UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA MECÁNICA

PROYECTO DE FIN DE CARRERA

INGENIERÍA TÉCNICA INDUSTRIAL MECÁNICA

Desarrollo de un macro para el cálculo, diseño y modelado de resortes de torsión en Solid Edge.

Autor:

Antonio Álvarez Punzano

Tutor:

Jesús Meneses Alonso

Índice de contenido

Índice de figuras.

1. Introducción.

1.1 Introducción.

El uso de software para el diseño asistido por computador, comúnmente conocido como CAD, supone una gran avance en el mundo de la ingeniería, permitiendo diseñar y simular elementos mecánicos cuyo desarrollo de otra manera resultarían mucho más costosos tanto términos económicos como temporales.

El dominio de las aplicaciones CAD se ha convertido en un requisito prácticamente indispensable dentro del campo de la ingeniería, permitiendo aumentar la productividad y realizar tareas en un tiempo que hace no muchos años resultaría impensable. El avance de las aplicaciones ha ido de la mano del avance de las prestaciones ofrecidas por los equipos informáticos lo cual ha permitido dotar al ingeniero de herramientas muy potentes sin las cuales muchos de los proyectos actuales no serían posibles.

Del mismo modo que la informática y la electrónica cobra cada día más peso en la vida cotidiana de las personas, se ha hecho necesario incorporar estas nuevas tecnologías al mundo de la industria a fin de mejorar la competitividad y las herramientas disponibles para desarrollar y evolucionar los productos creados.

Dentro de la funcionalidades que poseen muchas aplicaciones informáticas destinadas al entorno comercial estará incluida la posibilidad del uso de macros. Esta característica permitirá complementar la funcionalidad de los programas de tal forma que será posible satisfacer muchas necesidades que en un primer momento el programa en cuestión no sea capaz de satisfacer.

Otra posibilidad que ofrece el uso de macros es automatizar las tareas repetitivas que se producen de manera cotidiana y que pueden repercutir en la productividad, resultar tediosas o susceptibles de ser causa de errores.

Simplificar o automatizar por completo ciertos procesos supondrá en la

mayoría de los caso una gran ventaja por parte del usuario a la hora de trabajar con un programa determinado, también reducirá la necesidad de tener ciertos conocimientos técnicos lo cual hará más accesible el uso de determinadas herramientas cuya compresión pudiera resultar no demasiado sencilla.

En este proyecto se ha creado una herramienta informática que sirviéndose del entorno de Solid Edge es capaz de calcular, diseñar y modelar resortes helicoidales que satisfacen los requisitos impuestos sin necesidad de que el usuario final conozca las ecuaciones que rigen el comportamiento de los resortes helicoidales de torsión, o las técnicas necesarias para modelar una pieza de estas características en Solid Edge. A través de una pantalla con una información mostrada de manera asequible el usuario será capaz de calcular y obtener el diseño de un resorte que cumpla con sus necesidades.

La simplificación del proceso de trabajo por parte del usuario de una aplicación habitualmente irá asociada a la necesidad de unos conocimientos y experiencia más avanzados por parte del creador de la aplicación, dado que por lo general la creación de macros implicará un profundo conocimiento de la aplicación en la que se desee hacer funcionar el macro así como de las herramientas necesarias para desarrollar dicho macro.

Por tanto siempre será necesario sopesar en qué medida la inclusión del macro mejora la funcionalidad de un programa, dado que en la mayoría de los casos desarrollar un macro supondrá una inversión de tiempo considerable y también una inversión económica dado que frecuentemente será necesario el uso de software de terceros para generar dicho macro.

La posibilidad de cubrir funcionalidades no incluidas por defecto en una aplicación será el mayor incentivo a la hora de desarrollar macros, permitiendo aprovechar al programador las herramientas y utilidades de un programa ya existente para crear nuevas herramientas sin necesidad de crear una nueva aplicación, hecho que permitirá abaratar costes evitando tener que cambiar una aplicación que por todo lo demás satisface las necesidades del usuario.

Este es el caso para este proyecto, en el que se ha añadido una nueva

funcionalidad al programa de Solid Edge en su versión ST7, automatizando una secuencia de operaciones e implementando la funcionalidad del programa añadiéndole la capacidad de realizar los cálculos necesarios sin necesidad de aplicaciones externas.

