de longitudes de grieta en mm
```

```
N_i    = []          #Vector de ciclos de iniciación
N_p    = []          #Vector de ciclos de propagación
N_t    = []          #Vector de ciclos totales

a_i_min = round(x[1], 8) #Tamaño mínimo de longitud de grieta de
                      #iniciacion. Redondeamos para evitar errores
                      #numericos.
a_i_max = round(x[-2], 8) #Tamaño máximo de longitud de grieta de
    iniciacion
da     = a_i_min     #Paso entre longitudes de grietas
N      = int((a_i_max - a_i_min)/da)+1 #Número de grietas a calcular

#Creamos el vector de longitudes de grieta de iniciacion
a_i    = a_i_min

for i in range(N):
    v_ai.append(a_i + i*da)

#Calculamos los ciclos de iniciación, de propagación y totales para
    cada
#longitud de grieta de iniciacion
for i in v_ai:
    ind_a    = indice_a(i, x) #Indice asociado a esa longitud de
        grieta
    param_med = np.mean(param[:ind_a + 1]) #Valor medio del
        parametro para
                    #realizar la interpolacion
    #Realizamos la interpolacion para calcular los ciclos de
        iniciacion
    n_i = function_interp(i, param_med)[0]

    #La interpolacion puede dar valores menores que 0 para ciclos
        muy bajos
    if n_i < 0:
        n_i = 0

    #Calculamos los ciclos de propagacion
    n_p = fase_propagacion(sxx_max, sxx_min, ind_a, i, da, W, MAT)

    #Se añaden los datos calculados a las curvas
    N_i.append(n_i)
    N_p.append(n_p)
    N_t.append(n_i+n_p)
    v_ai_mm.append(i*1e3)

#Pintamos los curvas de iniciación, de propagación y totales
plt.close(exp_id + '_' + par)
plt.figure(exp_id + '_' + par)
plt.yscale('log')
plt.xlabel('Longitud de iniciacion (mm)')
```

```
    plt.ylabel('Ciclos')
    plot_inic = []
    plot_prop = []
    plot_tot  = []
    plot_inic += plt.plot(v_ai_mm, N_i, 'b')
    plot_prop += plt.plot(v_ai_mm, N_p, 'k')
    plot_tot += plt.plot(v_ai_mm, N_t, 'r')

    #Calculamos el numero de ciclos hasta el fallo y la longitud de
        iniciación
    #de la grieta, que se producen en el mínimo de la curva de ciclos
        totales
    N_t_min  = min(N_t)
    i_N_t_min = N_t.index(N_t_min)
    N_i_min  = N_i[i_N_t_min]
    N_p_min  = N_p[i_N_t_min]
    a_inic   = v_ai_mm[i_N_t_min]

    #Pintamos el punto donde se da el minimo
    plt.plot(a_inic, N_t_min, 'ro')
    plt.ylim([1e3,1e8])
    plt.xlim([0.0,a_i_max*1e3])
    plt.legend([plot_inic[0], plot_prop[0], plot_tot[0]],
              ['Vida de iniciacion', 'Vida de propagacion', 'Vida total'
                  ],
              loc = 0)
    plt.show()

    #Guardamos la figura y la cerramos
    plt.savefig(ruta_fig + '/{}.png'.format(exp_id))
    plt.close(exp_id + '_' + par)

    #Pintamos la figura con la evolucion de la longitud de grieta y
        guardamos
    #en un archivo los datos
    ciclos = open(ruta_datos + '/{}.dat'.format(exp_id), 'w')
    ciclos.write('{:<5}\t{:<12}\t{:<12}\t{:<12}\t{:<12}'.format('a_i', '
        N_t',
                                                            'N_i', 'N_p',
                                                            'N_a'))

    N_a = []             #Vector de ciclos con la evolución de la grieta

    for i in v_ai:
        #Hasta la longitud de iniciacion crece como los ciclos de
            iniciacion
        if i <= a_inic*1e-3:
            n_a = N_i[v_ai.index(i)]
            N_a.append(n_a)
```

```
        #A partir de la longitud de iniciacion crece de acuerdo con los
            ciclos
        #de propagacion
        else:
            n_a = N_a[-1] + N_p[v_ai.index(i)-1] - N_p[v_ai.index(i)]
            N_a.append(n_a)
        n_i = N_i[v_ai.index(i)]
        n_p = N_p[v_ai.index(i)]

        ciclos.write('\n{:.3f}\t{:.6e}\t{:.6e}\t{:.6e}\t{:.6e}'.format(i
            *1e3,
                    n_i+n_p, n_i, n_p, n_a))
    ciclos.close()

    plt.figure('Longitud de grieta')
    plt.xscale('log')
    plt.xlabel('Ciclos')
    plt.ylabel('Longitud de grieta (mm)')
    plt.plot(N_a, v_ai_mm, 'k')
    plt.xlim([5e2,1e8])
    plt.show()

    #Generamos/actualizamos el archivo con los resultados para la
        estimacion
    lines = np.loadtxt('resultados.dat', dtype = str, skiprows = 1).
        tolist()

    results = open('resultados.dat', 'w')
    results.write('{:<13}\t{:<}\t{:<12}\t{:<12}\t{:<12}\t{:<5}\t{:<5}\t
        {:<}'.format('exp_id',
                 'param', 'N_t_min', 'N_i_min', 'N_p_min', '% N_i', '%
                    N_p', 'a_inic (mm)'))

    #Se reescriben las lineas que ya estaban en el archivo. EL if else
        es debido
    #a que el formato de lines varía según haya una línea de resultados
        escrita
    #o mas de una.
    if len(lines[0][0]) == 1:
        #Solo se escriben los resultados que no pertenezcan al calculo
            actual
        if lines[0] != exp_id or lines[1] != par:
            results.write('\n')
            for j in i:
                results.write('{}\t'.format(j))
    else:
        for i in lines:
            #Solo se escriben los resultados que no pertenezcan al
                calculo actual
            if i[0] != exp_id or i[1] != par:
```

```
                results.write('\n')
                for j in i:
                    results.write('{}\t'.format(j))

    #Se escribe en el archivo el calculo actual
    results.write('\n{}\t{}\t{:.6e}\t{:.6e}\t{:.6e}\t{:.1%}\t{:.1%}\t
        {:.3f}'.format(exp_id, par,
                  N_t_min, N_i_min, N_p_min, float(N_i_min)/N_t_min,
                  float(N_p_min)/N_t_min, a_inic))
    results.close()

    print 'Longitud de iniciación de la grieta: {} mm'.format(a_inic)
    print 'Numero de ciclos hasta el fallo: {}\n'.format(N_t_min)
```

4.4 Descripción del *script graficas.py*

El *script graficas.py* se utiliza para generar gráficas a partir de los resultados obtenidos con los *scripts* descritos anteriormente además de para obtener parámetro para poder comparar cuantitativamente los resultados. Hay funciones tanto para generar nuevas gráficas que no se habían obtenido anteriormente, como funciones para volver a generar las ya existentes en caso de que se quiera modificar algún aspecto de las mismas. El Código 4.15 muestra los módulos importados en el *script*. La única novedad es la importación de la función *LinearRegression*, que pertenece al módulo *sklearn.linear_model*.

Código 4.15 Módulos importados en *graficas.py*.

```
import os
import math
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

En el Código 4.16 se muestra la función *grafs_globales()*, en la que se realizan tres gráficas distintas. Recibe como entrada dos variables. La primera, "vida_exp", es un diccionario con los resultados experimentales de un grupo de ensayos. La segunda, "crit" es el nombre del criterio de fatiga utilizado.

La primera gráfica que se obtiene muestra la vida estimada frente a la vida experimental. La segunda muestra el porcentaje de iniciación frente a la vida estimada. Por último, la tercera muestra la longitud de iniciación frente a la vida estimada. Se utiliza una gráfica 0 para ayudar a poner la leyenda en las gráficas.

Código 4.16 Código de la función *grafs_globales()*.

```
def grafs_globales(vida_exp, crit):
    """Funcion para generar las graficas de vida estimada frente a vida
    experimental, porcentaje de iniciacion y longitud de iniciacion."""

    sin_SP = []
    SP     = []
```

```
    plt.figure(0)
    sin_SP += plt.plot([1], [1], 'bo')
    SP    += plt.plot([1], [1], 'ro')
    plt.close(0)

    #Cargamos los resultamos de las simulaciones
    vida_est = {}
    inic     = {}
    long_inic = {}

    f = np.loadtxt('resultados.dat', dtype = str, skiprows = 1)

    for i in f:
        exp_id = i[0]
        par = i[1]
        if par == crit:
            vida_est[exp_id] = float(i[2])
            inic[exp_id]    = float(i[5][:-1])
            long_inic[exp_id] = float(i[7])

    #Vida estimada frente a vida experimental
    plt.figure(1)
    pts = []

    for i in vida_exp:
        try:
            pts += plt.plot(vida_exp[i], [vida_est[i], vida_est[i]], 'bo'
                )
        except KeyError:
            pass

    plt.plot([0, 1e10], [0, 1e10], 'k')
    plt.plot([0, 1e10], [0, 5e9], 'k')
    plt.plot([0, 1e10], [0, 2e10], 'k')
    plt.yscale('log')
    plt.xscale('log')
    plt.ylim([1e3,1e9])
    plt.xlim([1e3,1e9])
    plt.xlabel('Vida experimental')
    plt.ylabel('Vida estimada')
    plt.grid(which = 'major', color = 'k', linestyle='-', linewidth=0.4)
    plt.grid(which = 'minor', color = 'gray', linestyle=':', linewidth
        =0.2)
#    plt.legend([sin_SP[0], SP[0], pts_SWT[0]],
#               ['Sin tratamiento', 'Shot peening', 'Electroerosion'],
#               numpoints = 1, loc = 0)
    plt.show()

    #Vida estimada frente a porcentaje de iniciacion
    plt.figure(2)
```

```
    pts_inic = []

    for i in vida_exp:
        try:
            pts_inic += plt.plot(vida_est[i], inic[i], 'bo')
        except KeyError:
            pass

    plt.xscale('log')
    plt.ylim([0,102])
    plt.xlim([1e3,1e8])
    plt.xlabel('Vida estimada')
    plt.ylabel('% inic')
    plt.grid(which = 'major', color = 'k', linestyle='-', linewidth=0.4)
    plt.grid(which = 'minor', color = 'gray', linestyle=':', linewidth
        =0.2)
#    plt.legend([sin_SP[0], SP[0], pts_inic_SWT[0]],
#               ['Sin tratamiento', 'Shot peening', 'Electroerosion'],
#               numpoints = 1, loc = 4)
    plt.show()

    #Vida estimada frente a longitud de iniciacion
    plt.figure(3)

    pts_long = []

    for i in vida_exp:
        try:
            pts_long += plt.plot(vida_est[i], long_inic[i], 'bo')
        except KeyError:
            pass

    plt.xscale('log')
    plt.yscale('log')
    plt.ylim([0.01,6])
    plt.xlim([1e3,1e8])
    plt.xlabel('Vida estimada')
    plt.ylabel('Longitud inic (mm)')
    plt.grid(which = 'major', color = 'k', linestyle='-', linewidth=0.4)
    plt.grid(which = 'minor', color = 'gray', linestyle=':', linewidth
        =0.2)
#    plt.legend([sin_SP[0], SP[0], pts_long_SWT[0]],
#               ['Sin tratamiento', 'Shot peening', 'Electroerosion'],
#               numpoints = 1, loc = 0)
    plt.show()
```

La función *error()* mostrada en el Código 4.17 calcula el antilogaritmo de la media y la desviación típica del logaritmo de los cocientes entre vida estimada y vida experimental, además de realizar una regresión lineal de los resultados para obtener la pendiente de la recta. Con estas variables se podrá

realizar una comparación cuantitativa de los resultados. Como entrada puede recibir de dos a cuatro variables. La primera, “crit” con el nombre del criterio utilizado. La segunda, “vida_exp1” con los resultados experimentales de un grupo de ensayos en un diccionario. Estas dos primeras variables es obligatorio utilizarlas. Las dos siguientes, “vida_exp2” y “vida_exp3” tienen por defecto asignado una lista vacía. Estas variables se pueden utilizar para calcular los parámetros para mas de un grupo de ensayos juntos, en cuyo caso, deben contener los resultados experimentales en un diccionario al igual que ocurría con la variable “vida_exp1”.

Código 4.17 Código de la función *error()*.

```
def error(crit, vida_exp1, vida_exp2 = [], vida_exp3 = []):
    """Funcion para obtener el antilogaritmo de la media y la desviacion
        tipica
    del logaritmo de los cocientes entre vida estimada y vida
        experimental y
    la pendiente m de la recta de regresion de las estimaciones."""

    #Calculamos el numero de resultados excluyendo los de 6629_971_70
    l1 = (len(vida_exp1)-2)*2

    if type(vida_exp2) is dict:
        l2 = (len(vida_exp2)-2)*2
    else:
        l2 = 0

    if type(vida_exp3) is dict:
        l3 = (len(vida_exp3)-2)*2
    else:
        l3 = 0

    #Cargamos los resultados de las simulaciones
    f1 = np.loadtxt('resultados.dat', dtype = str, skiprows = 1)

    if vida_exp2 != 0:
        f2 = np.loadtxt('resultados_SP.dat', dtype = str, skiprows = 1)

    if vida_exp2 != 0:
        f3 = np.loadtxt('resultados_elec.dat', dtype = str, skiprows = 1)


    vida_est1    = {}
    vida_est1_log = {}
    vida_est2    = {}
    vida_est2_log = {}
    vida_est3    = {}
    vida_est3_log = {}

    for i in f1:
        exp_id = i[0]
        par = i[1]
```

```
        if par == crit:
            vida_est1[exp_id] = float(i[2])
            vida_est1_log[exp_id] = math.log10(float(i[2]))

    if type(vida_exp2) is dict:
        for i in f2:
            exp_id = i[0]
            par = i[1]
            if par == crit:
                vida_est2[exp_id] = float(i[2])
                vida_est2_log[exp_id] = math.log10(float(i[2]))

    if type(vida_exp3) is dict:
        for i in f3:
            exp_id = i[0]
            par = i[1]
            if par == crit:
                vida_est3[exp_id] = float(i[2])
                vida_est3_log[exp_id] = math.log10(float(i[2]))

    #Calculamos el antilogaritmo de la media y la desviacion tipica del
    #logaritmo del cociente de la vida estimada y la vida experimental
    alpha = 0.0
    sigma = 0.0

    #Se descartan los resultados del experimento 6629_971_70
    for i in vida_exp1:
        if i != '6629_971_70':
            alpha += math.log10(vida_est1[i]/vida_exp1[i][0])
            alpha += math.log10(vida_est1[i]/vida_exp1[i][1])

    if type(vida_exp2) is dict:
        for i in vida_exp2:
            if i != '6629_971_70':
                alpha += math.log10(vida_est2[i]/vida_exp2[i][0])
                alpha += math.log10(vida_est2[i]/vida_exp2[i][1])

    if type(vida_exp3) is dict:
        for i in vida_exp3:
            if i != '6629_971_70':
                alpha += math.log10(vida_est3[i]/vida_exp3[i][0])
                alpha += math.log10(vida_est3[i]/vida_exp3[i][1])

    alpha_m = alpha/(l1 + l2 + l3)

    for i in vida_exp1:
        if i != '6629_971_70':
            sigma += (math.log10(vida_est1[i]/vida_exp1[i][0])-alpha_m)
                **2.0
```

```
        sigma += (math.log10(vida_est1[i]/vida_exp1[i][1])-alpha_m)
            **2.0

if type(vida_exp2) is dict:
    for i in vida_exp2:
        if i != '6629_971_70':
            sigma += (math.log10(vida_est2[i]/vida_exp2[i][0])-
                alpha_m)**2.0
            sigma += (math.log10(vida_est2[i]/vida_exp2[i][1])-
                alpha_m)**2.0

if type(vida_exp3) is dict:
    for i in vida_exp3:
        if i != '6629_971_70':
            sigma += (math.log10(vida_est3[i]/vida_exp3[i][0])-
                alpha_m)**2.0
            sigma += (math.log10(vida_est3[i]/vida_exp3[i][1])-
                alpha_m)**2.0

sigma_alpha = math.sqrt(sigma/(l1 + l2 + l3-1))

x       = 10**alpha_m
sigma_x = 10**sigma_alpha

print '{}:\tx = {}'.format(crit, x)
print '\tsigma_x = {}'.format(sigma_x)

#Calculamos la pendiente de la recta de regresion
y = []
x = []

#Se descartan los resultados del experimento 6629_971_70
for i in vida_exp1:
    if i != '6629_971_70':
        y.append(vida_est1_log[i])
        y.append(vida_est1_log[i])

        x.append(math.log10(vida_exp1[i][0]))
        x.append(math.log10(vida_exp1[i][1]))

if type(vida_exp2) is dict:
    for i in vida_exp2:
        if i != '6629_971_70':
            y.append(vida_est2_log[i])
            y.append(vida_est2_log[i])

            x.append(math.log10(vida_exp2[i][0]))
            x.append(math.log10(vida_exp2[i][1]))

if type(vida_exp3) is dict:
```

```
        for i in vida_exp3:
            if i != '6629_971_70':
                y.append(vida_est3_log[i])
                y.append(vida_est3_log[i])

                x.append(math.log10(vida_exp3[i][0]))
                x.append(math.log10(vida_exp3[i][1]))

    y = np.array(y)
    x = np.array(x).reshape((-1, 1))

    model = LinearRegression().fit(x, y)
    m = model.coef_[0]

    print '\tm = {}'.format(m)
```

En el Código 4.18 se muestra la función *grafs_long_grietas()* en la que se realiza una gráfica con la evolución de la longitud de la grieta en función del número de ciclos de un experimento concreto. Esta gráfica ya se genera automáticamente al ejecutar la función *principal()* del *script principal.py*. Como entrada recibe dos variables; "exp", que contiene en un *string* el identificador del experimento del que se quiere realizar la gráfica y "crit", el nombre del criterio utilizado para conseguir los resultados.

Código 4.18 Código de la función *grafs_long_grietas()*.

```
def grafs_long_grieta(exp, crit):
    """Funcion para obtener las curvas con la evolución de la grieta en
    funcion del numero de ciclos."""

    cwd = os.getcwd()
    ruta_datos = cwd + '/resultados/datos/{}/'.format(crit)

    f = np.loadtxt('{}{}.dat'.format(ruta_datos, exp), dtype = float,
                   skiprows = 1)

    a_i = []
    N_a = []

    for i in f:
        a_i.append(i[0])
        N_a.append(i[4])

    plt.figure('Longitud de grieta')
    plt.xscale('log')
    plt.xlabel('Ciclos')
    plt.ylabel('Longitud de grieta (mm)')
    plt.plot(N_a, a_i, 'k')
    plt.xlim([5e2,1e8])
    plt.show()
```

```
    a_i = []
    N_a = []
```

La función *grafs_vida_est()* del Código 4.19 realiza las gráficas con las curvas de vida de iniciación, propagación y total en función de la longitud de iniciación. Estas curvas también se obtienen al ejecutar la función *principal()* del *script principal.py* Como entrada recibe las mismas dos variables que la función anterior; "exp", con el identificador del experimento y "crit" con el nombre del criterio utilizado.

Código 4.19 Código de la función *grafs_vida_est()*.

```
def grafs_vida_est(exp, crit):
    """Funcion para obtener las curvas de vida."""

    cwd = os.getcwd()
    ruta_datos = cwd + '/resultados/datos/{}/'.format(crit)

    f = np.loadtxt('{}{}.dat'.format(ruta_datos, exp), dtype = float,
                   skiprows = 1)

    a_i = []
    N_t = []
    N_i = []
    N_p = []

    for i in f:
        a_i.append(i[0])
        N_t.append(i[1])
        N_i.append(i[2])
        N_p.append(i[3])

    N_t_min  = min(N_t)
    i_N_t_min = N_t.index(N_t_min)
    a_inic   = a_i[i_N_t_min]

    plt.figure(exp + '_' + crit)
    plt.xlabel('Longitud de iniciacion (mm)')
    plt.ylabel('Ciclos')
    plt.yscale('log')
    plt.ylim([1e3,1e8])
    plt.xlim([0.0,a_i[-1]])
    plot_inic = []
    plot_prop = []
    plot_tot  = []
    plot_inic += plt.plot(a_i, N_i, 'b')
    plot_prop += plt.plot(a_i, N_p, 'k')
    plot_tot += plt.plot(a_i, N_t, 'r')

    #Pintamos el punto donde se da el minimo
    plt.plot(a_inic, N_t_min, 'ro')
    plt.legend([plot_inic[0], plot_prop[0], plot_tot[0]],
```

```
            ['Vida de iniciacion', 'Vida de propagacion', 'Vida total'
                ], loc = 0)
    plt.show()
```

5 Resultados

Para validar el código se van a comparar los resultados obtenidos con los resultados experimentales mostrados en el Capítulo 3. Se han utilizado los dos parámetros descritos anteriormente, Fatemi-Socie y Smith-Watson-Topper. Para cada uno de estos parámetros se han realizado los cálculos considerando frentes de grieta elípticos, con una relación de aspecto $\Phi = 0.5$ y planos tanto en la fase de iniciación como en la fase de propagación.

A continuación, en las Figuras 5.1-5.4 se muestran las vidas estimadas frente a las vidas experimentales para todos los casos. En estas figuras, las dos flechas indican que ese ensayo realmente son dos y que se interrumpió antes de ser finalizado, por lo que los puntos en realidad saldrían desplazados hacia la derecha.

Para realizar una comparación cuantitativa se van a utilizar tres parámetros distintos. Por un lado, el antilogaritmo de la media ($\bar{x}$) y la desviación típica (ς_x) de los logaritmos de los cocientes entre las vidas estimadas y las vidas experimentales. El cálculo de estos dos parámetros se muestra en las Ecuaciones 5.1 - 5.4. Por otro lado se va a realizar una regresión lineal de los resultados para obtener la pendiente m de la recta. $\bar{x}$ indica la desviación media respecto a la recta que marca estimaciones perfectas, por lo que se va a utilizar como una medida de la subestimación o sobreestimación de las vidas. ς_x se va a utilizar como una medida de la dispersión de los resultados. Por último, m va a reflejar si las estimaciones siguen la tendencia de los resultados experimentales. Si los tres parámetros fuesen igual a 1 significaría que la estimación de las vidas es perfecta; sin embargo, debido a la propia dispersión de los resultados experimentales nunca se van a llegar a obtener estos resultados. Por lo tanto, la mejor estimación es aquella en la que estos parámetros se aproximan más a 1.

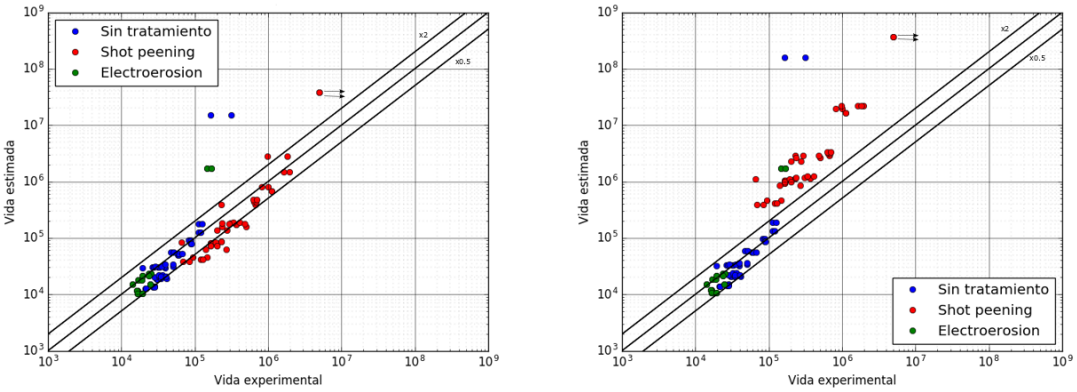
En la Tabla 5.1 se muestran los valores de los parámetros para los tres grupos de ensayo juntos. A la hora de obtener los parámetros se han obviado los resultados obtenidos para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$; ya que en todos los casos se sobrestima la vida en un número elevado de ciclos y provoca que los valores de los parámetros se alejen excesivamente de 1 y no permite una comparación útil.

$$\alpha_i = \log \frac{N_{\text{est}_i}}{N_{\text{exp}_i}} \tag{5.1}$$

$$\bar{\alpha} = \frac{1}{N} \sum_{i=1}^{N} \alpha_i \tag{5.2}$$

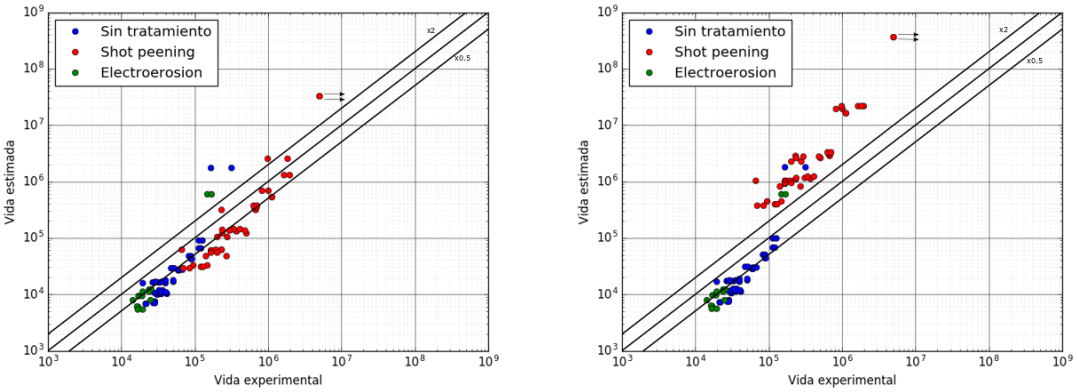
$$\varsigma_\alpha = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (\alpha_i - \bar{\alpha})^2} \tag{5.3}$$

$$\bar{x} = 10^{\bar{\alpha}} \mid \varsigma_x = 10^{\varsigma_\alpha} \tag{5.4}$$



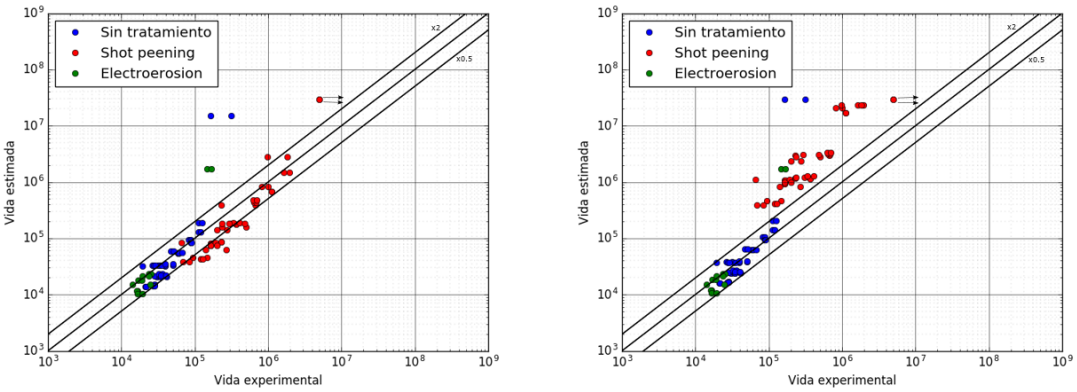
(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.1 Vida estimada utilizando iniciación y propagación con grieta elíptica.