1.2 Objetivos.

El objetivo principal de este proyecto es el desarrollo de una herramienta informática capaz de calcular y dimensionar resortes de torsión helicoidales que satisfagan las necesidades del usuario y que sirviéndose del entorno de trabajo de un aplicación informática CAD, sea capaz de trasladar los resultados numérico a un modelo 3D de manera automatizada.

Para el desarrollo de la aplicación ha sido necesario el uso de herramientas informáticas, un entorno de programación y un software de diseño asistido por computador. Por tanto parte del objetivo del proyecto es familiarizarse y dominar el funcionamiento de este tipo de herramientas.

Si bien el estudio de diseño asistido por computador forma parte del plan de estudios de la titulación Ingeniería Técnica Industrial Mecánica, en la asignatura homónima Diseño asistido por computador en segundo curso, el aprendizaje de la programación de tareas repetitivas no forma parte del contenido de la asignatura y para el correcto desarrollo del proyecto ha sido necesario el estudio y compresión de las técnicas necesarias para tal disciplina.

Del mismo modo parte del objetivo del proyecto ha sido el aprendizaje de las nociones necesarias de programación imprescindibles para el desarrollo de la aplicación que controlará de manera automáticas herramientas de Solid Edge. Para ello ha sido necesario adquirir unos conocimientos básicos en el lenguaje de programación elegido, dentro de los lenguajes compatibles con la programación de tareas en Solid Edge.

Aparte de los elementos implicados en el desarrollo de la aplicación que automatiza el proceso de modelado de resortes helicoidales, obviamente también ha sido necesario el estudio y compresión de las características y comportamiento mecánico de los elementos mecánicos cuyo diseño se quiere automatizar, los resortes de torsión helicoidales.

Por tanto, de manera esquemática los objetivos del proyecto quedan definidos como:

- Estudio y compresión de nociones básicas de en programación en Visual Basic.Net así como de su entorno de trabajo.

- Utilización del software CAD Solid Edge orientado al modelado de elementos mecánicos, en este caso de resortes helicoidales de torsión.

- Aprendizaje de la técnicas necesarias para la programación de tareas repetitivas orientadas al diseño asistido por computador para Solid Edge.

- Estudio de las propiedades mecánicas de los resortes helicoidales de torsión.

- Estudio del fenómeno de fatiga aplicado al cálculo de resortes de torsión.

- Desarrollo de una interfaz que permita al usuario introducir los requisitos a satisfacer por el resorte y obtener los resultados de los parámetros calculados. Esta interfaz también ejecutará la orden que desencadenará el modelado automático del resorte en Solid Edge.

1.3 Fases de desarrollo.

Gran parte del trabajo del proyecto se ha invertido en el aprendizaje de los conocimientos necesarios para desarrollar un macro que funcione en el entorno CAD elegido, Solid Edge.

La primera fase del proyecto ha sido el aprendizaje de las nociones básicas en el lenguaje de programación elegido, Visual Basic.Net y de su entorno de trabajo. Si bien no es necesario un gran dominio del lenguaje de programación, su conocimiento facilitará la aproximación a la disciplina de la programación de tareas repetitivas, comúnmente denominadas “macro”.

El siguiente paso es familiarizarse con la programación de macros para Solid Edge. Para ello Solid Edge viene acompañado de una guía de ayuda donde se incluye la información relativa a todas las operaciones de las herramientas incluidas.

Tras conseguir hacer funcionar los primeros macros para Solid Edge, el siguiente paso fue el estudio de las propiedades de los resortes sometidos a torsión, así como la comprensión y utilización de las fórmulas matemáticas que describen su comportamiento mecánico y su resistencia frente fenómenos de fatiga.

A partir de comprender el funcionamiento de los resortes y poseer la nociones necesarias en programación de tareas repetitivas en Solid Edge ha sido posible comenzar el desarrollo del macro para Solid Edge.

La primera fase del desarrollo de la herramienta fue la creación de un macro capaz de diseñar de manera automática una extrusión helicoidal a la que se añadirían dos extrusiones rectas, una en cada extremo de la hélice. Dicho macro permitía modificar las dimensiones de cada una de las partes de la hélice con extremos rectos a través de la consola del sistema.

La segunda fase del desarrollo del macro ha sido la creación de una interfaz gráfica con la que el usuario podrá interactuar, de tal forma que se ofrezca la

información necesaria de la manera más amigable posible.