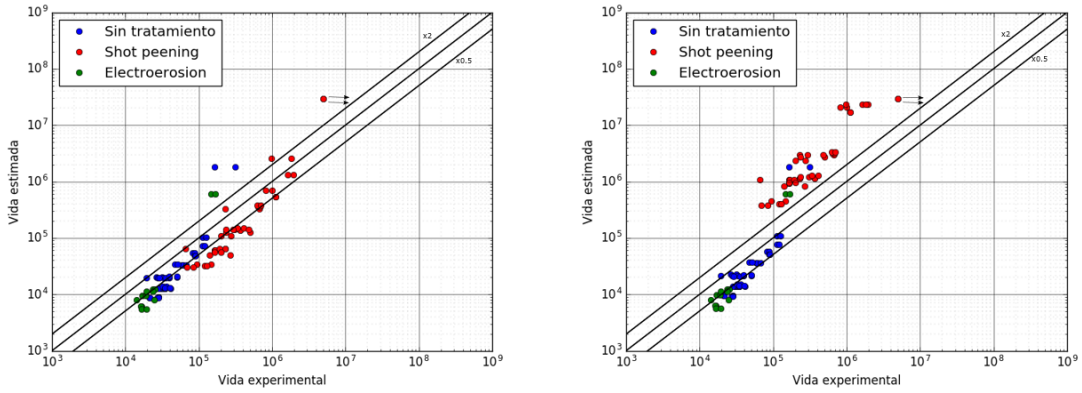
(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.2 Vida estimada utilizando iniciación con grieta elíptica y propagación con grieta plana.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.3 Vida estimada utilizando iniciación con grieta plana y propagación con grieta elíptica.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.4 Vida estimada utilizando iniciación y propagación con grieta plana.

Tabla 5.1 $\bar{x}, \varsigma_x$ y m para los grupos de ensayos juntos.

	Iniciación plana						Iniciación elíptica					
	Prop. plana			Prop. elíptica			Prop. plana			Prop. elíptica		
	$\bar{x}$	ς_x	m	$\bar{x}$	ς_x	m	$\bar{x}$	ς_x	m	$\bar{x}$	ς_x	m
FS	0.45	1.53	1.06	0.69	1.56	0.95	0.41	1.54	1.10	0.67	1.55	0.96
SWT	1.62	4.05	1.90	2.31	3.08	1.72	1.47	4.28	1.93	2.17	3.17	1.73

A partir de las gráficas y la tabla anterior se puede observar como el parámetro Fatemi-Socie obtiene mejores resultados que Smith-Watson-Topper utilizando este modelo. A partir del parámetro $\bar{x}$ se puede comprobar como con Fatemi-Socie se subestiman las vidas mientras que con Smith-Watson-Topper se sobreestiman, siendo con Fatemi-Socie con el parámetro de fatiga multiaxial al que más se aproxima a 1. En cuanto a ς_x, hay diferencias considerables entre ambos modelos, presentando Smith-Watson-Topper una dispersión mucho mayor de los resultados. Mirando los valores de la pendiente m se observa como nuevamente con Fatemi-Socie este parámetro se aproxima más a 1 que con Smith-Watson-Topper.

Respecto a los diferentes frentes de grieta utilizados, las diferencias entre utilizar un frente de grieta elíptico o plano en la fase de iniciación son pequeñas, sobre todo para Fatemi-Socie. Por ello, a partir de este momento solo se utilizará frente de grieta elíptico en la fase de iniciación, ya que tiene más sentido que el frente de grieta durante esta fase sea así. Al utilizar frentes de grieta distintos durante la fase de propagación si aparecen mayores diferencias. Con Fatemi-Socie, $\bar{x}$ se aproxima más a 1 utilizando frente de grieta elíptico; ς_x apenas presenta variaciones entre ambos frentes de grieta y la pendiente pasa de ser superior a 1 para frente de grieta plano a menor que 1 para frente de grieta elíptico. En vista de esto, los mejores resultados se obtienen utilizando Fatemi-Socie con frente de grieta elíptico en la fase de propagación.

En la Tabla 5.2 se muestran los resultados obtenidos para los tres parámetros para cada grupo de ensayos por separado utilizando siempre frente de grieta elíptico en la fase de iniciación y comparando entre los dos criterios de fatiga multiaxial y entre los dos tipos de frente de grieta.

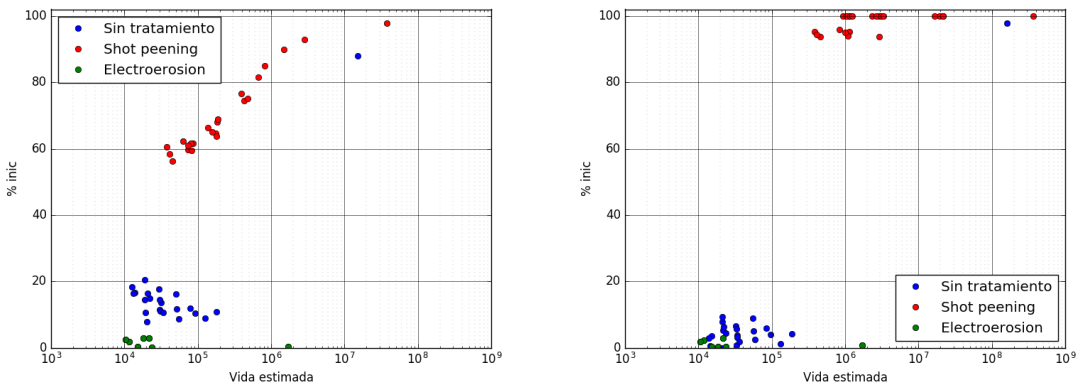
Se puede observar como tanto para el grupo de ensayos sin tratamiento superficial como para el grupo de ensayos con electroerosión apenas existen diferencias entre Fatemi-Socie y Smith-Watson-Topper. Es en el grupo de ensayos con *shot peening* donde aparecen diferencias significativas. Aunque la pendiente m es prácticamente la misma, ς_x en menor medida y sobre todo $\bar{x}$ son superiores en el

caso de Smith-Watson-Topper. En cuanto a las diferencias entre utilizar propagación con frente de grieta elíptico o plano, se puede observar como el único parámetro que varía significativamente es $\bar{x}$ que se aproxima mucho más a 1 en el caso de propagación elíptica. Por último, se puede comprobar como el grupo de ensayos con el que el modelo obtiene mejores estimaciones es el grupo con electroerosión, seguido del grupo sin tratamiento superficial. El grupo de ensayos que obtiene peores resultados es el grupo sometido a *shot peening*. Se puede concluir a partir de estos resultados que el modelo parece obtener mejores resultados cuando las vidas son menores.

Tabla 5.2 $\bar{x}, \varsigma_x$ y m para cada grupo de ensayos por separado.

Sin tratamiento superficial						
	Prop. plana			Prop. elíptica		
	$\bar{x}$	ς_x	m	$\bar{x}$	ς_x	m
FS	0.40	1.40	1.30	0.77	1.41	1.32
SWT	0.43	1.40	1.29	0.83	1.40	1.30
Con *shot peening*						
	Prop. plana			Prop. elíptica		
	$\bar{x}$	ς_x	m	$\bar{x}$	ς_x	m
FS	0.43	1.71	1.29	0.56	1.64	1.24
SWT	7.05	1.83	1.28	7.13	1.83	1.27
Con electroerosión						
	Prop. plana			Prop. elíptica		
	$\bar{x}$	ς_x	m	$\bar{x}$	ς_x	m
FS	0.37	1.40	0.96	0.80	1.33	0.97
SWT	0.37	1.39	0.93	0.81	1.32	0.93

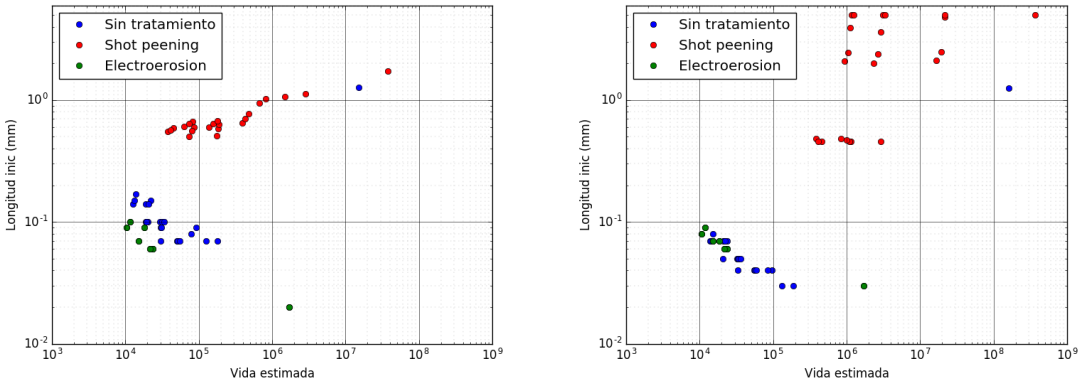
En la Figura 5.5 se muestra la proporción de vida dedicada a la iniciación en función de la vida estimada para FS y SWT utilizando propagación elíptica. Se puede ver como en términos generales, a mayor vida se obtienen mayores porcentajes de iniciación. Sin embargo, tanto en el grupo de ensayos sin tratamiento superficial como en el grupo de ensayos sometido a electroerosión la tendencia es la contraria; ya que el porcentaje de iniciación va disminuyendo. En cuanto a las diferencias entre los parámetros, utilizando SWT se obtienen mayores porcentajes de iniciación para vidas altas y menores para vidas bajas.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.5 Porcentaje de iniciación utilizando propagación con grieta elíptica.

En la Figura 5.6 se muestra la longitud de iniciación en función de la vida estimada para FS y SWT utilizando propagación elíptica. La tendencia de estas gráficas es la misma que la que se produce en las mencionadas anteriormente. En términos generales, la longitud de iniciación es mayor para vidas mayores. Sin embargo, en los grupos de ensayo sin tratamiento superficial y con electroerosión; la longitud de iniciación disminuye al aumentar la vida. Aparte, con SWT se obtienen mayores longitudes de iniciación para vidas altas y viceversa para vidas bajas.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.6 Longitud de iniciación utilizando propagación con grieta elíptica.

En las Figuras 5.8 y 5.9 se muestra la evolución de la longitud de la grieta en función del número de ciclos para la combinación de cargas $\sigma = 110|N = 971|Q = 5429$ sin tratamiento superficial y se compara con los resultados experimentales comentados en el Capítulo 3. La obtención de estas curvas se realiza a partir de las curvas de vida. Hasta la longitud de iniciación la evolución de la grieta está influenciada por la fase de iniciación. Por ello, entre dos longitudes de grieta sucesivas el número de ciclos necesario para propagar la grieta crece como la diferencia entre los ciclos de esos dos puntos de la curva de iniciación. A partir de la longitud de iniciación, la evolución de la grieta está influenciada por la fase de propagación. Debido a esto, entre dos longitudes de grieta sucesivas los ciclos necesarios para propagar la grieta serán la diferencia entre los ciclos de la curva de propagación entre esos dos puntos. En la Figura 5.7 se muestra como se debe hacer el cálculo tanto en puntos anteriores a la longitud de iniciación como en punto posteriores. Cabe destacar que debido a que la curva de propagación es decreciente, la longitud de la grieta va a presentar una asíntota; de forma que la grieta avanza muy rápido para un número pequeño de ciclos al final de la vida de la misma.

Lo ideal sería que la curva se aproximase lo máximo posible a los resultados de los ensayos interrumpidos; mientras que debería sobrepasar los resultados de los ensayos hasta rotura. Esto se debe a que en los ensayos hasta rotura, lo que se ha medido es la grieta secundaria; mientras que la principal evidentemente habrá tenido una longitud superior. Se puede comprobar como nuevamente el modelo se aproxima más a los resultados experimentales utilizando el parámetro Fatemi-Socie y la propagación elíptica. En este caso, de todas formas, utilizando Smith-Watson-Topper junto con grieta elíptica también se obtienen resultados que siguen más o menos la tendencia real.

En las Figuras 5.10 - 5.12 se muestra la evolución de la longitud de la grieta para todas las combinaciones de carga de los tres grupos de ensayo. Los casos mostrados se corresponden con los realizados utilizando el criterio de Fatemi-Socie junto con propagación elíptica, con la que se han obtenido los mejores resultados. Se puede comprobar como la evolución de las grietas sigue el mismo patrón en todos los casos. Para pocos ciclos de carga, durante la fase de iniciación, la grieta

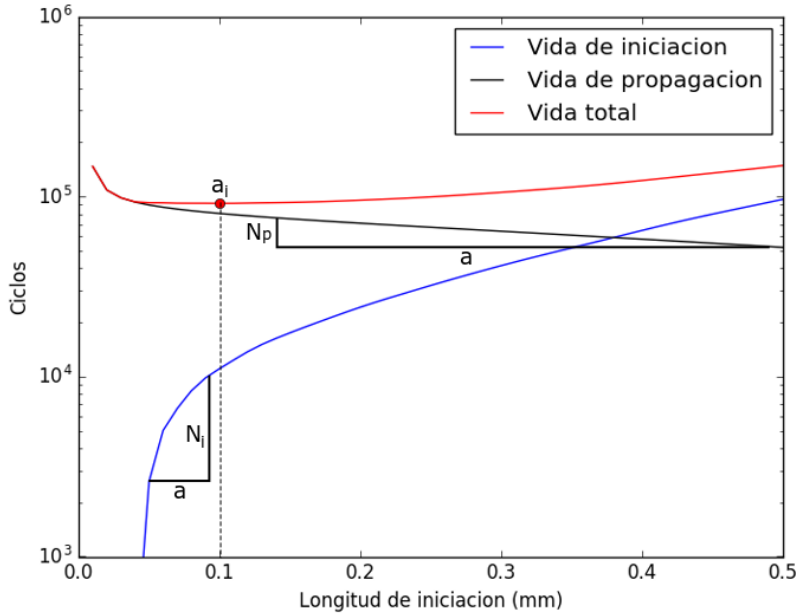
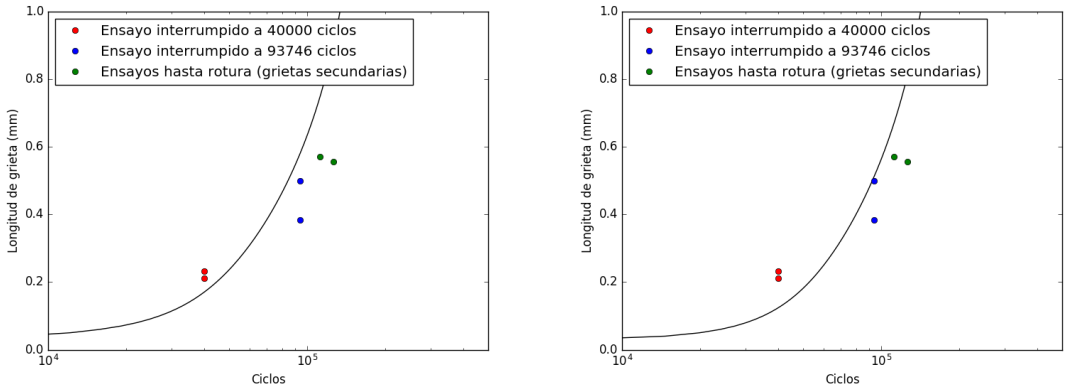


Figura 5.7 Longitud de la grieta a partir de las curvas de vida.

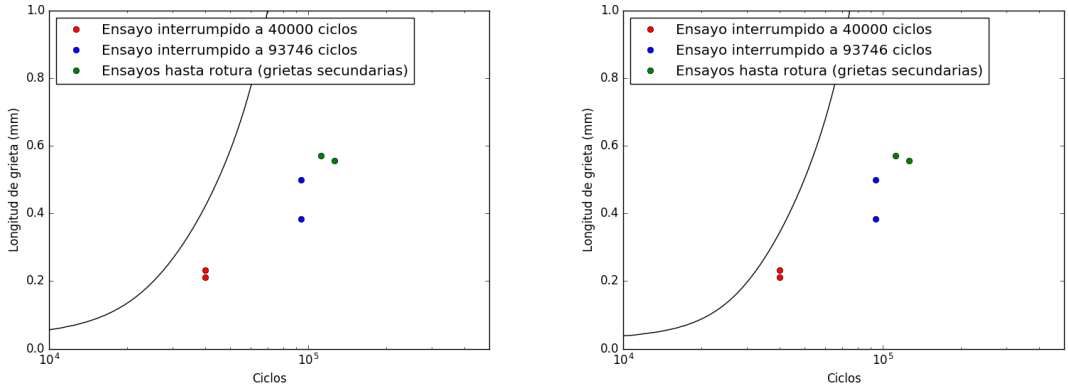


(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.8 Longitud de la grieta utilizando propagación con grieta elíptica.

crece lentamente. A medida que se aumenta el número de ciclos, la velocidad de propagación de la grieta aumenta hasta que llegado un momento determinado crece inestablemente. El grupo de ensayos con *shot peening* presentan una ligera desviación respecto a este comportamiento ya que parece producirse una pequeña ralentización en la propagación de la grieta justo antes de que esta se propague inestablemente.

A continuación, se muestran curvas de vida de iniciación, de propagación y total para algunas combinaciones de carga concretas para los distintos grupos de ensayo; junto con la evolución de la tensión σ_x. En primer lugar, en las Figuras 5.13 - 5.26 se muestran ejemplos para ensayos realizados en probetas sin tratamiento superficial. En todos los casos, las figuras mostradas se corresponden a propagación elíptica. Como recordatorio, las curvas de propagación son las mismas para Fatemi-Socie y Smith-Watson-Topper; ya que estos criterios solo se utilizan en la fase de iniciación.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.9 Longitud de la grieta utilizando propagación con grieta plana.

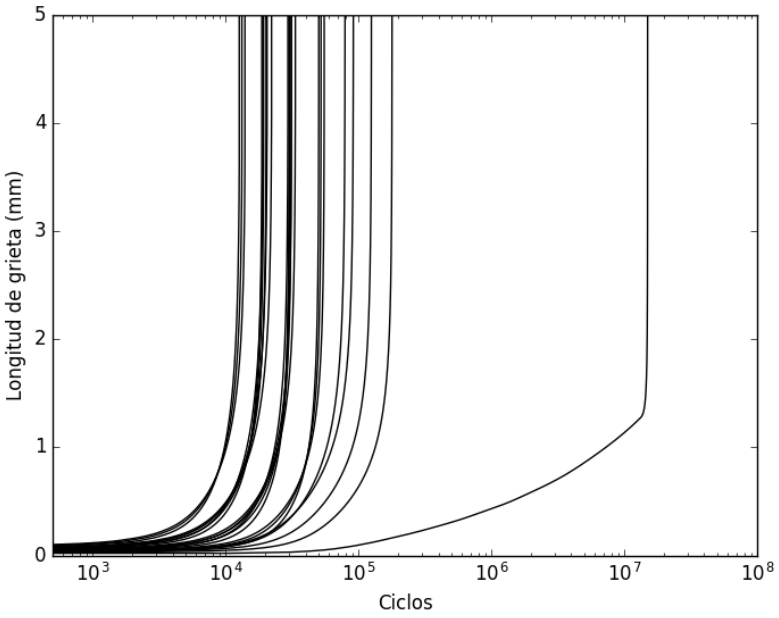


Figura 5.10 Evolución de la longitud de la grieta para los casos sin tratamiento superficial.

Se observa como en todos los casos, las curvas de iniciación de Smith-Watson-Topper están situadas por encima de las curvas de Fatemi-Socie. Sin embargo, la evolución de la curva es muy similar y las diferencias entre ambas no son excesivas. Por otro lado, en lo que respecta a las diferencias entre las distintas combinaciones de cargas, se puede comprobar como la tensión global σ tiene una influencia mucho mayor que las cargas de contacto N y Q. Mientras que para las tensiones de contacto apenas aparecen diferencias apreciables; se puede ver como a mayor tensión global, menor vidas se obtienen, tal y como cabría esperar.

Mención aparte merece la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ mostrada en la Figura 5.14, en la que se puede apreciar como la evolución de la curva de propagación cambia radicalmente respecto al resto. Esta se corresponde al experimento realizado con una menor tensión global y la curva de propagación presenta una asíntota vertical, lo que desplaza en gran medida la longitud de iniciación y retrasa el fallo de la probeta. Además, mientras que para el resto de

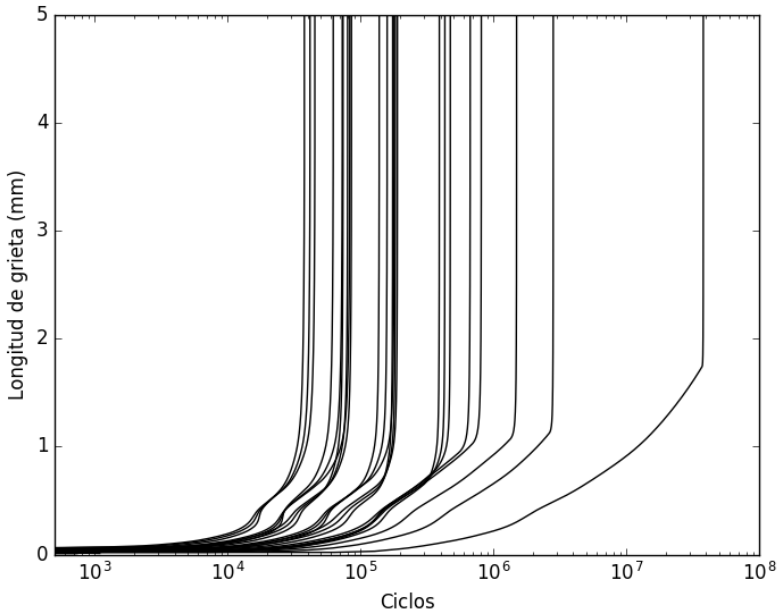


Figura 5.11 Evolución de la longitud de la grieta para los casos con *shot peening*.