La tercera fase ha sido el desarrollo del código del programa, asociado a los botones presentes en la pantalla del programa, que permite realizar los cálculos relativos a las propiedades del resorte, así como desencadenar la secuencia de operaciones en Solid Edge que darán como resultado un diseño en tres dimensiones cuyas cotas se ajustarán a los valores numéricos obtenidos de los cálculos realizados por la aplicación.

La última parte del desarrollo del proyecto ha sido la comprobación y depuración de los diversos errores del programa y más en concreto de los errores asociados al diseño automático de la pieza en Solid Edge, dado que esta ha sido la parte en la que más complicaciones has surgido durante el desarrollo del macro.

1.4 Medios empleados.

Para alcanzar los objetivos del proyecto, además de un ordenador, son necesarios un software de diseño asistido por computador, capaz de soportar el uso de macros y un entorno de programación que utilice un lenguaje compatible con el programa de diseño elegido.

El software de diseño asistido por computador elegido ha sido Solid Edge ST7 bajo licencia académica:

Figura 1: Inicio Solid Edge ST7

La versión académica ofrece plena funcionalidad con la única limitación de que los archivos generados no son compatibles con las versiones comerciales del producto y por tanto no podrán ser abiertos en equipos con licencias distintas a la académica.

Solid Edge permite la inclusión de macros en su entorno de trabajo, siendo posible integrarlos en su interfaz o bien ejecutarlos a través del correspondiente archivo .exe del macro utilizado.

Para generar el macro es necesario un entorno de programación donde crear el código y compilarlo para poder obtener el archivo ejecutable o macro. Solid Edge es compatible con múltiples lenguajes informáticos que soporte el uso de instrucciones COM, siendo los más conocidos C# y Visual Basic.Net.

El software elegido para la realización del macro ha sido Microsoft Visual Basic 2010 Express , que posee un entorno de programación compatible con múltiples lenguajes de programación, entre los que se incluyen C# y .NET, así como una completa colección de herramientas destinadas a facilitar la programación y el aprendizaje de la misma.

Otra herramienta imprescindible a la hora de realizar el macro de cálculo y diseño de resortes dentro de Solid Edge será la guía de ayuda a la programación de automatización incluida dentro de las opciones de ayuda de Solid Edge.

Si bien a la hora de crear un macro se podrá ser tan flexible como flexible permite ser la interfaz y las herramientas propias de Solid Edge (una misma pieza se puede hace de muy distintas maneras) y por tanto no habrá un código único capaz de generar un macro que realice una determinada operación, a la hora de programar una función u operación propia de Solid Edge será necesario cumplir escrupulosamente las instrucciones del manual relativas a dicha operación o función.

Existen disponibles guías de introducción a la programación en Solid Edge en las que se resumen y se exponen algunos ejemplos y pasos imprescindibles para iniciarse en la programación de tareas repetitivas en Solid Edge, pero dada la inmensa cantidad de operaciones realizables y de los múltiples entornos de trabajo de Solid Edge éstas no recogen la información necesaria para todas y cada una de las posibilidades que ofrece la creación de macros en Solid Edge, quedando en su mayoría como ayuda a la iniciación a la programación de macros.

Por tanto, y muy en especial para el programador novel, el uso de la guía de ayuda para la programación de Solid Edge, Solid Edge SDK ST7 (o su versión contemporánea), será el documento de referencia imprescindible para todo aquel que quiera exprimir al máximo las posibilidades que ofrece la creación de macros para Solid Edge.

Esta guía, accesible a través del menú de ayuda de Solid Edge, en su versión ST7, sólo está disponible a través de internet, por tanto será necesario una conexión a internet para poder consultarla.

Por último, si bien no es una herramienta en si mismo, queda mencionar la elección del lenguaje de programación elegido de entre las opciones disponibles, elección que cobra especial relevancia en el caso de carecer de conocimientos de programación. Las opciones más recomendable, salvo que se tengan conocimientos de programación en otro lenguaje compatible, son C# y Visual Basic .NET debido a la cantidad de documentación disponible y la ayuda facilitada por Solid Edge para aprender a programar macros.

En este caso el lenguaje seleccionado para la realización del Macro ha sido Visual Basic .NET.

1.5 Estructura de la memoria.

La redacción de este documento se ha estructurado en varia secciones diferenciadas acorde a la temática del contenido tratado en cada una de ellas:

- En el capítulo 2 se expone la teoría de resistencia mecánica de materiales orientada en la descripción el comportamiento de elementos mecánicos sometidos a esfuerzos tanto estáticos como dinámicos. Este capítulo sirve como base para introducir los conocimientos necesarios para el siguiente apartado.