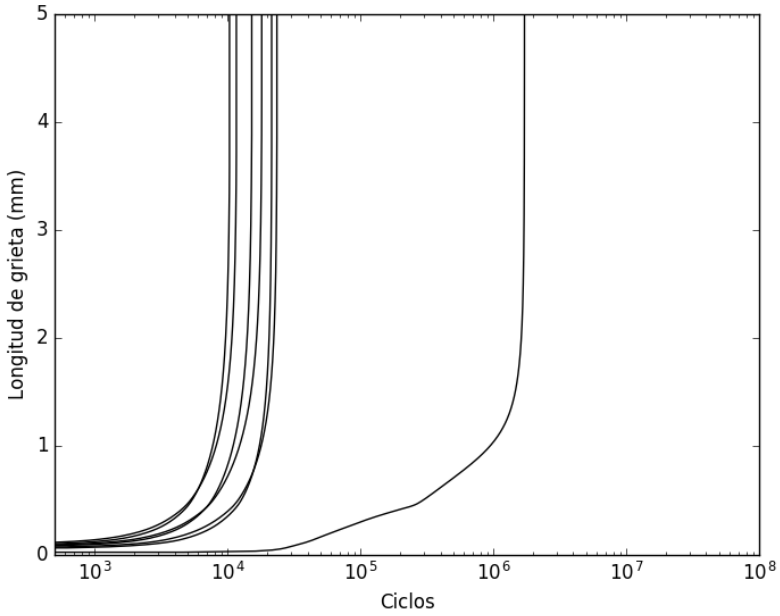


Figura 5.12 Evolución de la longitud de la grieta para los casos con electroerosión.

combinaciones de cargas de este grupo de ensayos la propagación es más importante; para esta combinación de cargas es la fase de iniciación la que cobra mayor relevancia.

En las Figuras 5.27 - 5.40 se muestran las curvas para las mismas combinaciones de carga, aunque para probetas con *shot peening* junto con la evolución de la tensión σ_x. Se aprecia como nuevamente es la tensión global la que tiene una mayor influencia en los resultados; mientras que variar las cargas de contacto apenas produce diferencias en las curvas.

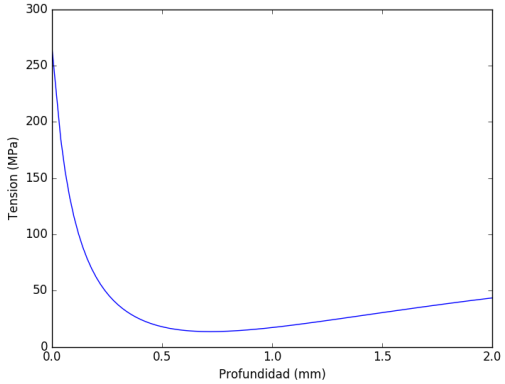
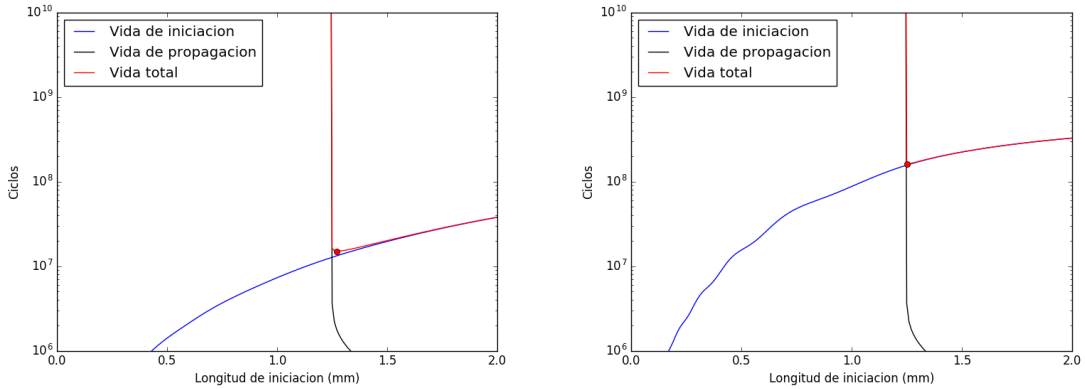


Figura 5.13 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ sin tratamiento superficial.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.14 Curvas de vida estimada para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ sin tratamiento superficial.

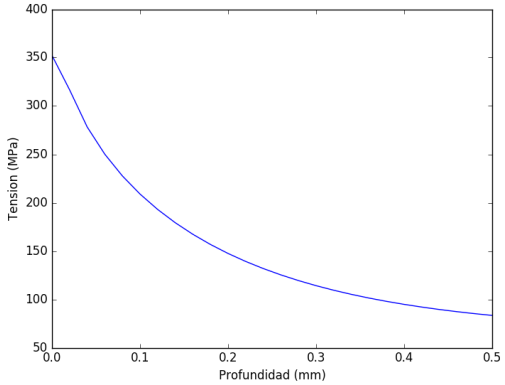
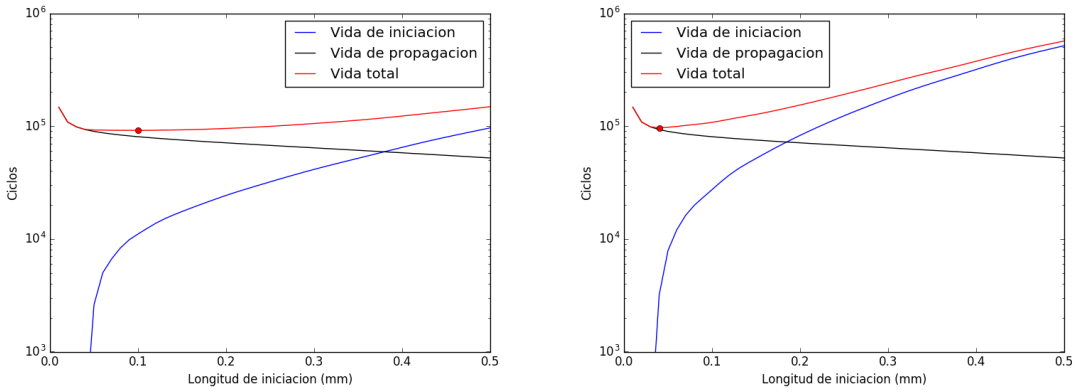


Figura 5.15 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 110|N = 1543|Q = 5429$ sin tratamiento superficial.

En este caso la diferencia entre los dos criterios se hace más evidente. Las curvas de iniciación realizadas con el criterio de Smith-Watson-Topper vuelven a estar situadas por encima de las



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.16 Curvas de vida estimada para la combinación de cargas $\sigma = 110|N = 1543|Q = 5429$ sin tratamiento superficial.

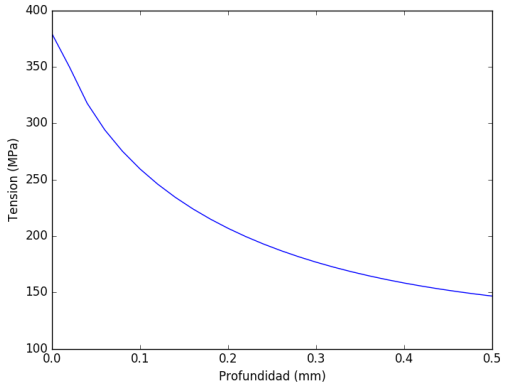
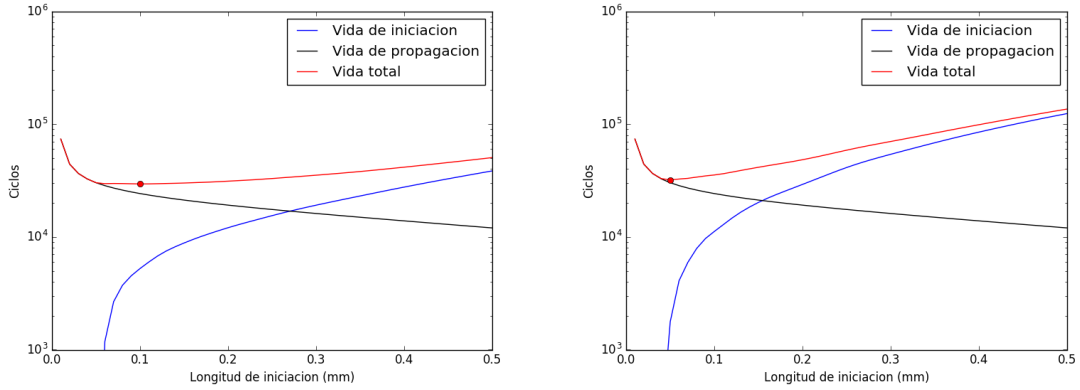


Figura 5.17 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 1543|Q = 3006$ sin tratamiento superficial.

realizadas con el criterio de Fatemi-Socie. Sin embargo, se puede observar como las curvas de Smith-Watson-Topper están mucho más influenciadas por las tensiones residuales introducidas en las probetas, presentando un máximo; a diferencia del grupo de ensayos anterior en el que las curvas de iniciación eran monótonamente crecientes. Las curvas de Fatemi-Socie presentan una evolución similar a las mostradas en las curvas del grupo de ensayo sin tratamiento superficial.

Las curvas de propagación presentan en todos los casos una asíntota vertical. Con el criterio de Smith-Watson-Topper, en algunos casos el mínimo de la curva de vida total coincide con el primer mínimo relativo de la curva de iniciación y en otros casos se da para longitudes de iniciación muy superiores; dependiendo de donde se sitúe la asíntota de la curva de propagación.

La combinación $\sigma = 70|N = 971|Q = 6629$ vuelve a tener un comportamiento diferente al resto de combinaciones de cargas. En el caso de Smith-Watson-Topper, la curva de iniciación presenta una fluctuación hasta que se estabiliza. Además, se observa como la longitud de iniciación se sitúa en los 5 mm. Esta es la profundidad máxima a la que se tienen los datos del tensor de tensiones y deformaciones, por lo que es imposible seguir propagando la grieta y llegar al mínimo real de la curva de vida total. De todas formas, la curva de iniciación para estas longitudes es lo suficientemente plana como para que no vaya a variar demasiado el resultado final.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.18 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 1543|Q = 3006$ sin tratamiento superficial.

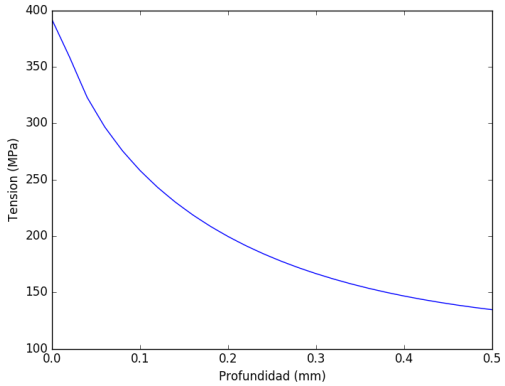
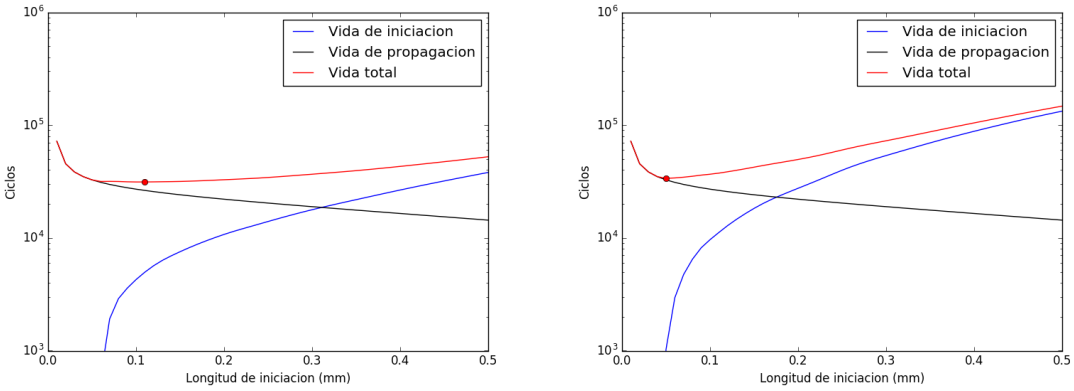


Figura 5.19 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 1543|Q = 4217$ sin tratamiento superficial.

Por último, en las Figuras 5.41 - 5.54 se muestran las curvas para todas las combinaciones de carga del grupo de ensayos con electroerosión junto con la evolución de la tensión σ_x. En este caso, las curvas son muy similares a las obtenidas en el grupo de ensayos sin tratamiento superficial. Esto se debe al hecho de que la evolución del tensor de tensiones y deformaciones es muy similar entre ambos grupos de ensayos, como se pudo pudo comprobar en el Capítulo 3. Con Smith-Watson-Topper se vuelven a obtener vidas estimadas superiores a las que se obtienen utilizando Fatemi-Socie.

En el caso de la combinación de cargas $\sigma = 70|N = 971|Q = 6629$, sí que aparecen diferencias significativas con respecto al caso sin tratamiento superficial. Ahora, la longitud de iniciación se ha adelantado y la curva de propagación ya no presenta una asíntota vertical.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.20 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 1543|Q = 4217$ sin tratamiento superficial.

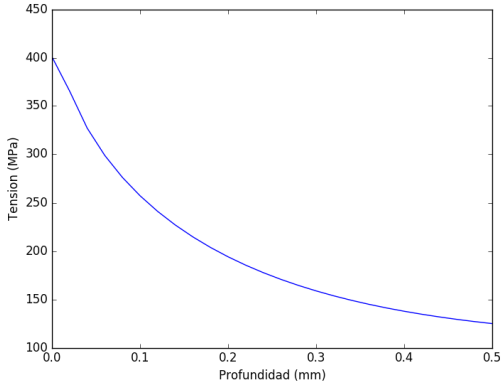
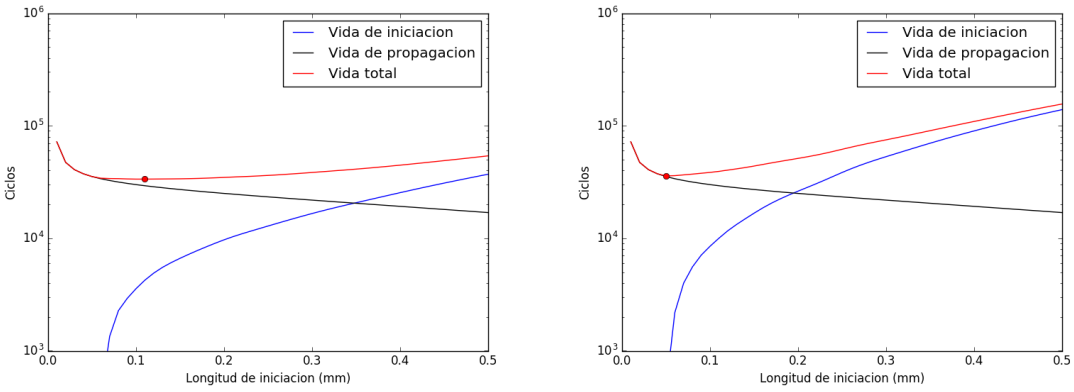


Figura 5.21 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 1543|Q = 5429$ sin tratamiento superficial.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.22 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 1543|Q = 5429$ sin tratamiento superficial.

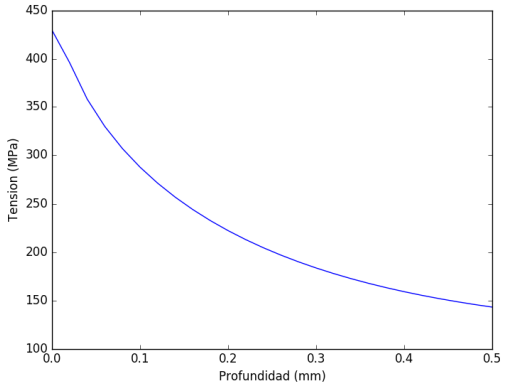
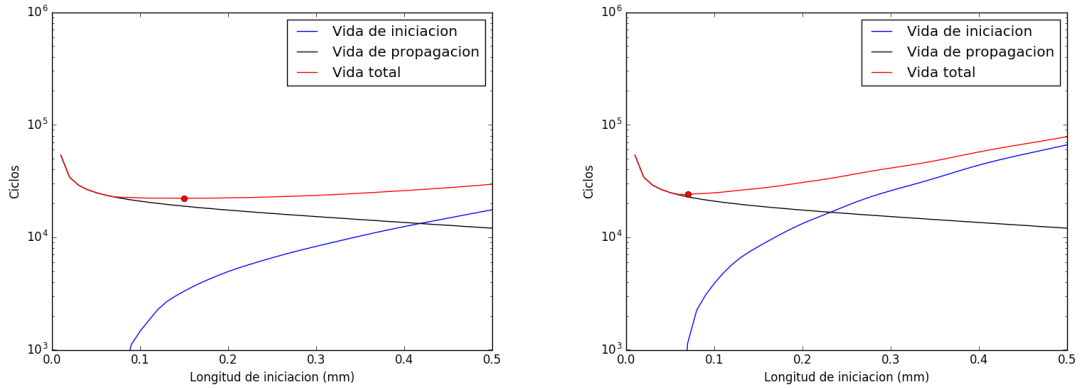


Figura 5.23 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 2113|Q = 5429$ sin tratamiento superficial.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.24 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 2113|Q = 5429$ sin tratamiento superficial.

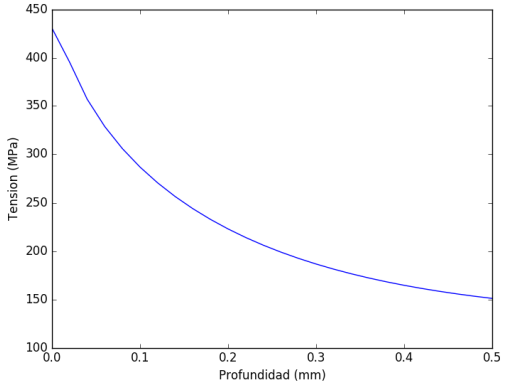
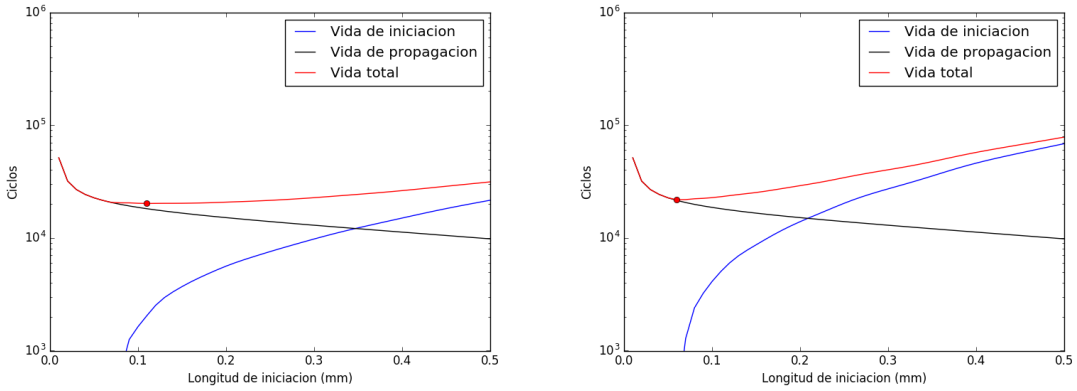


Figura 5.25 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 175|N = 1543|Q = 5429$ sin tratamiento superficial.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.26 Curvas de vida estimada para la combinación de cargas $\sigma = 175|N = 1543|Q = 5429$ sin tratamiento superficial.

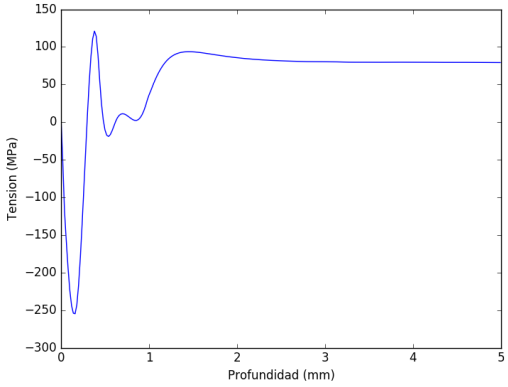
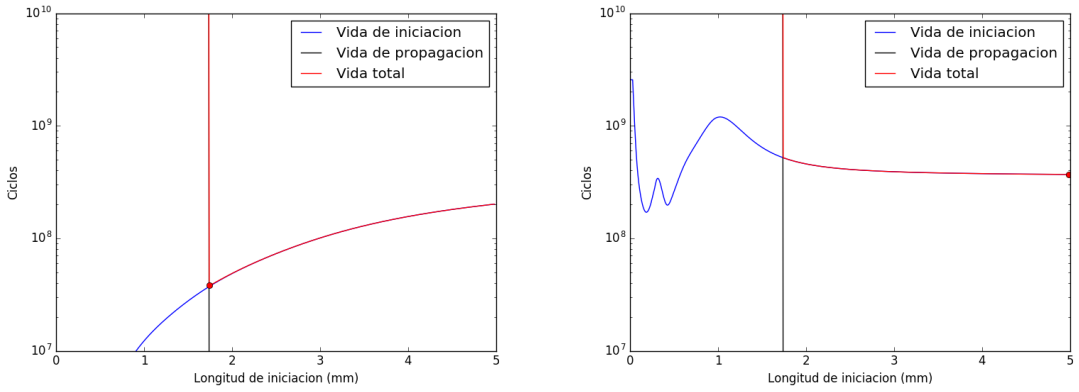


Figura 5.27 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.28 Curvas de vida estimada para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ con *shot peening*.

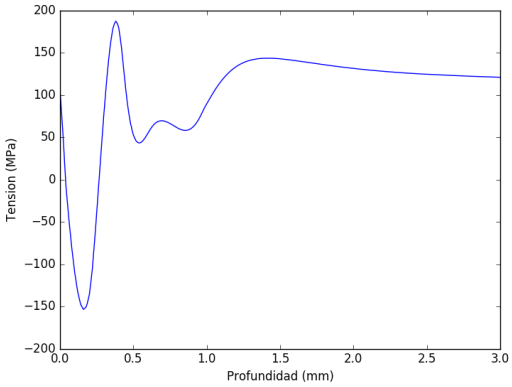
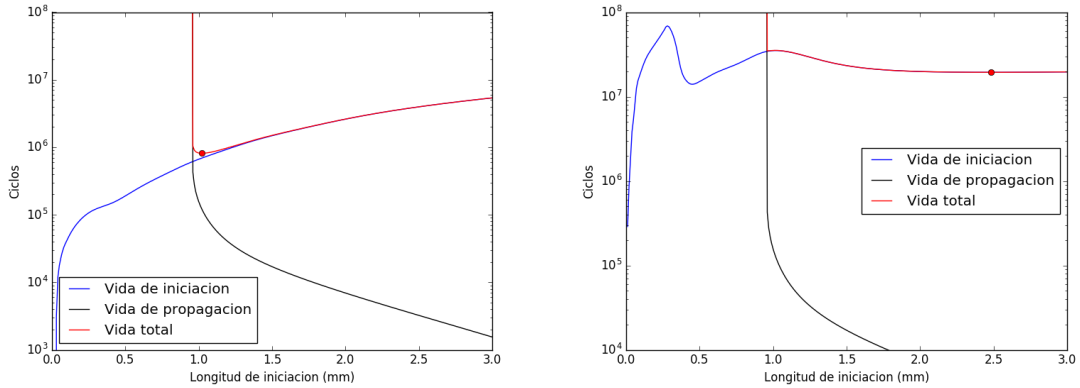


Figura 5.29 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 110|N = 1543|Q = 5429$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.30 Curvas de vida estimada para la combinación de cargas $\sigma = 110|N = 1543|Q = 5429$ con *shot peening*.

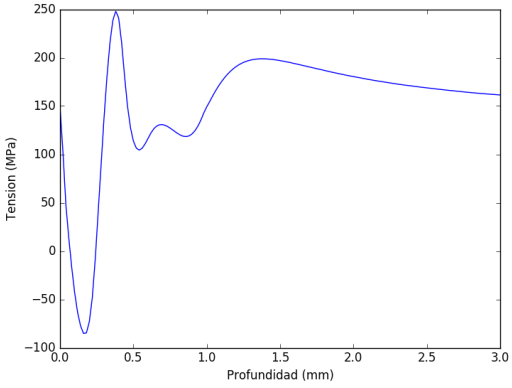
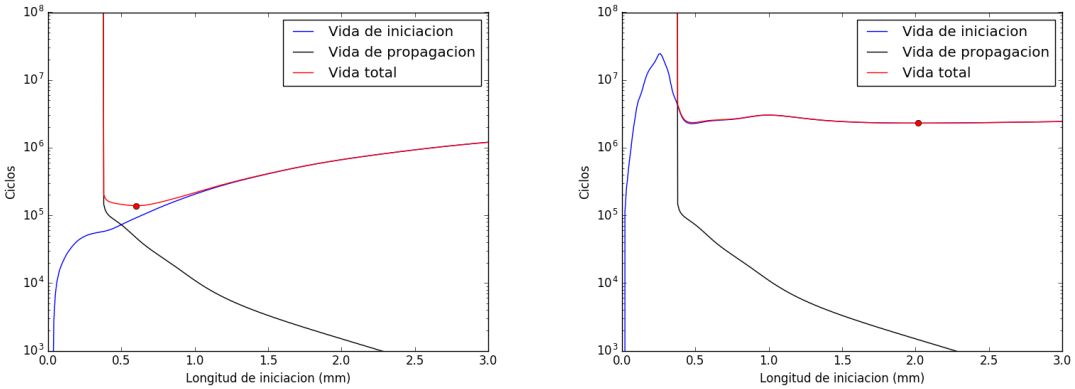


Figura 5.31 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 1543|Q = 3006$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.32 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 1543|Q = 3006$ con *shot peening*.