- El capítulo 3 se centra en la descripción del comportamiento de los resortes sometidos a torsión. En él se ilustran las leyes que permiten diseñar y calcular dichos elementos mecánicos así como la determinación de las condiciones necesarias para garantizar la vida infinita de los resortes.

- El capítulo 4 esta centrado en Visual Basic. En este apartado se describen los conocimientos fundamentales necesarios para el correcto desarrollo de la herramienta objetivo del proyecto.

- El capítulo 5 describe el uso de macros dentro de Solid Edge. En él se ilustra la manera de proceder a la hora de desarrollar macros que permitan automatizar tareas dentro del entorno de Solid Edge.

- El capítulo 6 muestra el proceso de desarrollo del Macro para el diseño de resortes de torsión helicoidales en Solid Edge. En este capítulo se describe el proceso de desarrollo, el código del macro desarrollado y su funcionamiento dentro de Solid Edge. Abarca también la creación de la interfaz de la herramienta.

- El capítulo 7 contiene las conclusiones extraídas del proyecto.

- En el capítulo 8 se enumeran y describen posibles trabajos futuros basados en la herramienta desarrollada.

- El capítulo 9 contiene la bibliografía consultada para la elaboración del proyecto.

- El capítulo 10, anexo del proyecto, contiene el código completo de la herramienta desarrollado en Visual Basic.Net.

2 Resistencia de materiales.

2.1 Introducción.

Uno de los condicionantes más importantes a la hora de diseñar o desarrollar un elemento mecánico es la resistencia de dicho objeto o del material o materiales que lo conforman.

La resistencia del objeto dependerá tanto de los materiales de las piezas que lo conforman como de su forma, así como del los distintos procesos de fabricación que se hayan llevado a cabo a la hora de su fabricación, tanto para la obtención de los materiales como para dar su forma final al elemento mecánico. La resistencia de los elemento mecánicos también se verá afectada por factores ambientales, tales como las temperaturas de trabajo o la corrosión.

Dada la importancia de optimizar costes tanto económicos como materiales y evitar las posibles consecuencias materiales y personales es de suma importancia ser capaz de conocer con la máxima fiabilidad las capacidades mecánicas de los distintos elementos que componen la maquinaria y utensilios utilizados.

Las cargas o esfuerzos aplicados a un elemento mecánico pueden ser estáticas o dinámicas, precisando cada una de ellas de un tratamiento y estudio diferenciado para garantizar su integridad.

Las cargas estáticas (fuerza o momentos) permanecen invariantes en el tiempo, tanto en módulo como la dirección y punto de aplicación. El elemento mecánico fallará estructuralmente si se le somete a un esfuerzo mayor del que sus materiales y geometría pueden soportar.

Las cargas dinámicas por el contrario son aquellas en las que la fuerza y/o el momento varían en el tiempo, pudiendo cambiar tanto su módulo y/o dirección como la zona en la que el esfuerzo es aplicado. Esta variaciones pueden dar lugar a la aparición

de un fenómeno, denominado fatiga, que puede dar como resultado que el elemento mecánico falle ante una carga muy inferior al que podría soportar de manera estática.

Dependiendo de si un elemento mecánico está sometido a cargas estáticas o dinámicas sera preciso realizar diferentes tipos de estudio, dado que cada fenómeno se caracteriza de manera distinta y requiere tratamientos diferenciados a fin de determinar y garantizar la integridad estructural de los elementos mecánicos diseñados.

Si bien es posible calcular la resistencia de un elemento de manera práctica, es decir, a partir de un modelo fabricado sometido a las cargas y situaciones requeridas, los costes derivados esta práctica de manera generalizada hacen muy valiosos el cálculo teórico sus propiedades mecánicas, bien manualmente (lo cual puede llegar a ser muy laborioso e incluso imposible según el caso) o bien mediante diseño asistido por computador y la aplicación de métodos numéricos.

2.2 Resistencia de materiales sometidos a cargas estáticas.

En el caso de un elemento sometido a una carga estática será necesario comprobar que las tensiones sufridas debidas a la carga no sobrepasen la tensión máxima permitida por el material, que dependiendo del caso este valor vendrá acotado o bien por el límite elástico del material, o bien por su resistencia última a rotura.