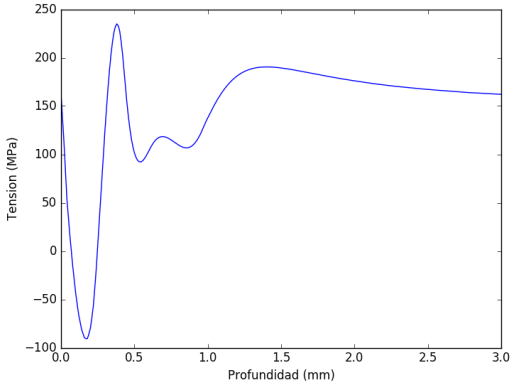
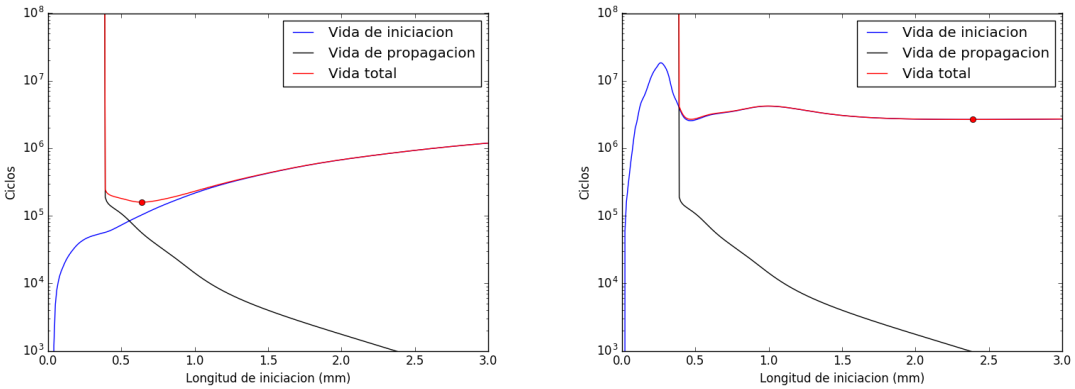


Figura 5.33 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 1543|Q = 4217$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.34 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 1543|Q = 4217$ con *shot peening*.

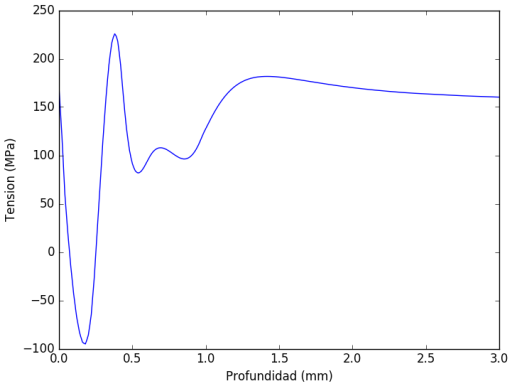
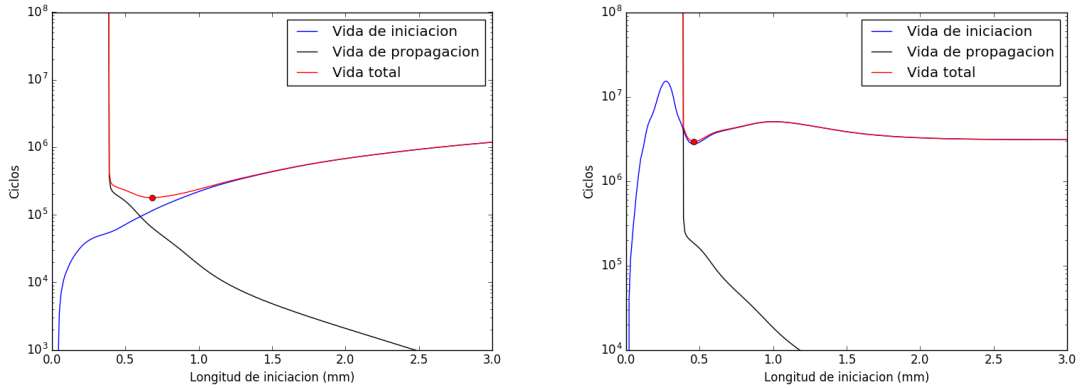


Figura 5.35 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 1543|Q = 5429$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.36 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 1543|Q = 5429$ con *shot peening*.

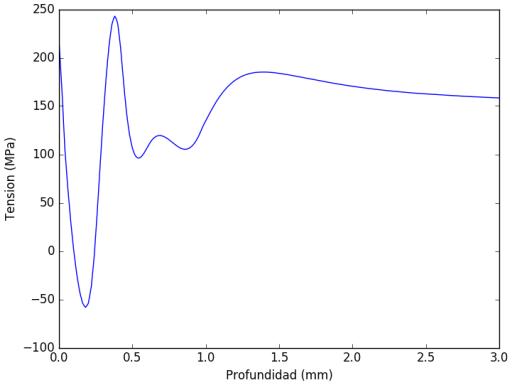
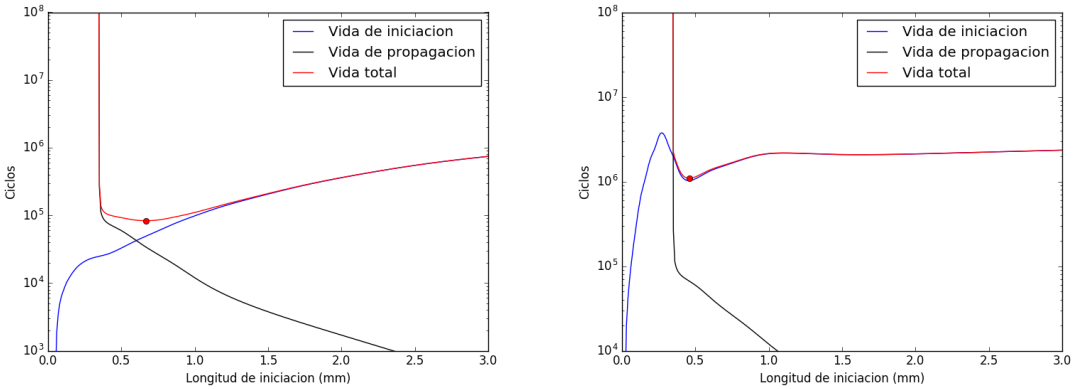


Figura 5.37 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 2113|Q = 5429$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.38 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 2113|Q = 5429$ con *shot peening*.

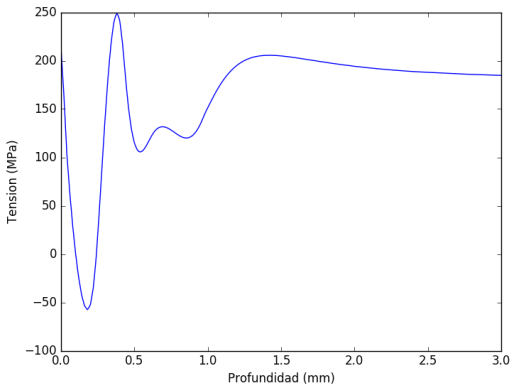
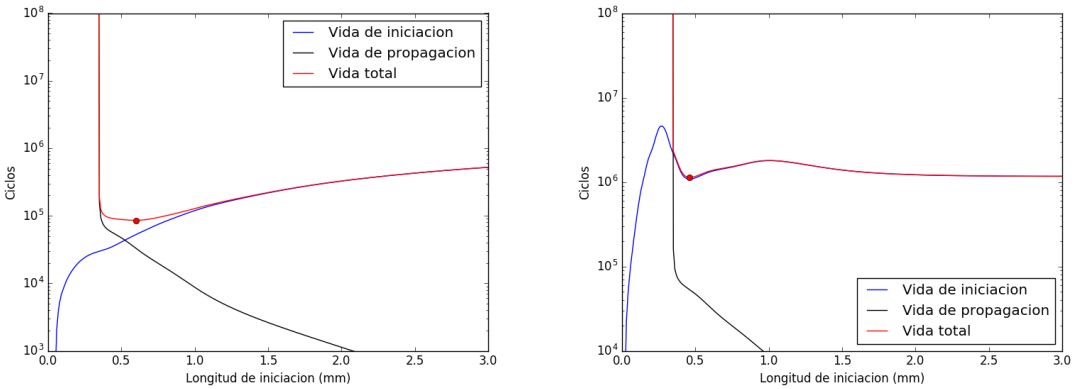


Figura 5.39 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 175|N = 1543|Q = 5429$ con *shot peening*.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.40 Curvas de vida estimada para la combinación de cargas $\sigma = 175|N = 1543|Q = 5429$ con *shot peening*.

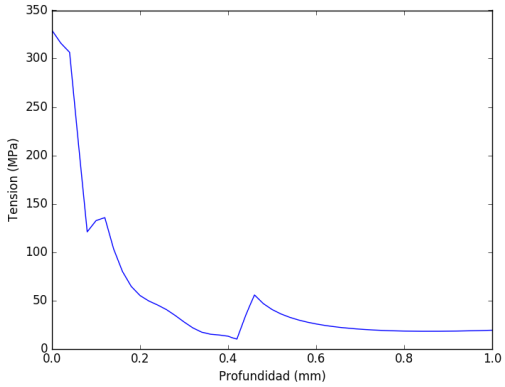
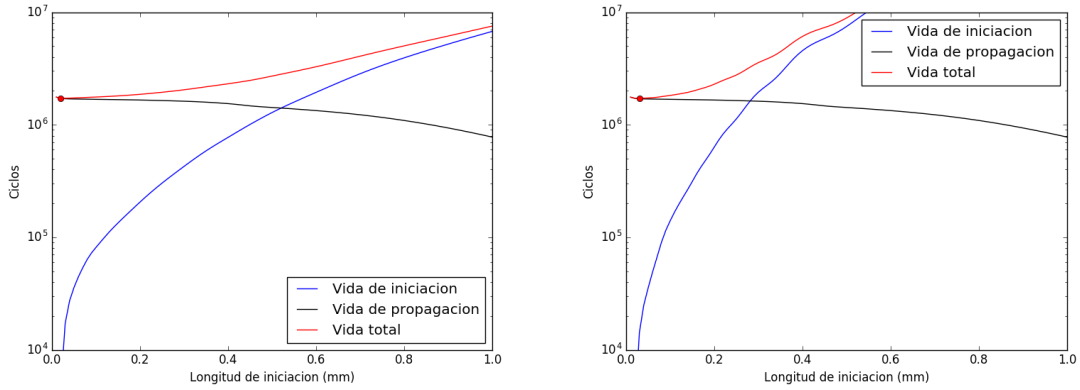


Figura 5.41 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ con electroerosión.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.42 Curvas de vida estimada para la combinación de cargas $\sigma = 70|N = 971|Q = 6629$ con electroerosión.

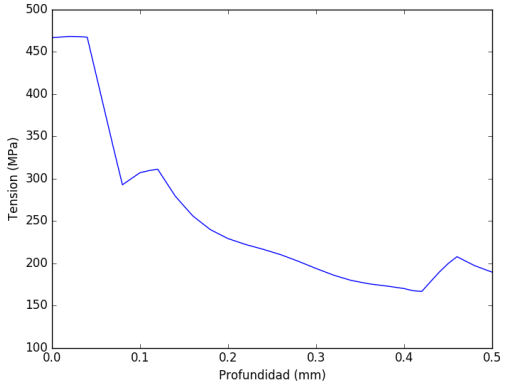
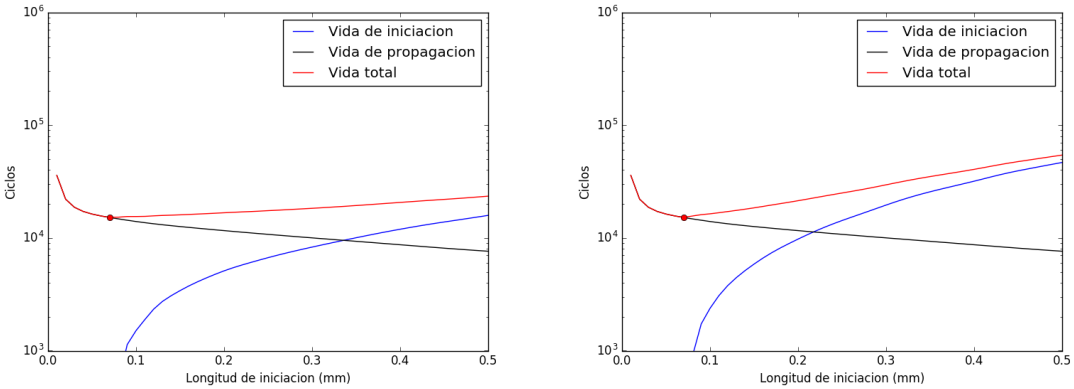


Figura 5.43 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 2113|Q = 3006$ con electroerosión.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.44 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 2113|Q = 3006$ con electroerosión.

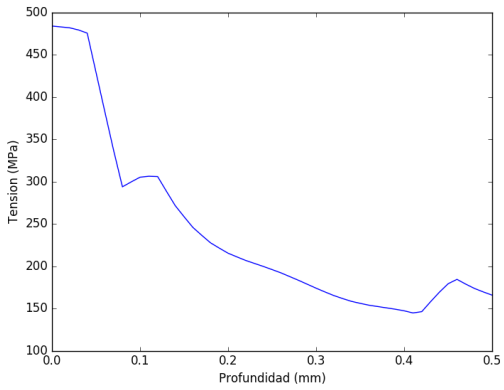
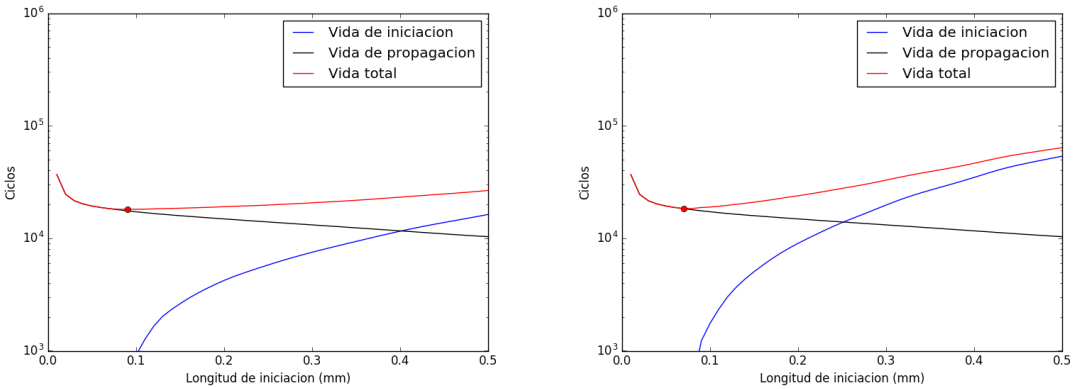


Figura 5.45 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 150|N = 2113|Q = 5429$ con electroerosión.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.46 Curvas de vida estimada para la combinación de cargas $\sigma = 150|N = 2113|Q = 5429$ con electroerosión.

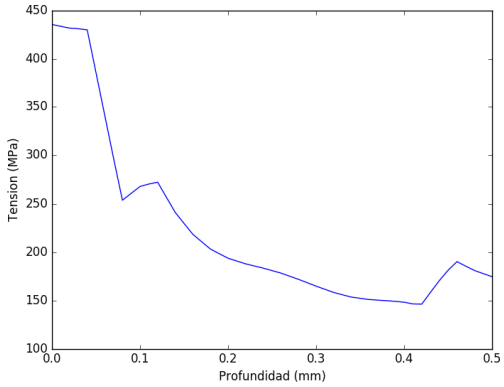
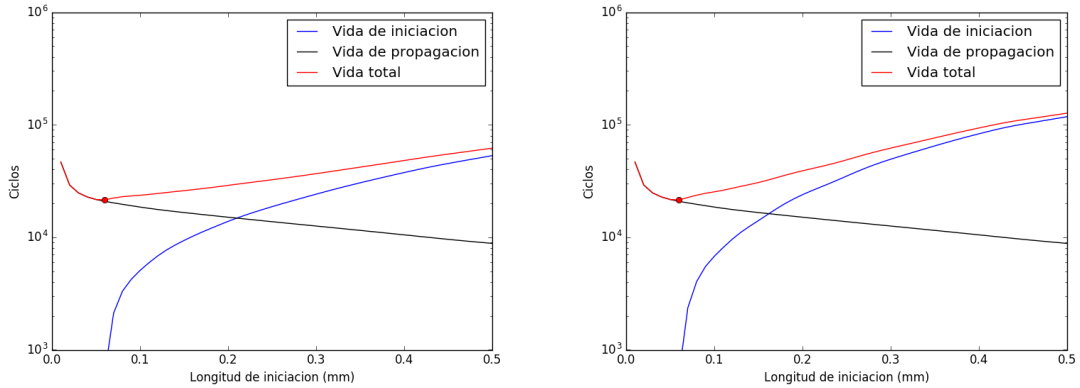


Figura 5.47 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 175|N = 971|Q = 3006$ con electroerosión.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.48 Curvas de vida estimada para la combinación de cargas $\sigma = 175|N = 971|Q = 3006$ con electroerosión.

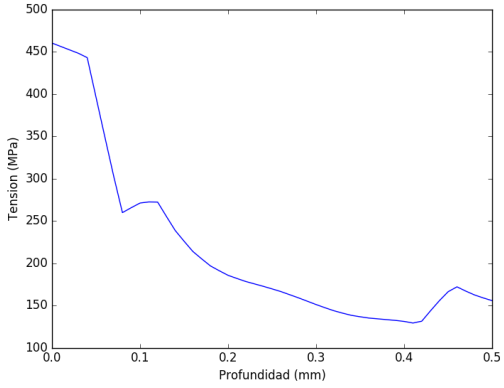
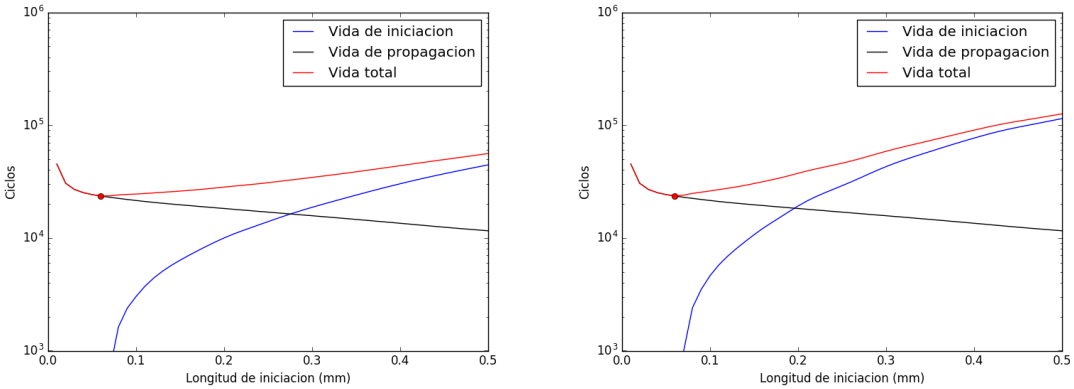


Figura 5.49 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 175|N = 971|Q = 5429$ con electroerosión.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.50 Curvas de vida estimada para la combinación de cargas $\sigma = 175|N = 971|Q = 5429$ con electroerosión.

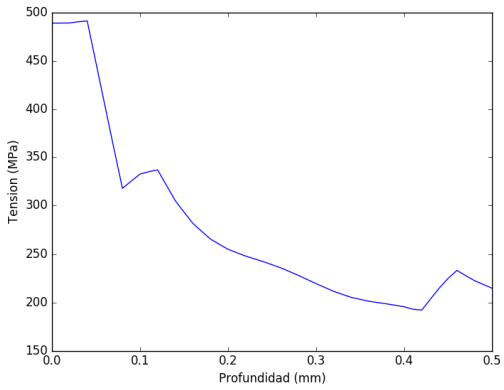
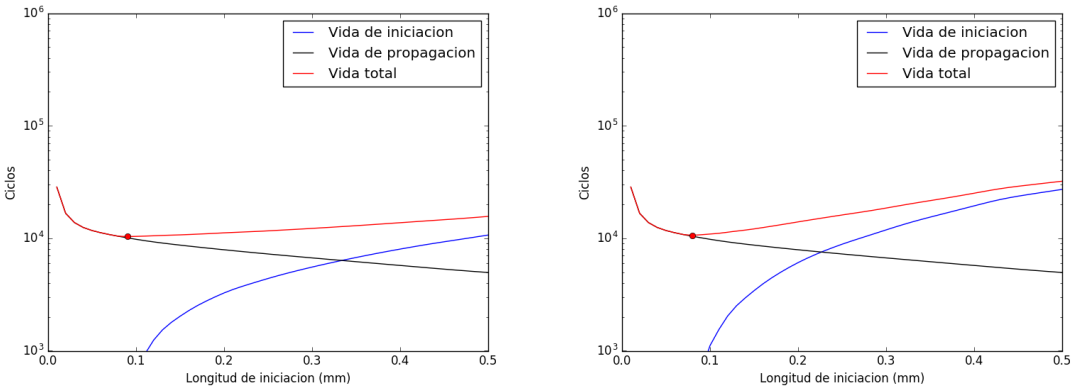


Figura 5.51 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 175|N = 2113|Q = 3006$ con electroerosión.



(a) Fatemi-Socie. **(b)** Smith-Watson-Topper.

Figura 5.52 Curvas de vida estimada para la combinación de cargas $\sigma = 175|N = 2113|Q = 3006$ con electroerosión.

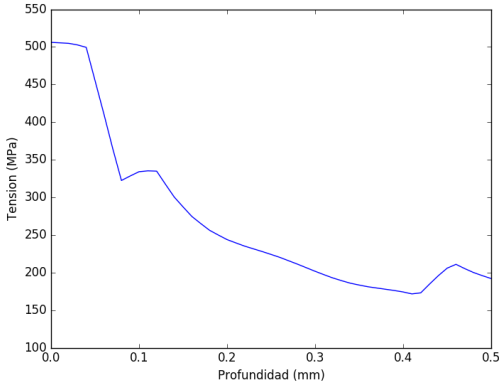
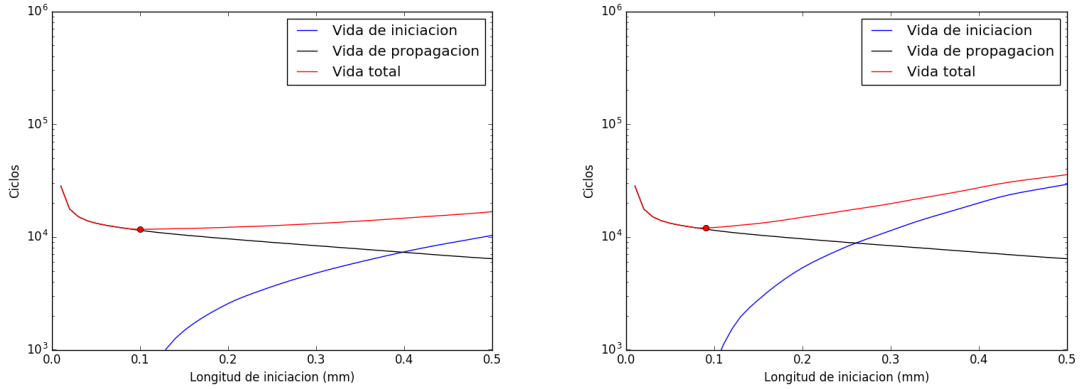


Figura 5.53 Evolución de la tensión σ_x para la combinación de cargas $\sigma = 175|N = 2113|Q = 5429$ con electroerosión.



(a) Fatemi-Socie.

(b) Smith-Watson-Topper.

Figura 5.54 Curvas de vida estimada para la combinación de cargas $\sigma = 175|N = 2113|Q = 5429$ con electroerosión.

6 Conclusiones

A lo largo de este trabajo se ha descrito la implementación de un modelo de vida a fatiga que combina las fases de iniciación y propagación utilizando una longitud de iniciación variable; además de evaluarse la calidad de los diferentes resultados obtenidos.

De entre los dos criterios de fatiga multiaxial utilizados; Fatemi-Socie y Smith-Watson-Topper, se ha podido constatar como en los casos analizados ha funcionado mejor, en términos generales, el criterio de Fatemi-Socie. Las diferencias entre ambos criterios en lo que a vida a fatiga se refiere son apreciables sobre todo para el grupo de ensayos con *shot peening*. Este grupo de ensayos se caracteriza por la existencia de grandes tensiones residuales; mientras que en los grupos de ensayo sin tratamiento superficial y con electroerosión, donde el comportamiento de las tensiones es similar, apenas aparecen diferencias. En lo que respecta a la duración de las fases de iniciación y propagación y a la longitud de iniciación sí aparecen diferencias relevantes si se comparan los dos criterios. Por otro lado, las diferencias entre ambos en la evolución de la longitud de las grietas en función del número de ciclos no son tan acusadas; aunque el criterio de Fatemi-Socie sigue obteniendo mejores resultados.