A la hora de caracterizar la resistencia de un elemento sometido a cargas estáticas, será necesario diferenciar entre los distintos tipos de materiales en función del tipo de fractura que suelen presentar:

- Materiales dúctiles: los materiales dúctiles son aquellos que presentan una deformación plástica, es decir, permanente, antes de presentarse la rotura del mismo. Los materiales dúctiles son capaces de soportar cierta deformación bajo la cual son capaces de recuperar su forma si la carga desaparece, manteniendo una relación lineal entre la carga aplicada y la deformación del material. Una vez superado cierto grado de deformación comenzará la deformación permanente. Para la mayoría de aplicaciones en ingeniería las deformaciones permanentes no son aceptables y por tanto la tensión tomada como referencia será aquella que el material sea capaz de tolerar sin sufrir deformaciones permanentes.

- Materiales frágiles: los materiales frágiles por el contrario, prácticamente no sufren deformación plástica antes de llegar a romperse. Sufrirán una deformación lineal con respecto a la tensión hasta prácticamente alcanzar el valor de tensión máximo soportado, su resistencia última o de rotura S_u, tras lo cual el material se fracturará. Para los materiales frágiles el valor tomado como referencia a la hora de caracterizarlo será su resistencia última S_u.

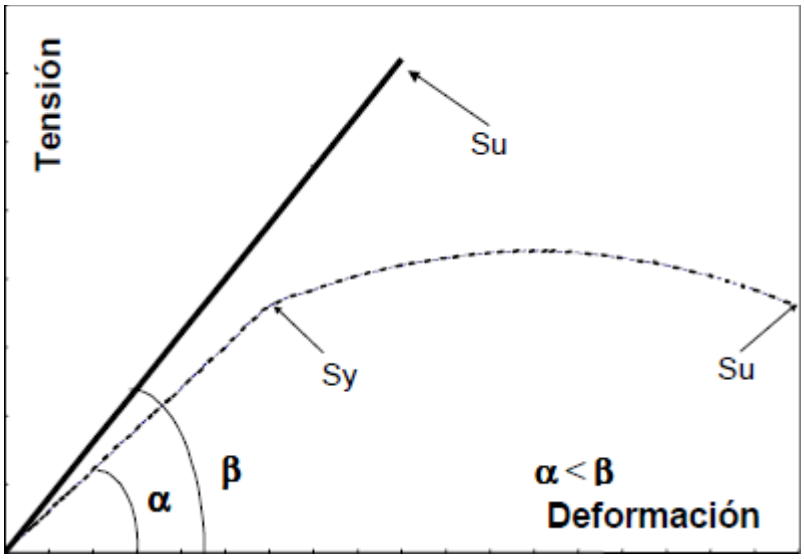

Figura 2: Curva Tensión-Deformación[1].

En la figura 2 puede apreciarse la diferencia entre un material dúctil y un material frágil. La curva que describe la relación entre la deformación y la tensión aplicada para una material dúctil presenta dos zonas claramente diferencias. Una primera zona en la que la deformación y la tensión son directamente proporcionales hasta alcanzar un determinado valor, el límite a fluencia del material S_y, y una segunda zona donde dicha linealidad desaparece hasta alcanzar el límite a rotura S_u.

En la curva que describe la relación entre la deformación y la tensión aplicada prácticamente solo existe la zona en la que la relación entre la deformación y la tensión son directamente proporcionales, tras lo cual el material alcanza su resistencia última y se fractura sin sufrir apenas deformaciones permanentes.

El gráfico no se especifica si el material esta siendo sometido a un esfuerzo de compresión o de tracción. Es necesario mencionar que mientras que para los materiales dúctiles este hecho no reviste de gran importancia dado que sus propiedades serán la mismas, para un material frágil será imprescindible conocer el sentido de las tensiones aplicadas, pues su límite a rotura será sensiblemente inferior en caso de estar sometido a esfuerzos de tracción en lugar de esfuerzos de compresión.

2.3 Cálculo de resistencia para materiales dúctiles.

2.3.1 Criterio del esfuerzo normal máximo.

Esta teoría establece que la rotura del material se produce si alguna de las tensiones principales es mayor que el límite a rotura del material. Esta teoría no describe correctamente el comportamiento de la resistencia a rotura de los materiales, llegando a establecer en ocasiones resultados que quedan fuera de los límites reales soportados[1].