En cuanto a la influencia entre considerar el frente de grieta plano o elíptico utilizando el factor Φ calculado por Irwin; se ha comprobado como es mucho mayor en la fase de propagación que en la fase de iniciación, en la que las diferencias son apenas apreciables. Aparte, los resultados obtenidos utilizando frente de grieta elíptico en la fase de propagación son mucho mejores que aquellos obtenidos utilizando frente de grieta plano.

El modelo utilizado, como ventaja frente a otros, no necesita del ajuste de una gran cantidad de parámetros para obtener los resultados. Basta con elegir un criterio de fatiga multiaxial y seleccionar un tipo de frente de grieta. Se ha demostrado como la mejor combinación de estas variables es capaz de obtener buenos resultados en circunstancias muy distintas como las que se han analizado a lo largo del presente trabajo. Por un lado, se han utilizado experimentos en los que la combinación de cargas y los resultados de vida experimentales varían en gran medida. Por otro lado, los diferentes grupos de ensayo presentan características muy distintas con una evolución de tensiones muy diferente. Por ello, resulta interesante que los resultados obtenidos se ajusten a los resultados experimentales en todas las condiciones.

Apéndice A

Curvas de vida para Fatemi-Socie

A continuación, se muestran las gráficas individuales para cada experimento utilizando Fatemi-Socie e iniciación y propagación con frente de grieta elíptico, combinación con la que se han conseguido los mejores resultados con el modelo.

A.1 Sin tratamiento superficial

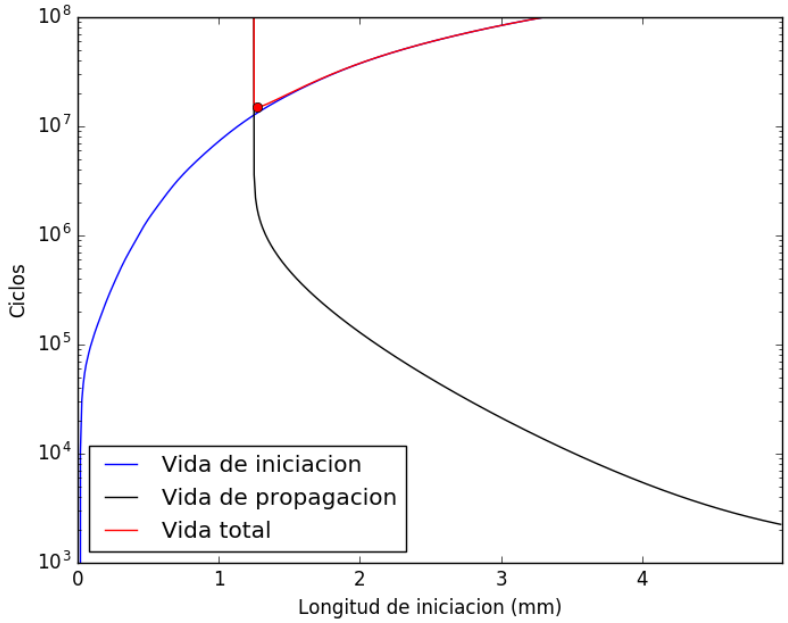


Figura A.1 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 70|N = 971|Q = 6629$ sin tratamiento superficial.

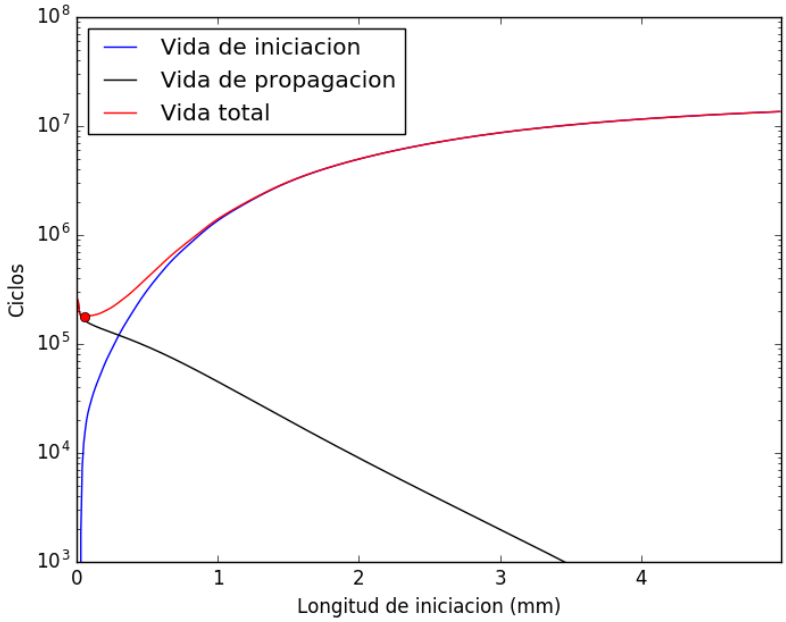


Figura A.2 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 971|Q = 5429$ sin tratamiento superficial.

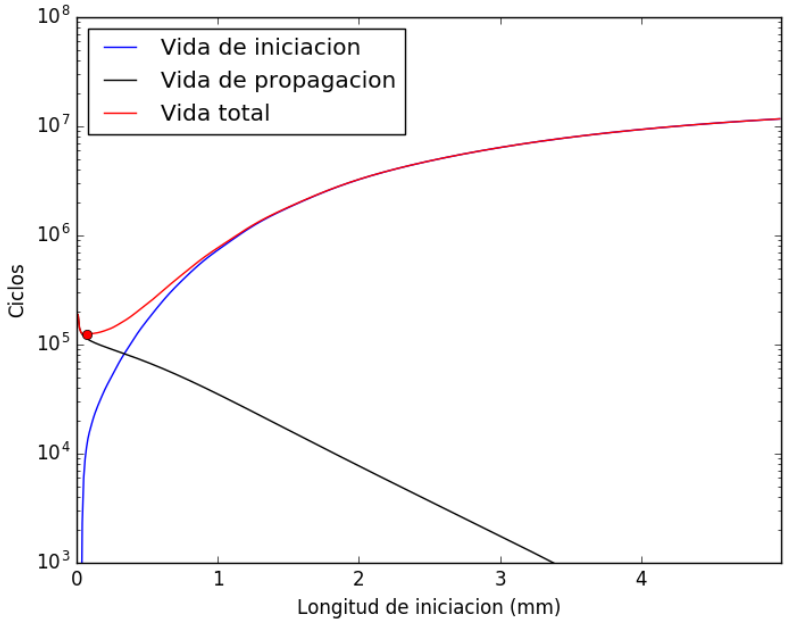


Figura A.3 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 1257|Q = 5429$ sin tratamiento superficial.

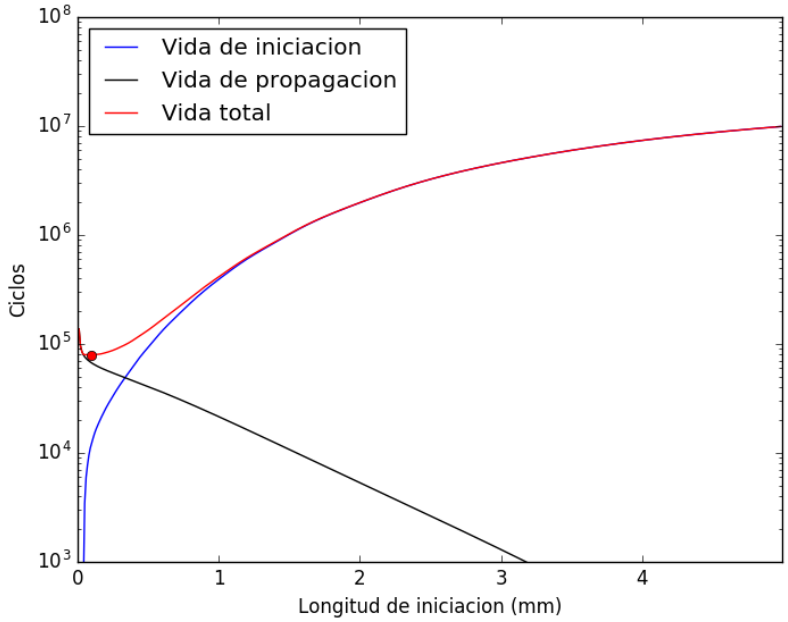


Figura A.4 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 1543|Q = 4217$ sin tratamiento superficial.

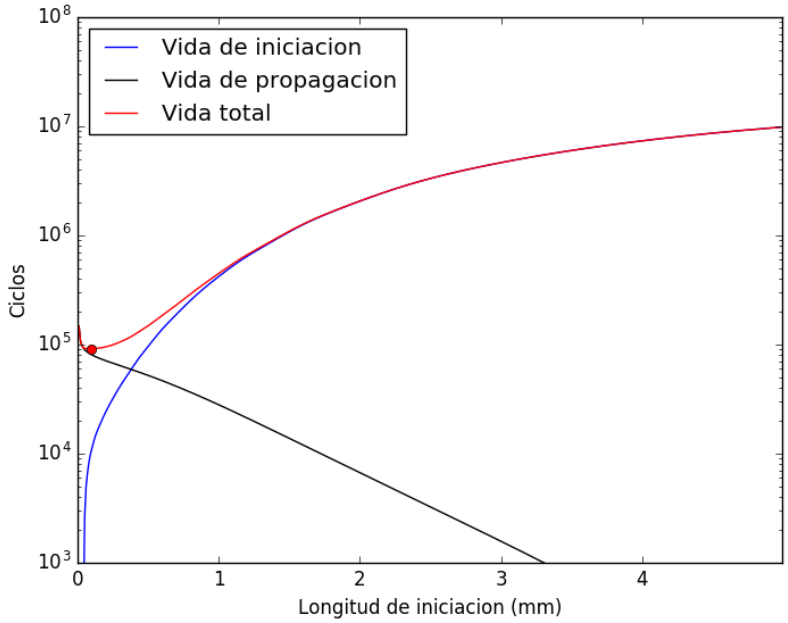


Figura A.5 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 1543|Q = 5429$ sin tratamiento superficial.

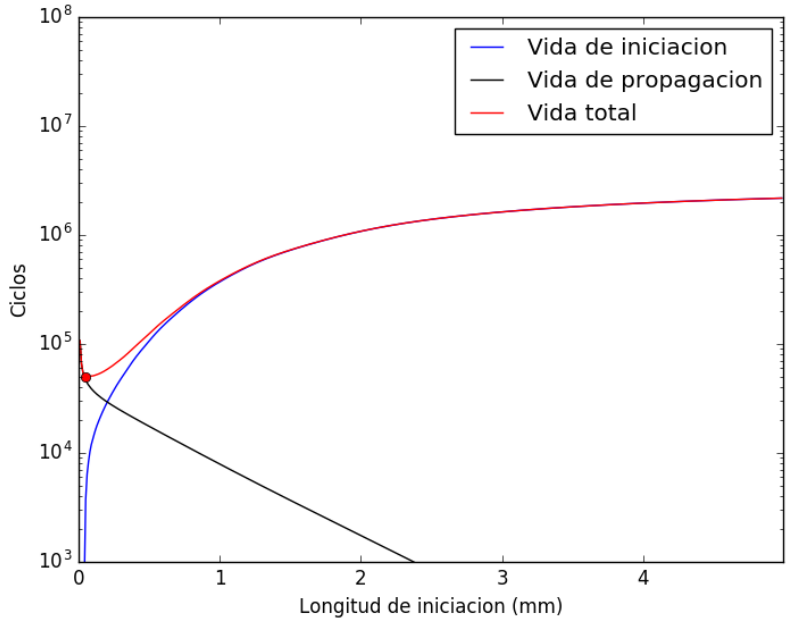


Figura A.6 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 971|Q = 3006$ sin tratamiento superficial.

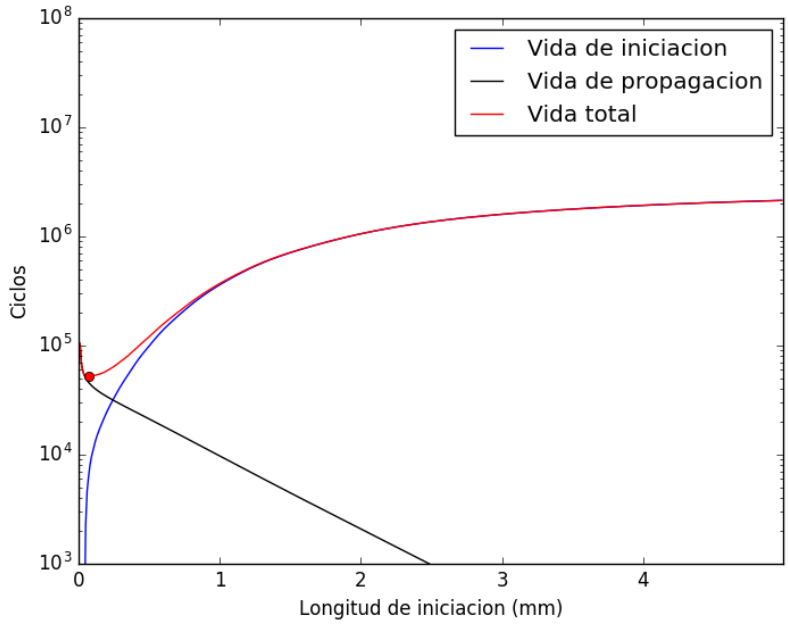


Figura A.7 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 971|Q = 4217$ sin tratamiento superficial.

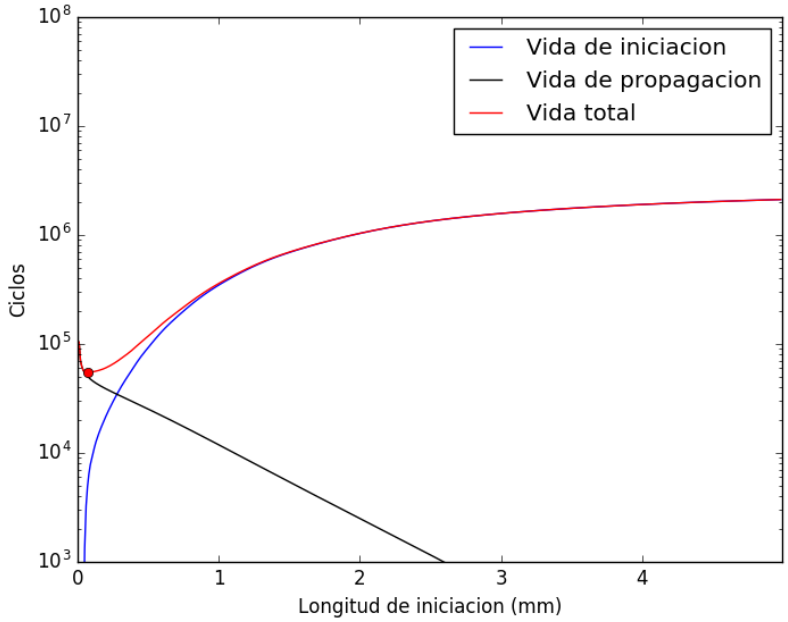


Figura A.8 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 971|Q = 5429$ sin tratamiento superficial.

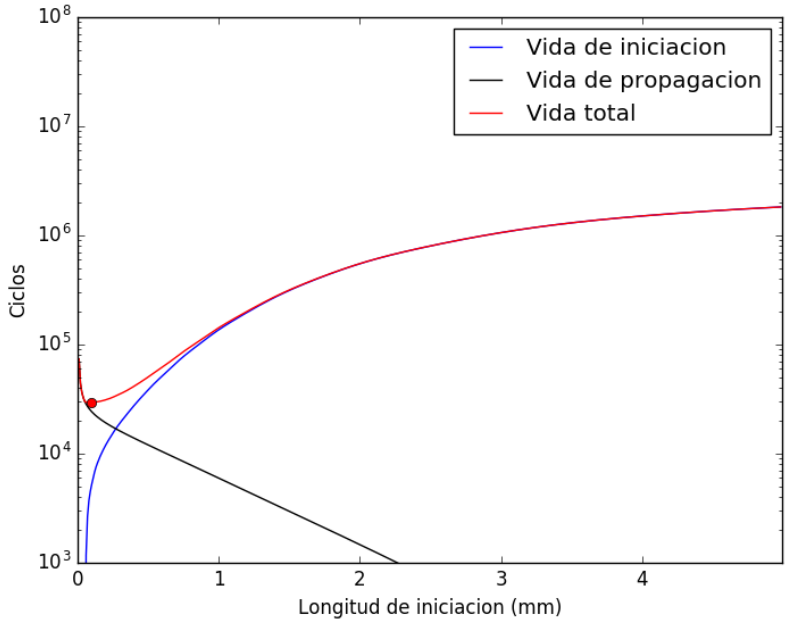


Figura A.9 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 1543|Q = 3006$ sin tratamiento superficial.

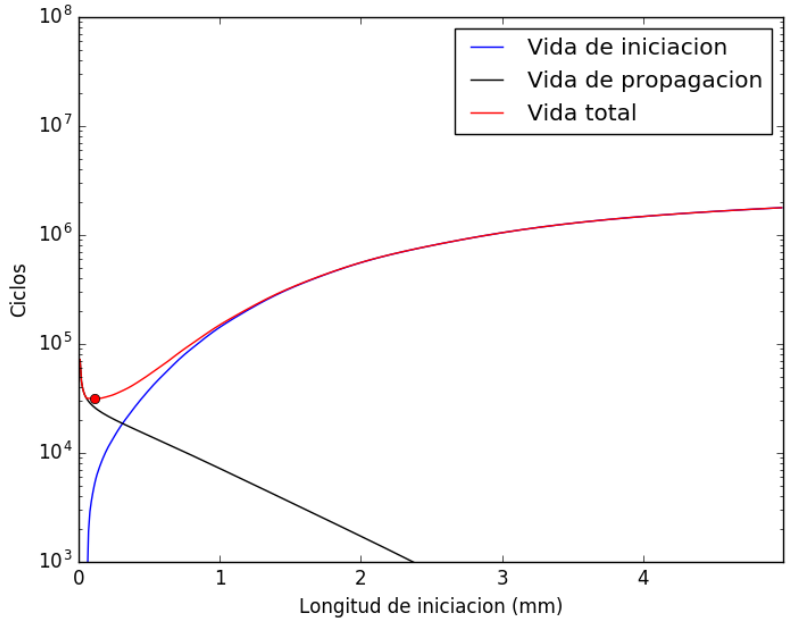


Figura A.10 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 1543|Q = 4217$ sin tratamiento superficial.

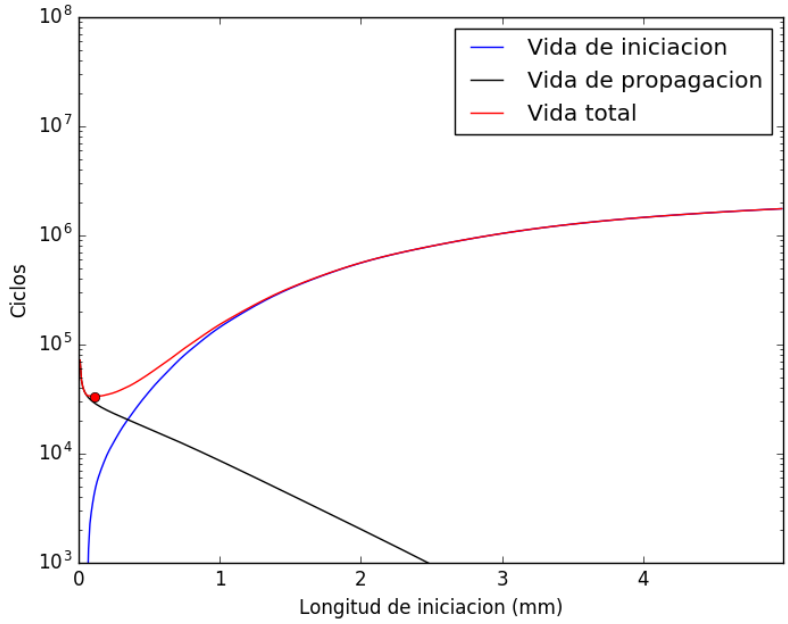


Figura A.11 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 1543|Q = 5429$ sin tratamiento superficial.

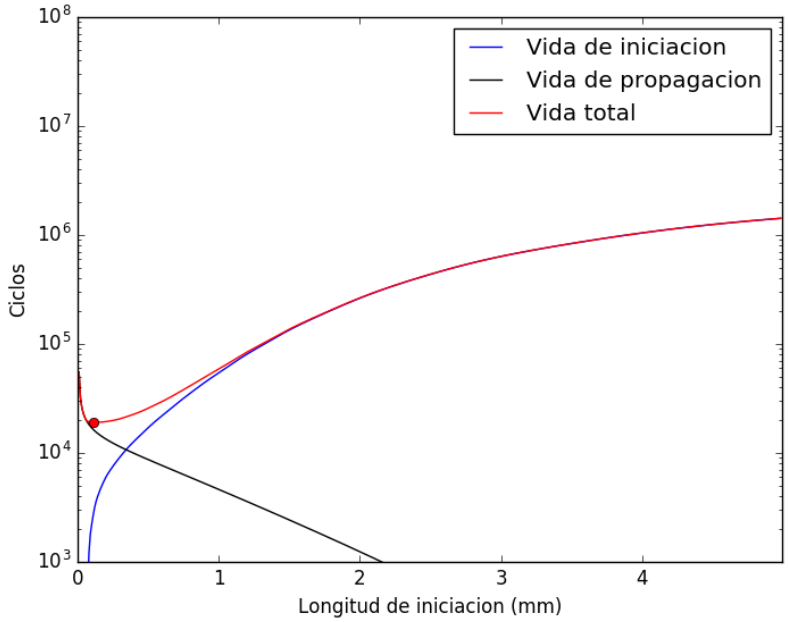


Figura A.12 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 3006$ sin tratamiento superficial.

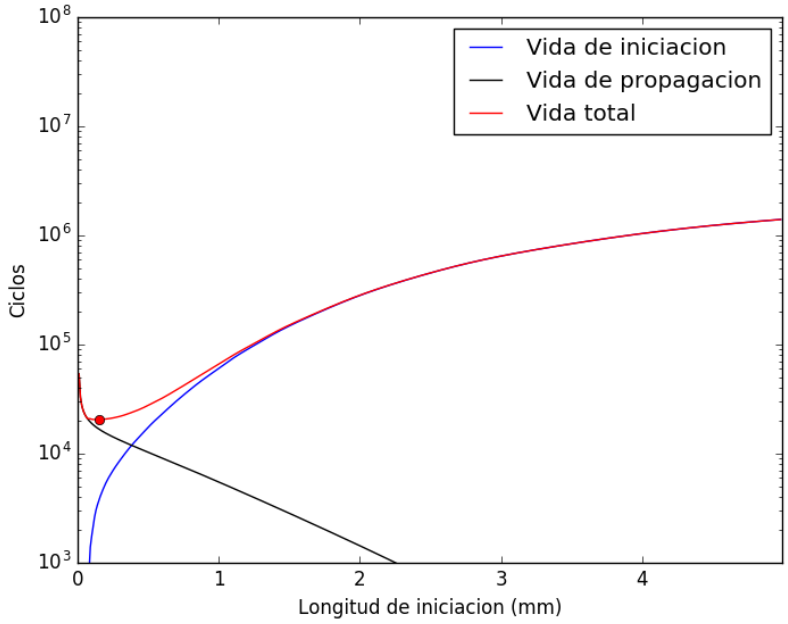


Figura A.13 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 4217$ sin tratamiento superficial.

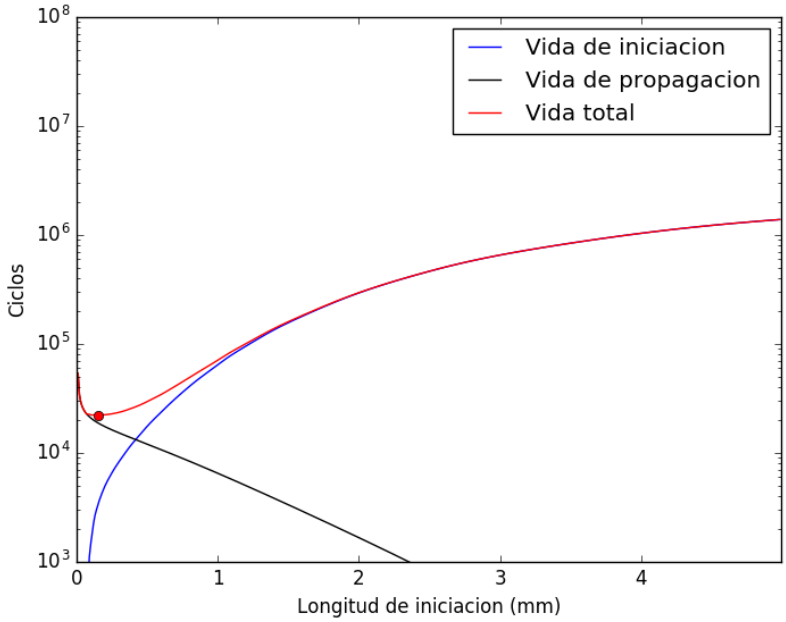


Figura A.14 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 5429$ sin tratamiento superficial.

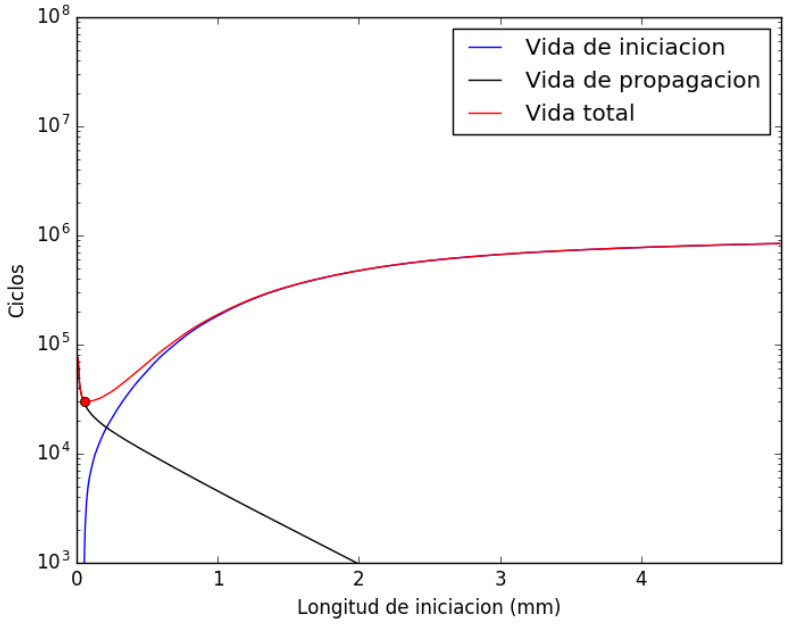


Figura A.15 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 3006$ sin tratamiento superficial.

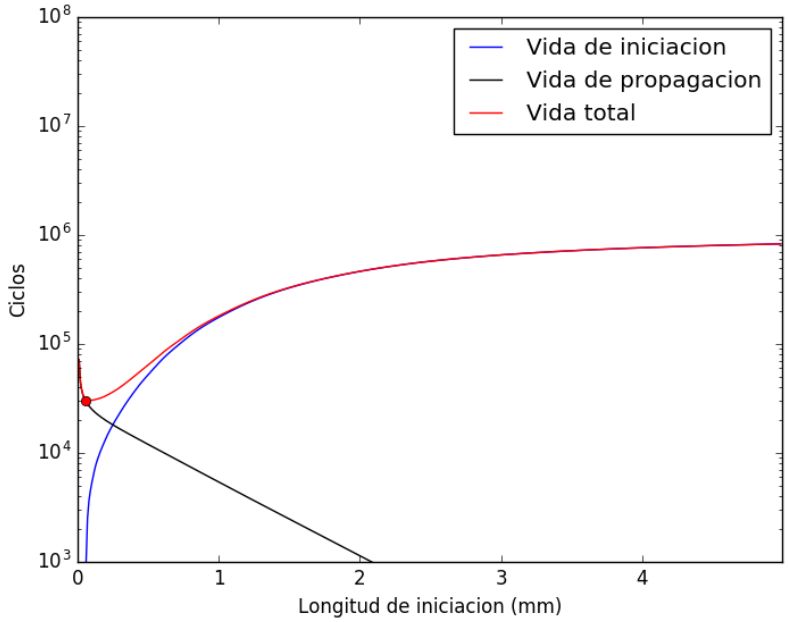


Figura A.16 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 4217$ sin tratamiento superficial.

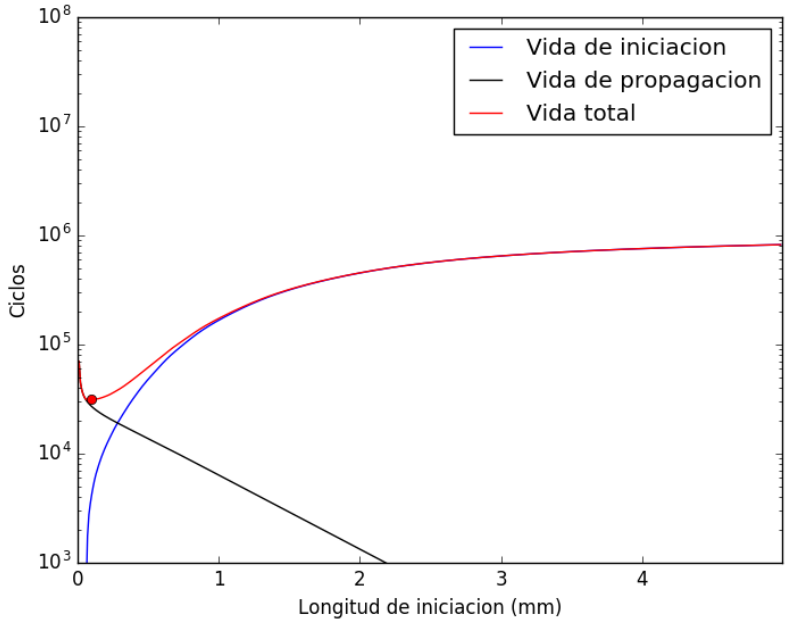


Figura A.17 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 5429$ sin tratamiento superficial.

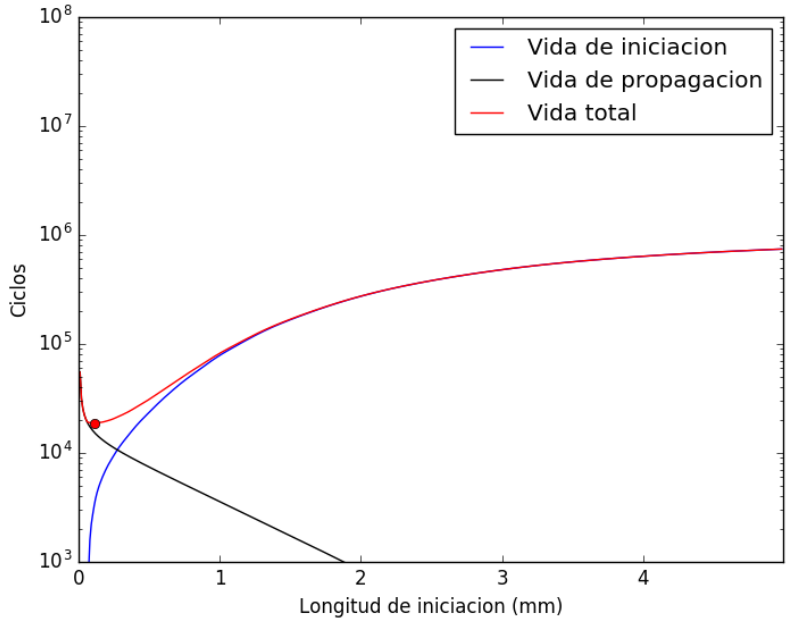


Figura A.18 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 1543|Q = 3006$ sin tratamiento superficial.

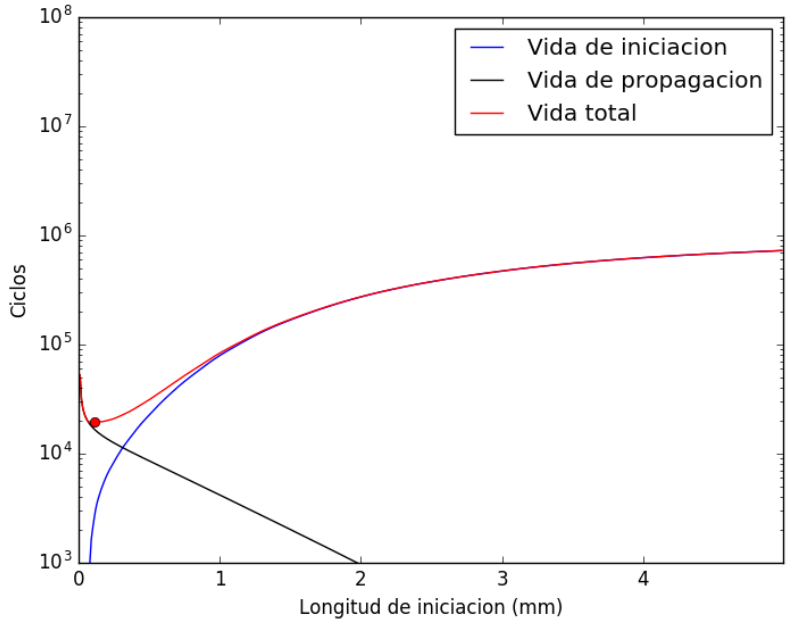


Figura A.19 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 1543|Q = 4217$ sin tratamiento superficial.

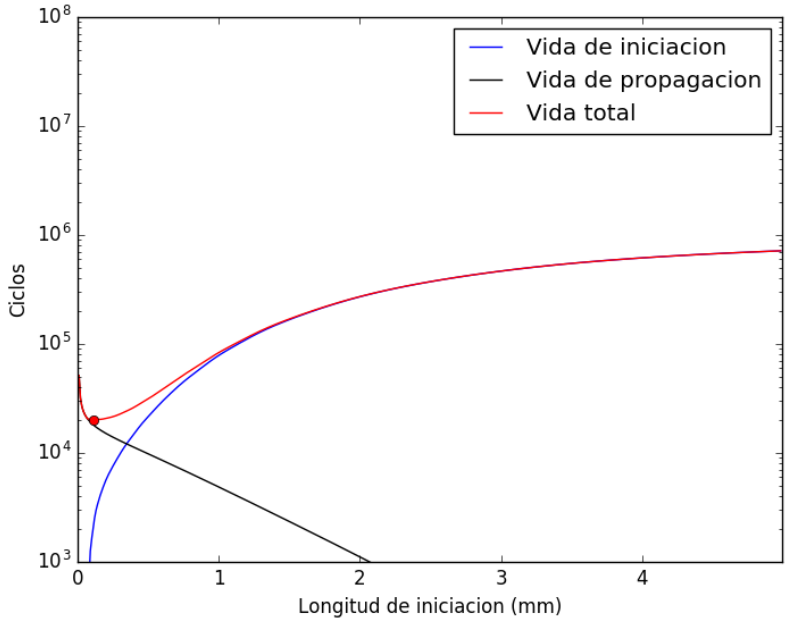


Figura A.20 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 1543|Q = 5429$ sin tratamiento superficial.

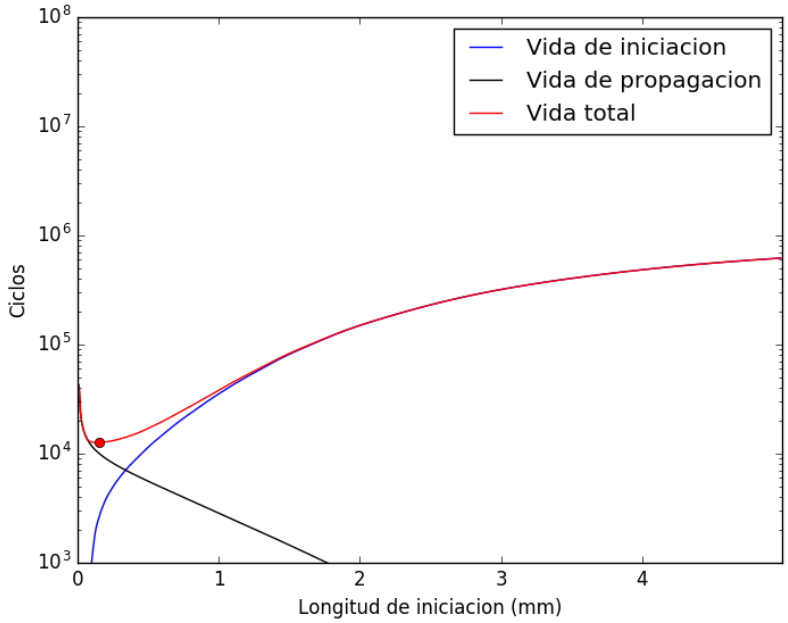


Figura A.21 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 3006$ sin tratamiento superficial.

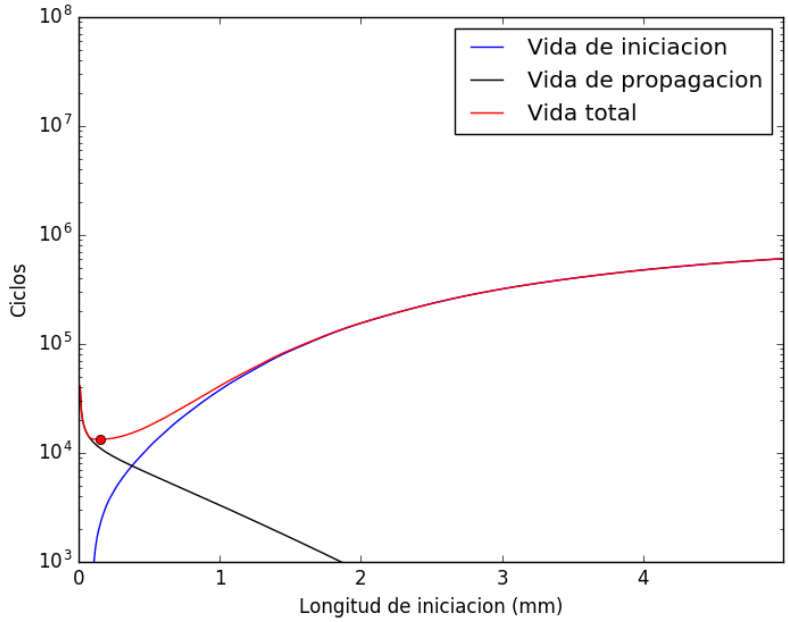


Figura A.22 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 4217$ sin tratamiento superficial.

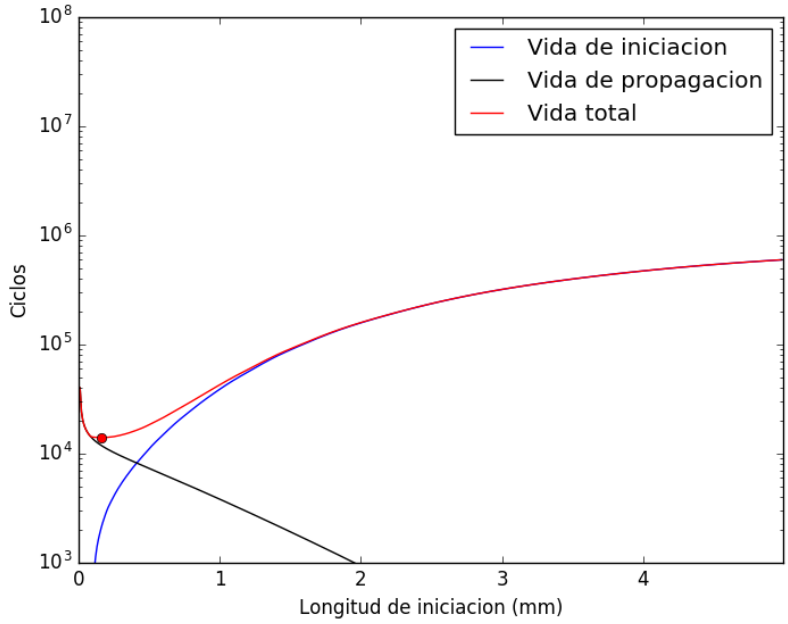


Figura A.23 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 5429$ sin tratamiento superficial.

A.2 Con *shot peening*

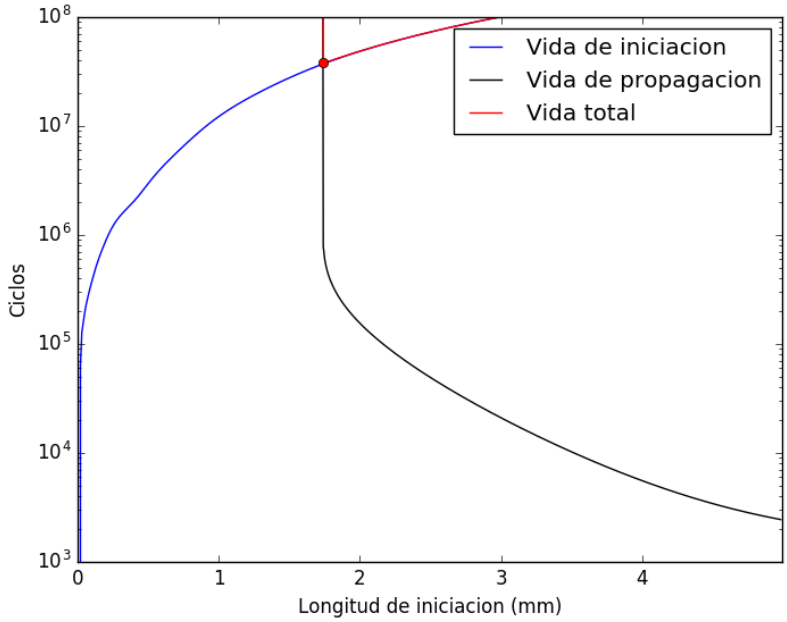


Figura A.24 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 70|N = 971|Q = 6629$ con *shot peening*.

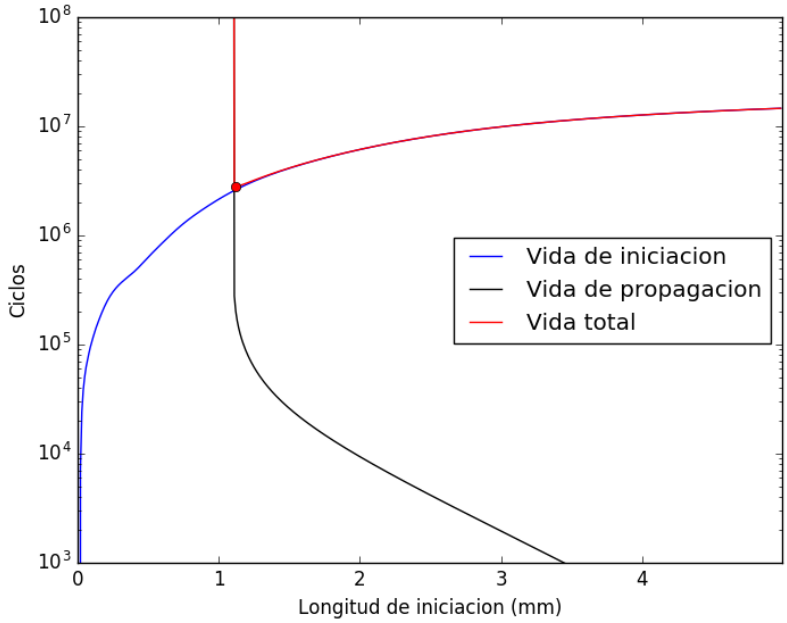


Figura A.25 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 971|Q = 5429$ con *shot peening*.

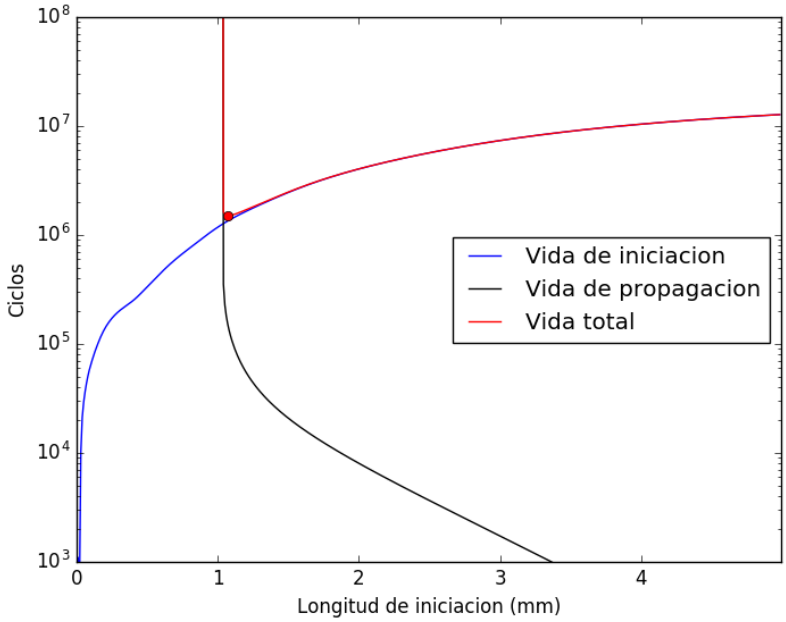


Figura A.26 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 1257|Q = 5429$ con *shot peening*.

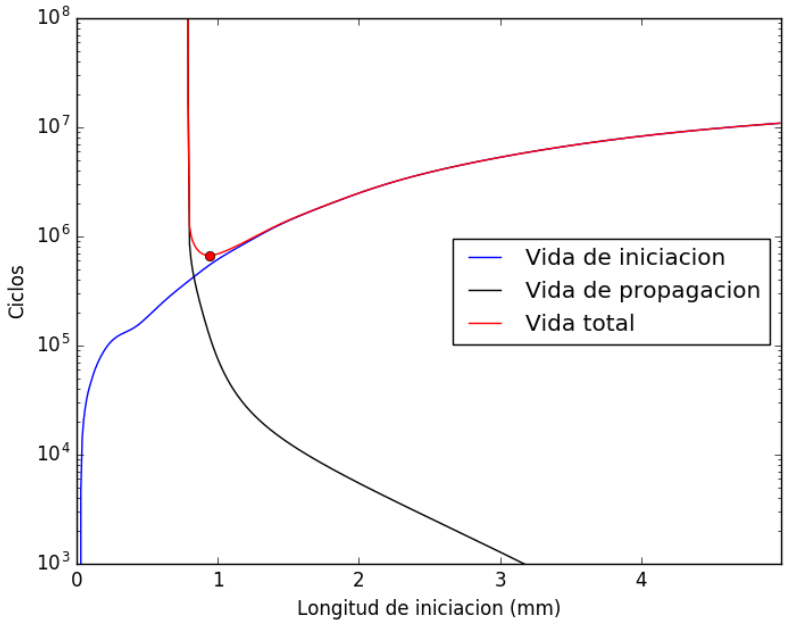


Figura A.27 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 1543|Q = 4217$ con *shot peening*.

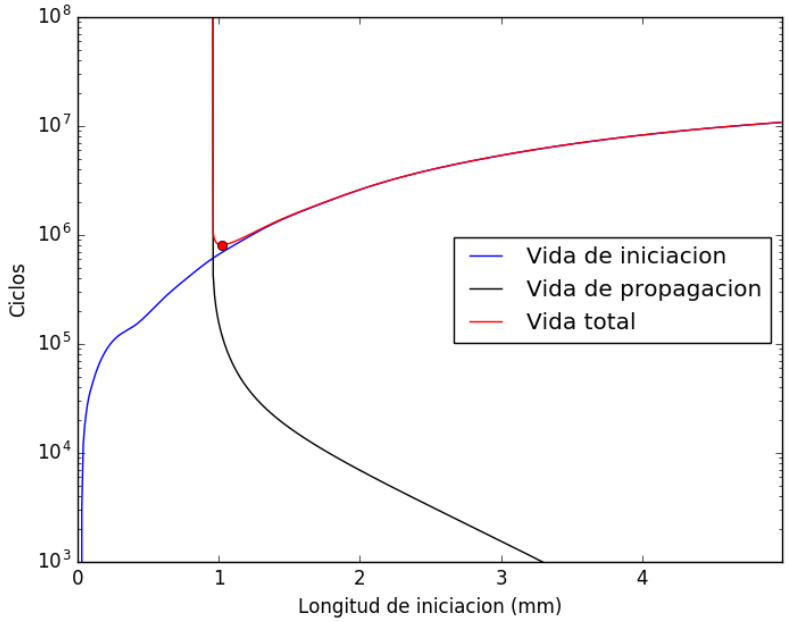


Figura A.28 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 110|N = 1543|Q = 5429$ con *shot peening*.

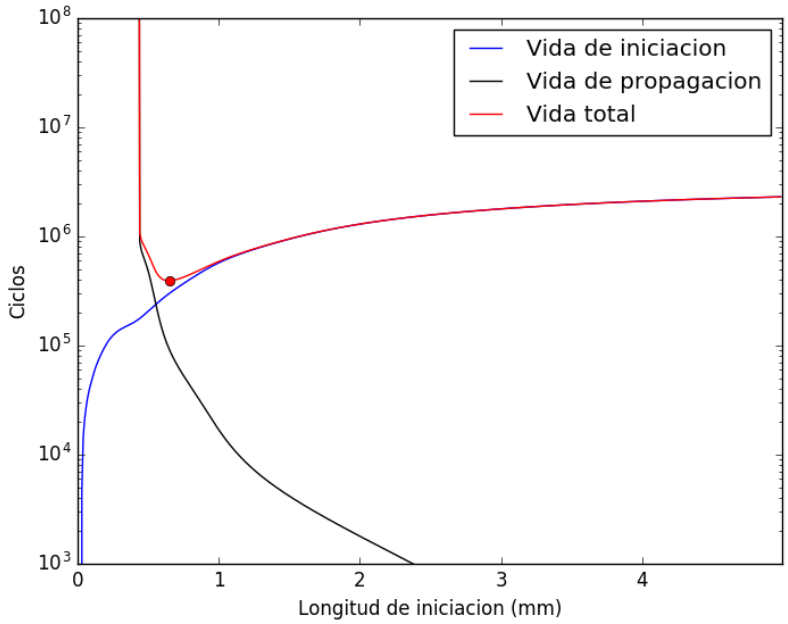


Figura A.29 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 971|Q = 3006$ con *shot peening*.