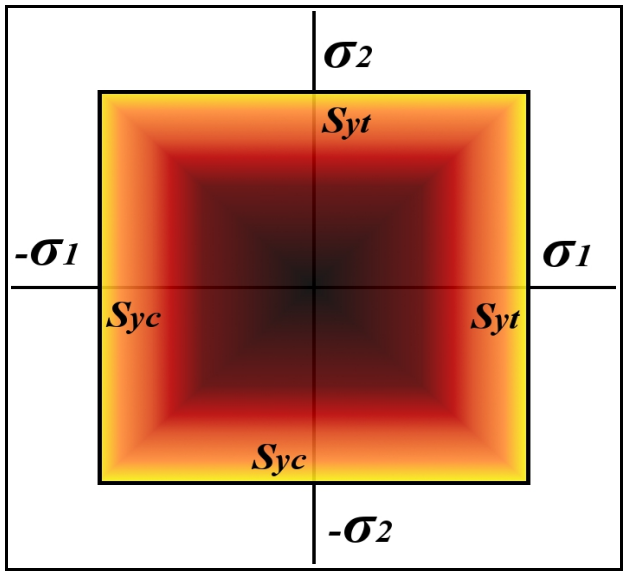

Figura 3: Esfuerzo Normal Máximo.

Todo esfuerzo que quede excluido del área de la figura producirá el fallo del material.

Aplicando este criterio para el caso de un esfuerzo a torsión pura se obtiene que la tensión a cortadura máxima soportada es:

$$\tau_{max} = S_y$$

Cuando experimentalmente se ha comprobado que este valor es aproximadamente de 0,6 S_y [4].

Por tanto, dado que no se garantiza que los resultados ofrecidos se correspondan con el comportamiento real del material, esta teoría no debe ser utilizada

para garantizar la integridad estructural de un elemento mecánico[1].

2.3.2 Criterio de plastificación de Tresca.

Esta teoría se basa en establecer que la fluencia del material es el resultado del esfuerzo cortante.

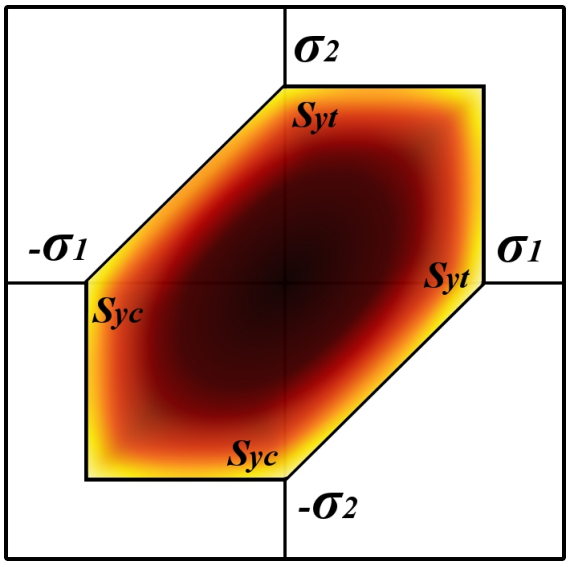

Figura 4: Criterio de Tresca.

En la figura 4 se puede observar gráficamente la corrección aplicada con respecto la teoría del esfuerzo normal máximo, el área de los cuadrantes 2 y 4 se ha modificado de tal manera que los resultados arrojados por el criterio de plastificación de Tresca siempre quedan dentro del lado de la seguridad y por tanto todos los resultados dentro del área de la figura garantizarán la integridad del elemento mecánico bajo esfuerzos estáticos.

La fluencia del material ocurre cuando el esfuerzo cortante máximo absoluto es igual al esfuerzo cortante máximo alcanzado durante un ensayo a tracción cuando en la probeta alcanza la fluencia.

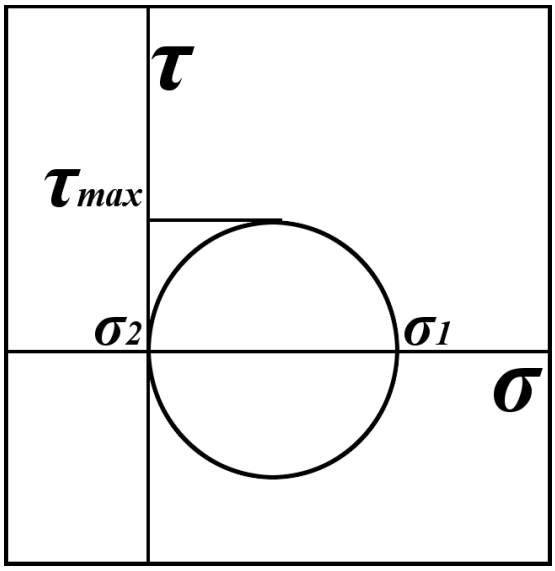

Figura 5: Tresca circulo de Morh.

En esta figura se muestran los resultados de un ensayo a tracción mediante el uso de los círculos de Morh, donde τ representa el esfuerzo de cortadura y σ las tensiones principales.

Observado el gráfico se puede concluir que la tensión a cortadura máxima soportada viene dada por:

$$\tau_{max} = \frac{(\sigma_1 - \sigma_2)}{2}$$

Por tanto, de acuerdo con la afirmación inicial se concluye que el esfuerzo cortante máximo viene dado por:

$\tau_{max} = S_{SY} = S_Y/2 = (\sigma_1 - \sigma_2)/2$ por tanto $S_{SY} = S_Y/2$

2.3.3 Criterio de plastificación de Von Mises-Henky.

Esta teoría surge como resultado de la observación de que los materiales sometidos a esfuerzos hidrostáticos (es decir, esfuerzos de tracción o compresión iguales en todas las direcciones) soportan cargas mucho mayores que sus esfuerzos de

fluencia en otros tipos de carga[1].

La teoría se basa en que es posible descomponer el estado de tensiones de un elemento como la superposición de dos estados, un estado en el que todas las tensiones aplicadas en las direcciones son iguales (tensiones hidrostáticas) más otro estado con las tensiones aplicadas en el estado inicial restando las las tensiones de estado ficticio con las tensiones hidrostáticas.

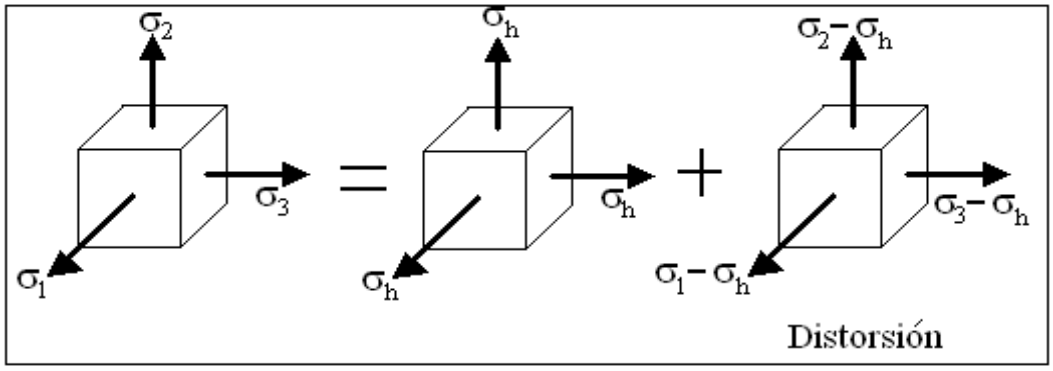

Figura 6: Tensiones hidrostáticas.

Donde :

$$\sigma_h^2 = \frac{(\sigma_1 + \sigma_2 \sigma_3)}{3}$$

En el estado de esfuerzos hidrostáticos sólo se produce cambio de volumen y en el otro estado auxiliar sólo se producen deformaciones angulares, sin cambio de volumen.

La energía asociada a cada una de las direcciones principales vendrá dad por:

$U_n = (\sigma_n - \varepsilon_n)/2$

Luego la energía total será la suma de la energía asociada a cada una de las direcciones principales:

$$U = U_1 + U_2 + U_3 = (\sigma_n - \varepsilon_n)/2$$

$$U = U_1 + U_2 + U_3 = \frac{(\sigma_1 \varepsilon_1)}{2} + \frac{(\sigma_1 2\varepsilon_2)}{2} + \frac{(\sigma_3 \varepsilon_3)}{2}$$

Dado que las deformaciones son:

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -v & -v \\ -v & 1 & -v \\ -v & -v & 1 \end{bmatrix}$$