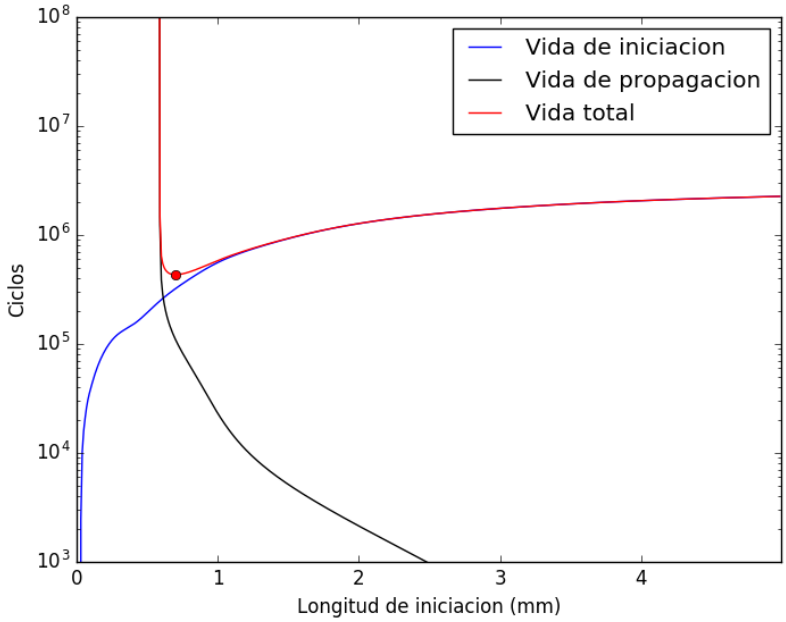


Figura A.30 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 971|Q = 4217$ con *shot peening*.

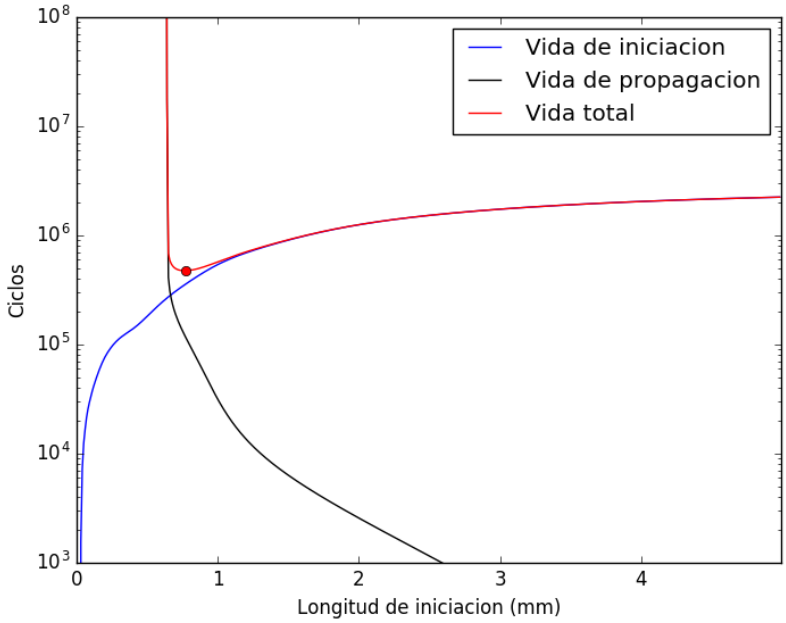


Figura A.31 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 971|Q = 5429$ con *shot peening*.

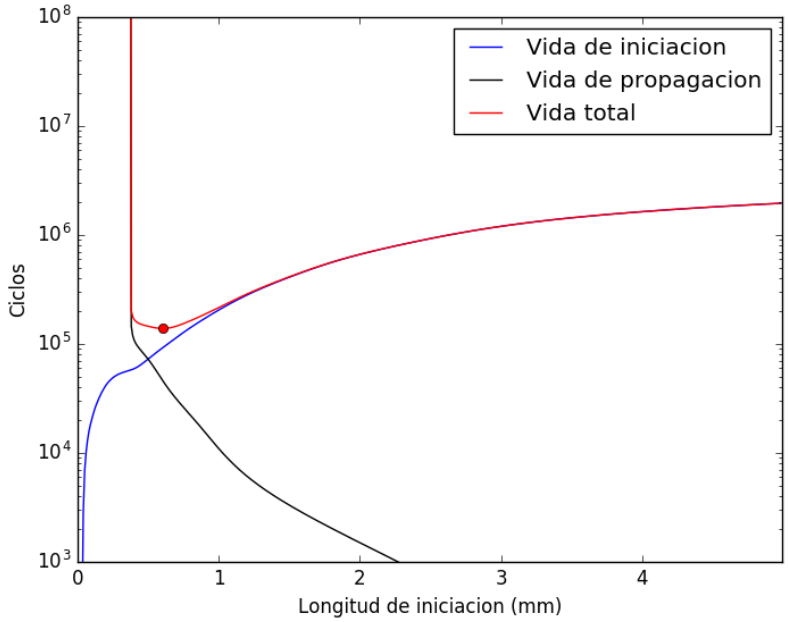


Figura A.32 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 1543|Q = 3006$ con *shot peening*.

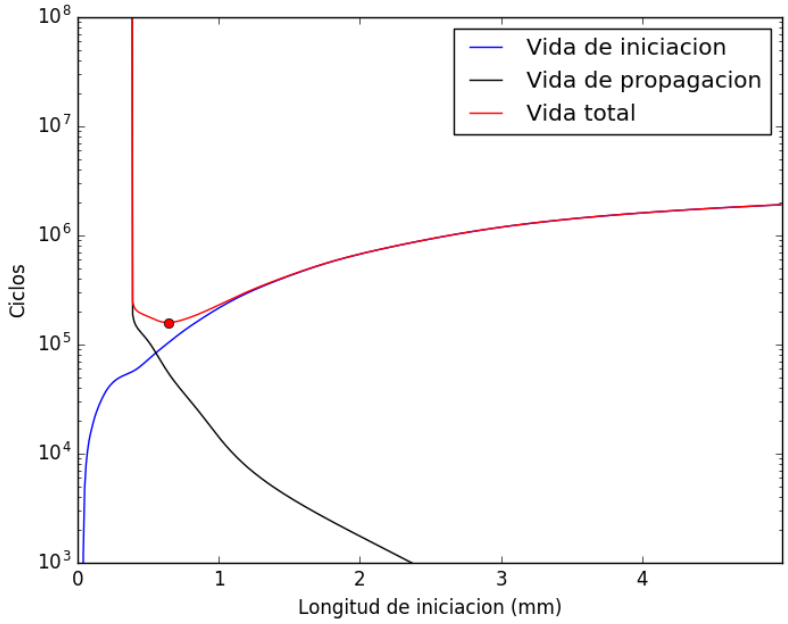


Figura A.33 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 1543|Q = 4217$ con *shot peening*.

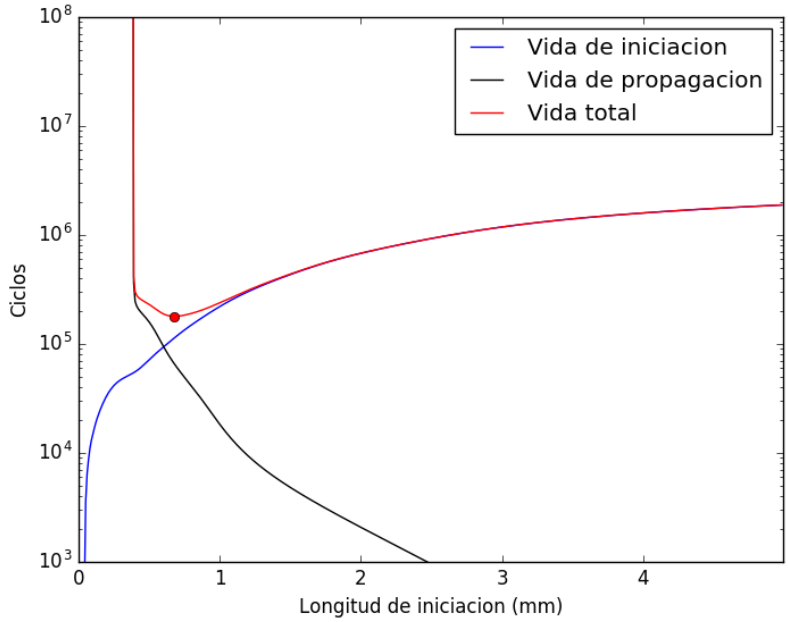


Figura A.34 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 1543|Q = 5429$ con *shot peening*.

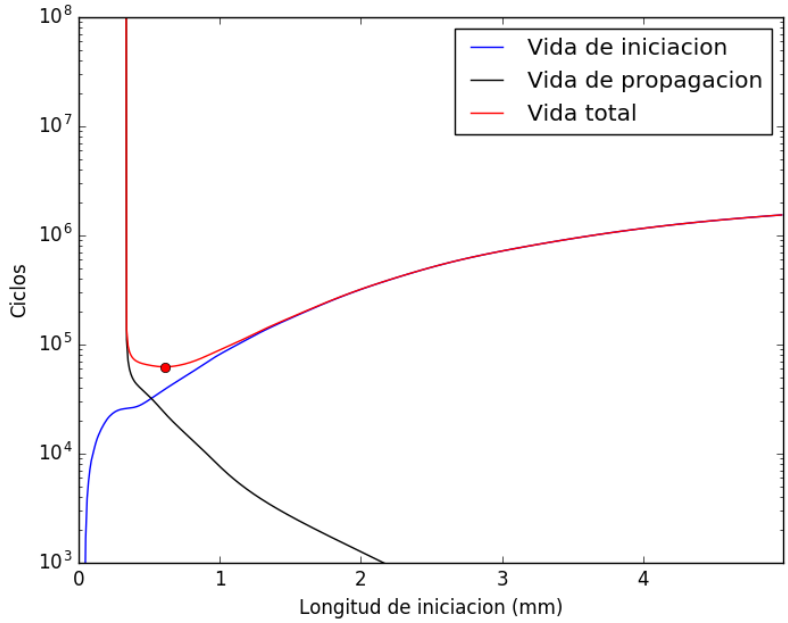


Figura A.35 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 3006$ con *shot peening*.

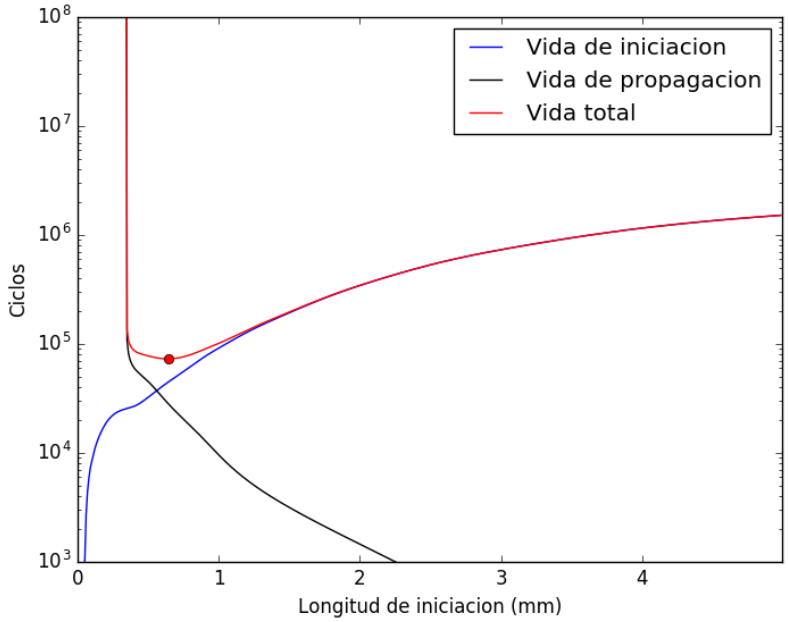


Figura A.36 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 4217$ con *shot peening*.

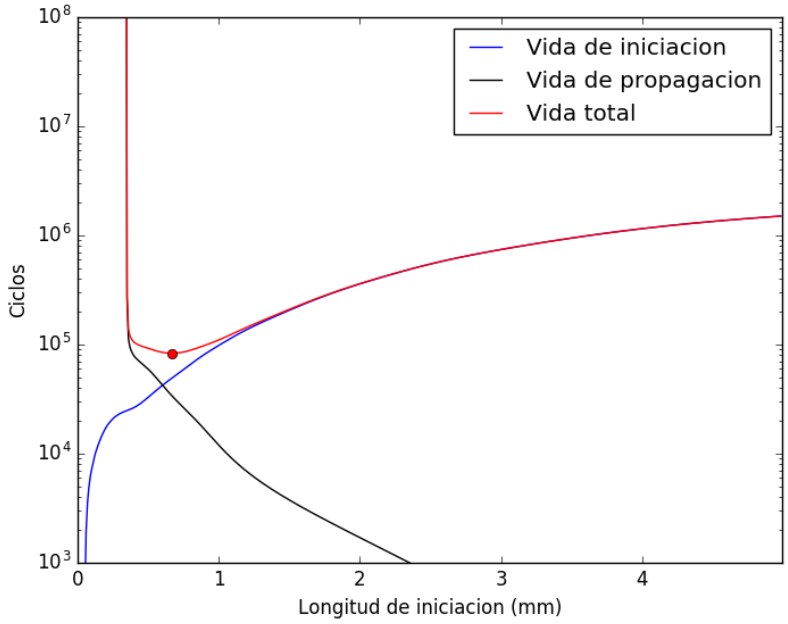


Figura A.37 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 5429$ con *shot peening*.

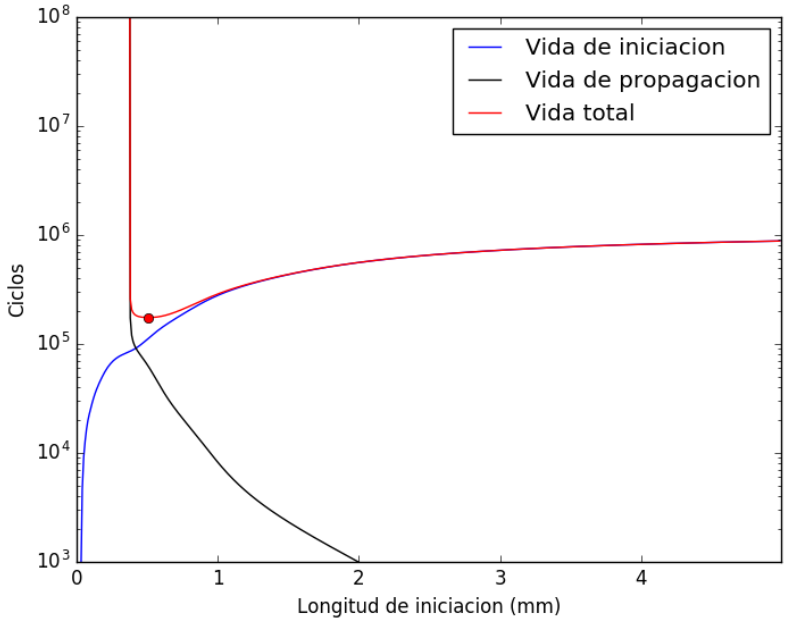


Figura A.38 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 3006$ con *shot peening*.

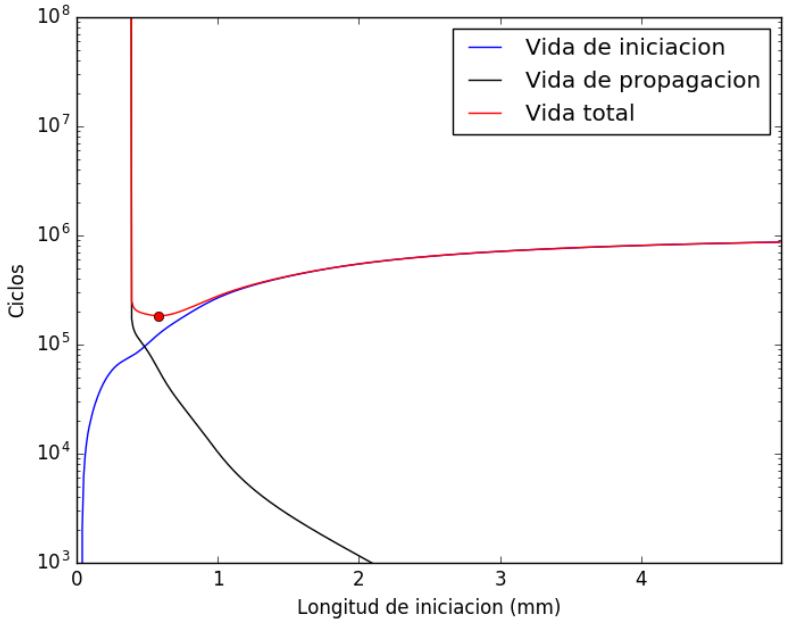


Figura A.39 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 4217$ con *shot peening*.

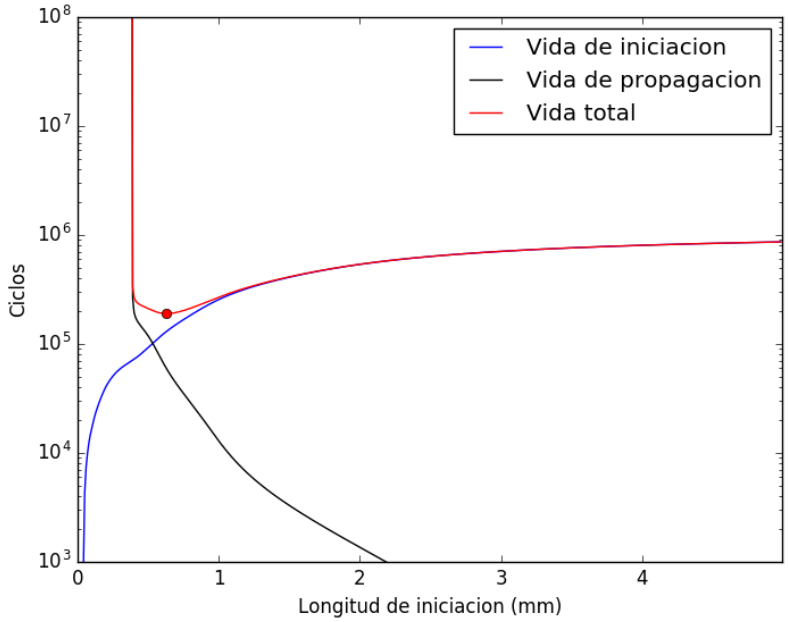


Figura A.40 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 5429$ con *shot peening*.

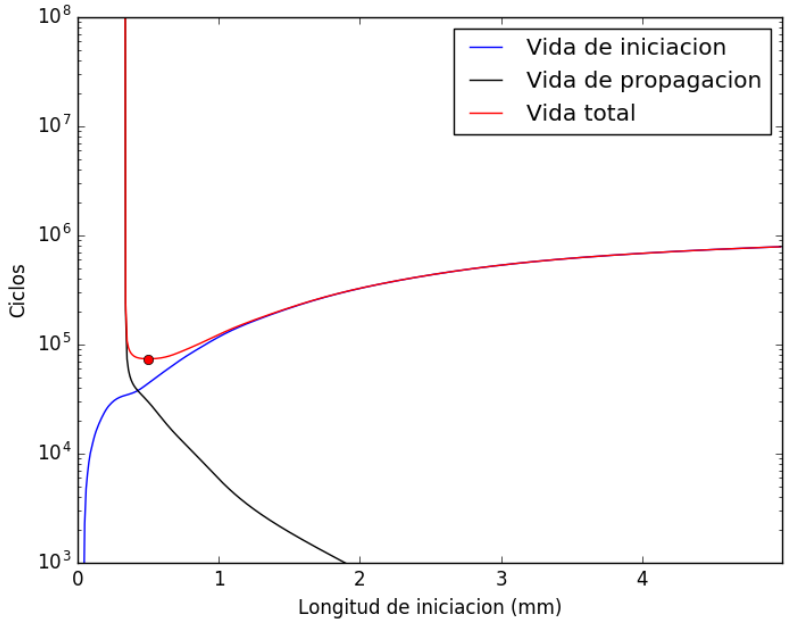


Figura A.41 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 1543|Q = 3006$ con *shot peening*.

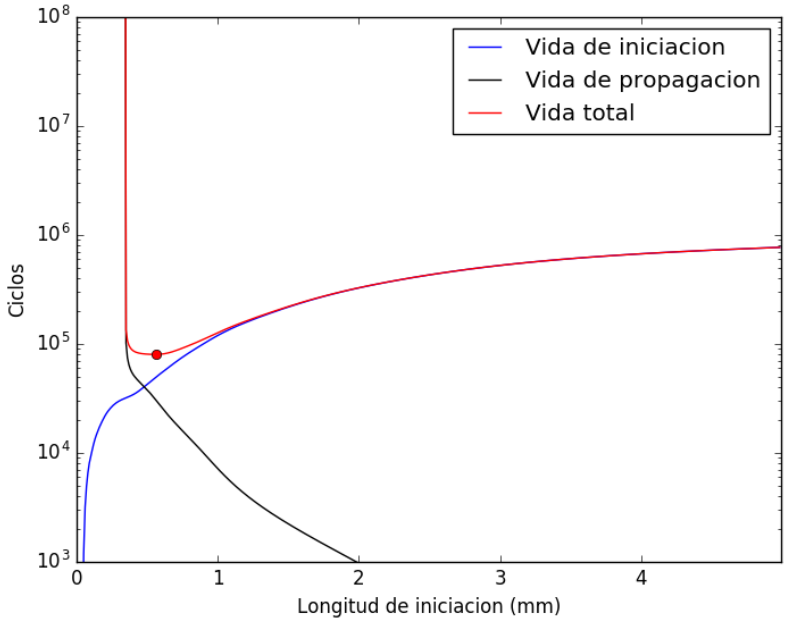


Figura A.42 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 1543|Q = 4217$ con *shot peening*.

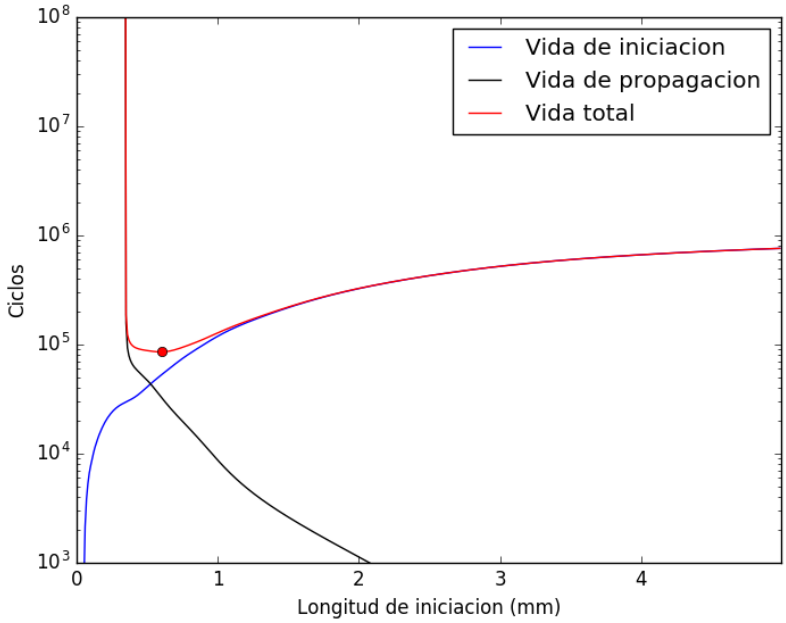


Figura A.43 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 1543|Q = 5429$ con *shot peening*.

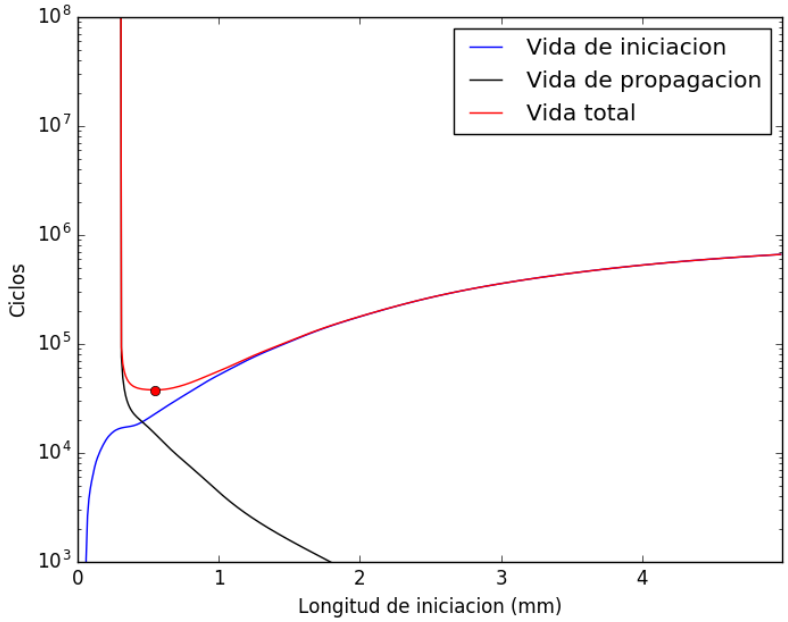


Figura A.44 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 3006$ con *shot peening*.

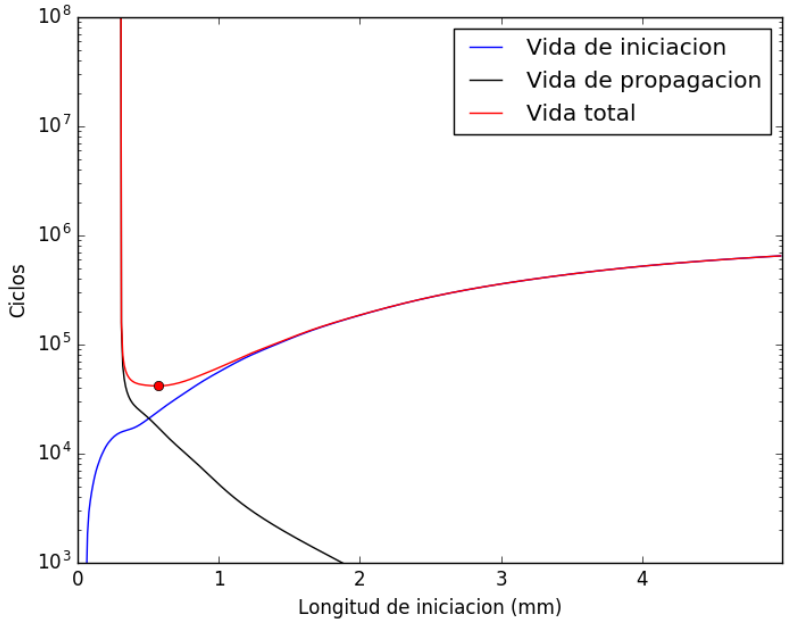


Figura A.45 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 4217$ con *shot peening*.

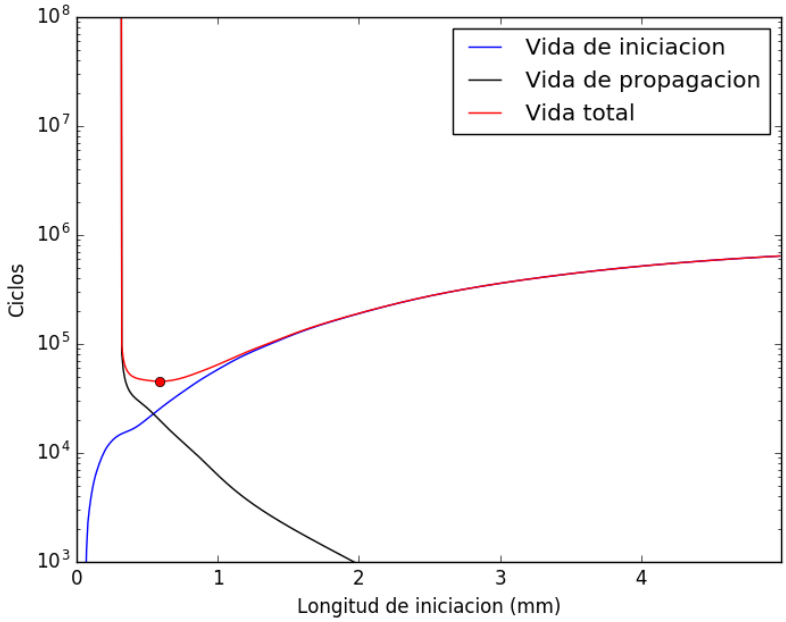


Figura A.46 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 5429$ con *shot peening*.

A.3 Con electroerosión

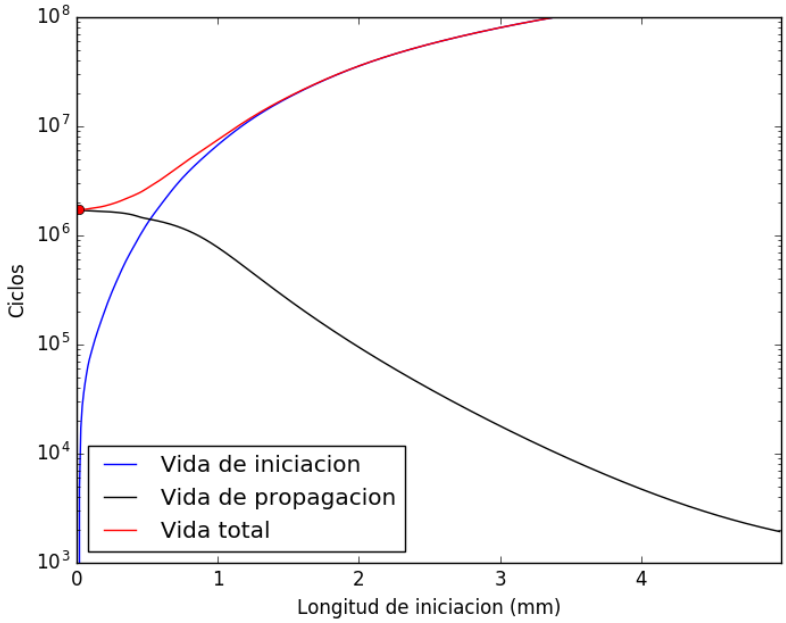


Figura A.47 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 70|N = 971|Q = 6629$ con electroerosión.

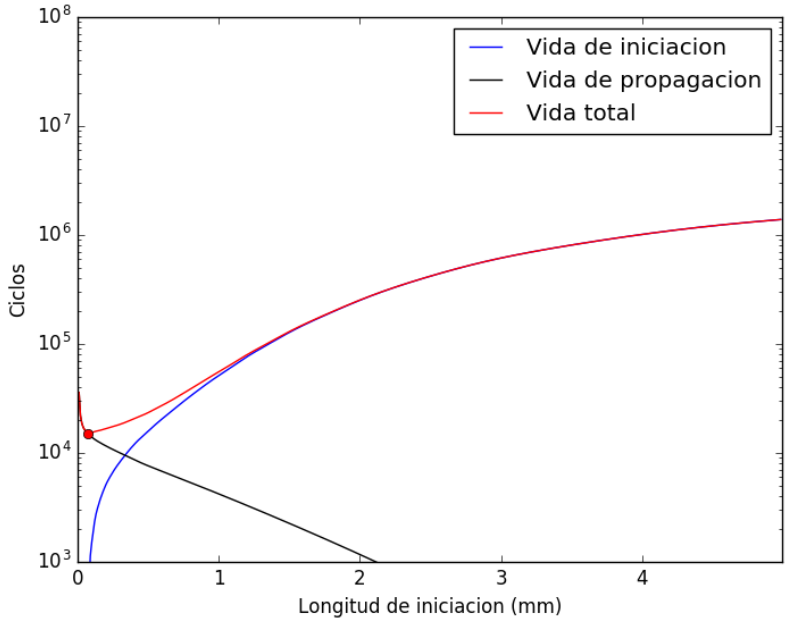


Figura A.48 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 3006$ con electroerosión.

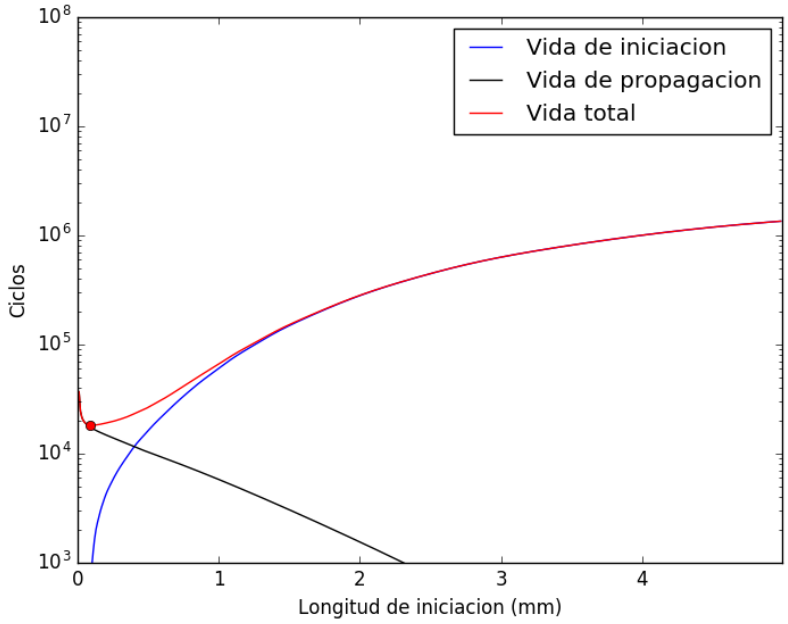


Figura A.49 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 150|N = 2113|Q = 5429$ con electroerosión.

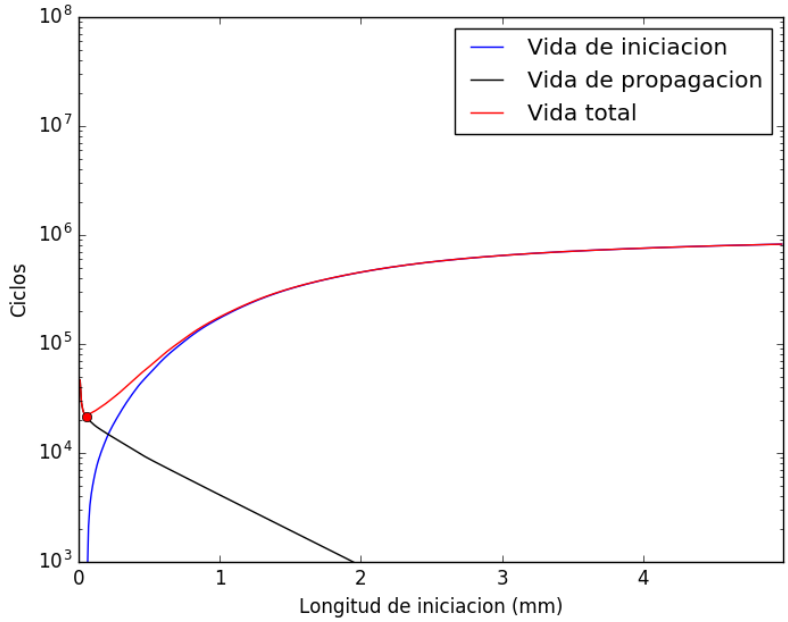


Figura A.50 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 3006$ con electroerosión.

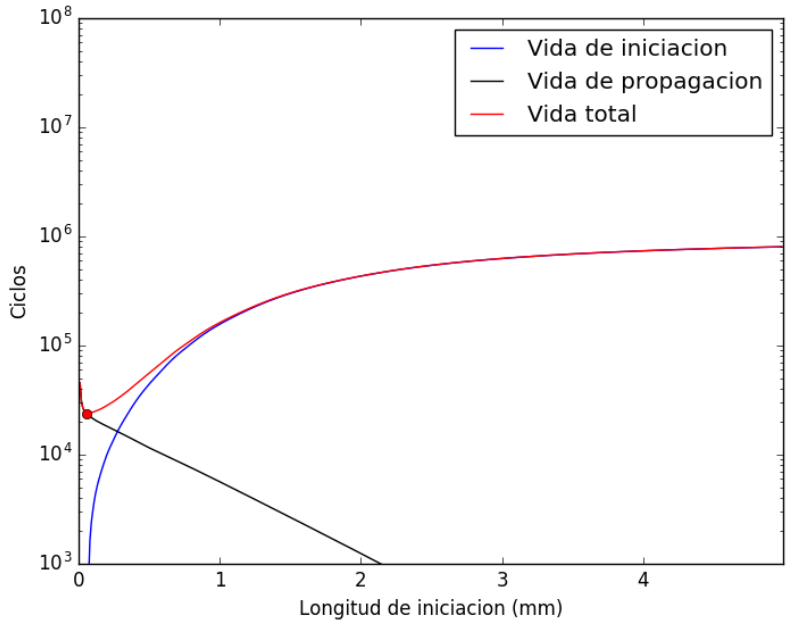


Figura A.51 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 971|Q = 5429$ con electroerosión.

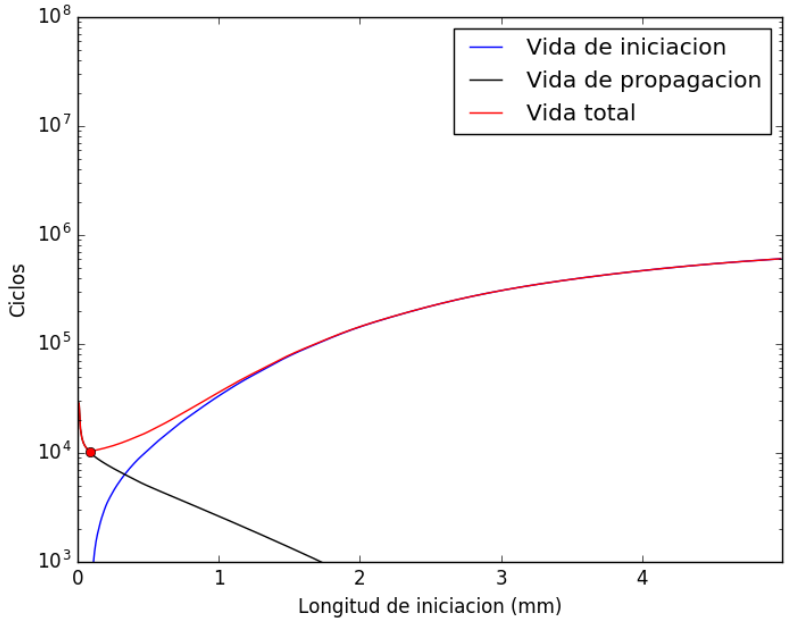


Figura A.52 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 3006$ con electroerosión.

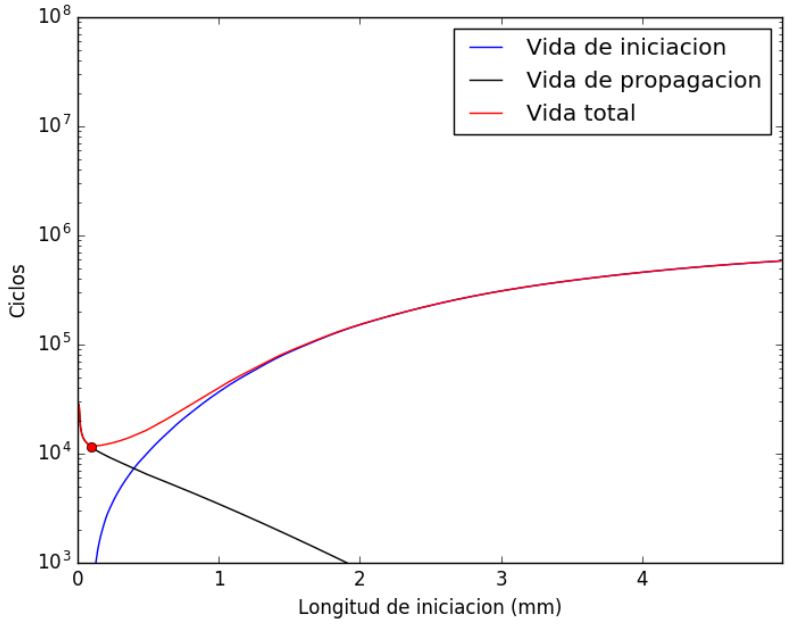


Figura A.53 Evolución de la vida de iniciación, propagación y total para la combinación $\sigma = 175|N = 2113|Q = 5429$ con electroerosión.

Apéndice B
Apuntes sobre el código

En este apéndice se describen ciertos detalles para facilitar la ejecución y edición del código implementado.

B.1 Ejecución del código

Al final de cada *script* se han añadido las lineas necesarias para poder ejecutar el código simplemente ejecutando el archivo. El *script propagacion.py* solo incluye funciones utilizadas por el resto de módulos, por lo que al ejecutarlo no sucede nada tal y como puede comprobarse en el Código B.1.

Código B.1 Ejecución del *script propagacion.py*.

```
if __name__ == "__main__":

    pass
```

El *script iniciacion.py* tiene implementadas las funciones necesarias para realizar las curvas de iniciación de un material para un criterio de fatiga multiaxial. En el Código B.2 se muestra un ejemplo de como debe ejecutarse para conseguir las curvas de iniciación de Fatemi-Socie. Entre las variables que puede interesar modificar a la hora de obtener las curvas de iniciación se encuentran "n_sigma", "sigma_max", "sigma_min", "n_a" y "ex"; dentro de la función *curvas_iniciacion()*.

Código B.2 Ejecución del *script iniciacion.py*.

```
if __name__ == "__main__":

    curvas=curvas_iniciacion(par = 'FS', da=0.2e-5, W = 10e-3, MAT = 0)
```

En el Código B.3 se muestra un ejemplo de ejecución del *script principal.py* para obtener la estimación de vida. Se ha implementado de forma que se puedan realizar todas las estimaciones de un grupo de ensayos al completo. Las variables "tracc", "comp" y "exp" se utilizan para formar los nombres de los archivos con los resultados experimentales, de forma que se pueden modificar aquí para ejecutar archivos con cualquier nombre. Aquí es muy importante destacar que dentro de los archivos con los resultados, además de los nombres de las gráficas obtenidas con las curvas, se utiliza un identificador del experimento que se guarda dentro de la variable "exp_id" al principio de la función *principal()*. Esta variable es creada a partir de la variable "exp_max" por lo que es necesario modificarla si el nombre del archivo sufre alguna variación.

Código B.3 Ejecución del *script principal.py*.

```
if __name__ == "__main__":

    par  = 'FS'
    W    = 10e-3
    MAT  = 0
    tracc = 'TENSOR_TRACCION_'
    comp  = 'TENSOR_COMPRESION_'
    exp   = ['6629_971_70', '5429_971_110',
             '5429_1257_110', '4217_1543_110',
             '5429_1543_110', '3006_971_150',
             '4217_971_150', '5429_971_150',
             '3006_1543_150', '4217_1543_150',
             '5429_1543_150', '3006_2113_150',
             '4217_2113_150', '5429_2113_150',
             '3006_971_175', '4217_971_175',
             '5429_971_175', '3006_1543_175',
             '4217_1543_175', '5429_1543_175',
             '3006_2113_175', '4217_2113_175', '5429_2113_175']

    for i in exp:
        exp_max = tracc + i
        exp_min = comp + i
        principal(par, W, MAT, exp_max, exp_min)
```

Por último, en el Código B.4 se muestra un ejemplo de ejecución del *script graficas.py*. Como se puede comprobar, en primer lugar se definen variables con todos los resultados experimentales de los distintos grupos de ensayo en tres diccionarios diferentes. Para ejecutar las funciones *grafs_globales()* y *error()* para un solo grupo de ensayos se les pasa directamente el diccionario junto con el nombre del criterio de fatiga multiaxial que se quiere utilizar.Si se quiere emplear la función *grafs_globales()* para más de un grupo de ensayos, se deberá ejecutar individualmente en cada carpeta donde se encuentren los resultados, cambiando el color en las variables "pts", "pts_inic" y "pts_long". Para el último grupo de ensayos hay que descomentar las líneas de código con la leyenda, asegurándose de que los colores de la figura 0 coinciden con los utilizados. Para utilizar la función *error()* para más de un grupo de ensayos aparte de las dos variables anteriores se le pasa otros dos nuevos diccionarios con los resultados de los grupos de ensayo. Conviene tener cuidado al ejecutar esta función, ya que es necesario que los archivos con los resultados se encuentren en la misma carpeta que el *script* y que los nombres coincidan con los que aparece en las variables en las que se cargan los datos.

Por otro lado, para ejecutar las funciones *grafs_long_grieta()* y *grafs_vida_est()* solo hay que pasarle el identificador de un experimento. Este identificador tiene que ser exactamente igual al que se haya utilizado en la variable "exp_id" dentro de la función *principal()*.

Código B.4 Ejecución del *script graficas.py*.

```
if __name__ == "__main__":

    vida_exp_sin = {'6629_971_70':[165696, 316603], '5429_971_110'
        :[112165, 126496],
```

```
                ’5429_1257_110’:[113799, 120663], ’4217_1543_110’
                    :[88216, 89376],
                ’5429_1543_110’:[82559, 87481], ’3006_971_150’:[59234,
                     60040],
                ’4217_971_150’:[60288, 67776], ’5429_971_150’:[47737,
                    51574],
                ’3006_1543_150’:[19223, 39408], ’4217_1543_150’
                    :[50369, 39001],
                ’5429_1543_150’:[50268, 39202], ’3006_2113_150’
                    :[34904, 41002],
                ’4217_2113_150’:[34716, 40004], ’5429_2113_150’
                    :[32339, 36431],
                ’3006_971_175’:[26587, 31815], ’4217_971_175’:[27724,
                    32843],
                ’5429_971_175’:[29100, 35171], ’3006_1543_175’:[30154,
                     31224],
                ’4217_1543_175’:[34748, 34930], ’5429_1543_175’
                    :[28005, 33349],
                ’3006_2113_175’:[21207, 21669], ’4217_2113_175’
                    :[26989, 28595],
                ’5429_2113_175’:[28112, 28178]}

vida_exp_SP = {’6629_971_70’:[5000000, 5000000], ’5429_971_110’
    :[980678, 1811104],
                ’5429_1257_110’:[1649736, 1941545], ’4217_1543_110’
                    :[1110174, 1117513],
                ’5429_1543_110’:[810402, 1008310], ’3006_971_150’
                    :[231459, 665167],
                ’4217_971_150’:[634259, 678676], ’5429_971_150’
                    :[631491, 707514],
                ’3006_1543_150’:[275417, 198884], ’4217_1543_150’
                    :[234651, 497260],
                ’5429_1543_150’:[290460, 482862], ’3006_2113_150’
                    :[140189, 267873],
                ’4217_2113_150’:[166088, 201105], ’5429_2113_150’
                    :[66564, 190763],
                ’3006_971_175’:[364153, 366164], ’4217_971_175’
                    :[308455, 235467],
                ’5429_971_175’:[338774, 413654], ’3006_1543_175’
                    :[163474, 164835],
                ’4217_1543_175’:[164384, 169382], ’5429_1543_175’
                    :[228379, 231132],
                ’3006_2113_175’:[68801, 83544], ’4217_2113_175’
                    :[121642, 127770],
                ’5429_2113_175’:[94741, 146553]}

vida_exp_elec = {’6629_971_70’:[148020, 170073], ’3006_2113_150’
    :[14239, 24875],
                 ’5429_2113_150’:[16783, 19005], ’3006_971_175’
                     :[19267, 23803],
```

```
                    '5429_971_175':[23591, 25318],'3006_2113_175':[16743,
                        19173],
                    '5429_2113_175':[16543, 16100]}

    crit = 'FS'

    grafs_globales(vida_exp_sin, crit)
    error(crit, vida_exp_sin)
    error(crit, vida_exp_sin, vida_exp_SP, vida_exp_elec)

    for exp in vida_exp_sin:
        grafs_long_grieta(exp, crit)
        grafs_vida_est(exp, crit)
```

Bibliografía

[1] Python software foundation. *https://www.python.org/psf/*. 21/07/2019.

[2] Python. *https://www.python.org*. 21/07/2019.

[3] Anaconda. *https://www.anaconda.com/*. 03/08/2019.

[4] C. Navarro Pintado. *Iniciación y Crecimiento de Grietas en Fatiga por Fretting*. PhD thesis, Universidad de Sevilla, 2005.

[5] A. Fatemi; D. F. Socie. A critical plane approach to multiaxial fatigue damage including out-of-phase loading. *Fatigue and Fracture of Engineering Materials and Structures*, 11:149–165, 1988.

[6] K. N. Smith; P. Watson; T. H. Topper. A stress-strain function for the fatigue of metals. *Journal of Materials, JMLSA*, 5:767–778, 1970.

[7] H. J. Bueckner. Weight functions and stress-intensity factors. En G.C Sih, Methods of analysis and solutions of crack problems, páginas 306-307, Leyden: Noordhoff International Publishing. 1973.

[8] G. R. Irwin. Crack-extension force for a part-through crack in a plate. *Journal of Applied Mechanics*, 29:651–654, 1962.

[9] V. Martín Rodríguez. *Caracterización del tratamiento superficial de shot peening en fatiga por fretting*. PhD thesis, Universidad de Sevilla, 2019.

[10] C. Navarro; S. Muñoz; J. Domínguez. On the use of multiaxial fatigue criteria for fretting fatigue life assessment. *International Journal of Fatigue*, 30:32–44, 2008.