Donde E es el módulo de Young y ν el coeficiente de Poisson. Si se despeja esta ecuación se obtiene que la energía total de la deformación es:

$$U = \frac{1}{(2E)} . (\sigma_1^2 + \sigma_2^2 + \sigma_3^2 - 2\mathrm{v}(\sigma_1 . \sigma_2 + \sigma_2 . \sigma_3 . + \sigma_1 . \sigma_3))$$

Aplicando el mismo razonamiento para el estado de deformación debido a la presión hidrostática se obtiene que:

$$U_h = 3 \frac{(1-2\mathrm{v})}{(2E)} . \sigma_h^2 = 3 \frac{(1-2\mathrm{v})}{(2E)} . \left(\frac{(\sigma_1 + \sigma_2 \sigma_3)}{3} \right)^2$$

La energía de distorsión es por tanto:

$$U_d = U - U_h$$

$$U_d = \left(\frac{1}{(2E)} . (\sigma_1^2 + \sigma_2^2 + \sigma_3^2 - 2\mathrm{v}(\sigma_1 . \sigma_2 + \sigma_2 . \sigma_3 . + \sigma_1 . \sigma_3)) - \left(3 \frac{(1-2\mathrm{v})}{(2E)} \right) . \left(\frac{(\sigma_1 + \sigma_2 \sigma_3)}{3} \right)^2 \right)$$

$$U_d = \frac{(1+v)}{(3E)} . (\sigma_1^2 + \sigma_2^2 + \sigma_3^2 - \sigma_1 . \sigma_2 - \sigma_2 . \sigma_3 - \sigma_1 . \sigma_3)$$

En el caso de un ensayo a tracción simple donde $\sigma_1 = S_y$ y $\sigma_2 = \sigma_3 = 0$ se obtiene que:

$$U_d = \frac{(1+v)}{(3E)} . S_y^2$$

Igualando ambas ecuaciones de U_d se obtiene que:

$$S_y = \sqrt{\frac{((\sigma_1-\sigma_2)^2+(\sigma_2-\sigma_3)^2+(\sigma_1-\sigma_3)^2)}{2}}$$

Se define el esfuerzo de Von Mises como:

$$\sigma_{vm} = \sqrt{\frac{((\sigma_1-\sigma_2)^2+(\sigma_2-\sigma_3)^2+(\sigma_1-\sigma_3)^2)}{2}}$$

En un caso bidimensional:

$$S_y^2 = \sigma_1^2 - \sigma_1 . \sigma_2 + \sigma_2^2$$

Y en caso de torsión pura se cumplirá que:

$$S_{SY} = 0{,}577\, S_Y$$

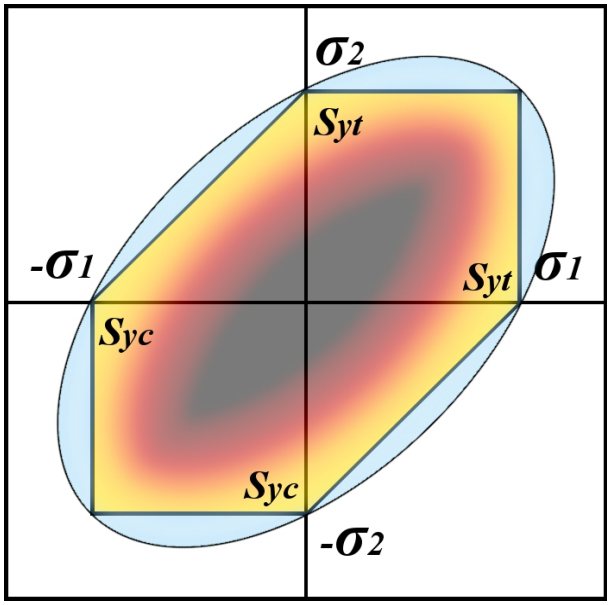

Figura 7: Criterio de Von Mises.

En la figura 7 se ve de manera gráfica la diferencia entre las tensiones admitidas para la teoría de Von Mises y la teoría de tensión de corte máximo en un

análisis bidimensional. El área de la elipse representa la teoría de Von Mises y las zonas coloreadas de azul indican las diferencias de tensiones entre la teoría de Von Mises y la teoría de tensión de corte máximo, representada por el hexaedro.

2.4 Resistencia mecánica en materiales frágiles.

Se considera que un material es catalogado como frágil cuando durante el ensayo de tensión éste presenta una deformación plástica de menos del 5 por ciento de su longitud original [4]. Dado que la deformación plástica es muy pequeña en los materiales frágiles, se considera su resistencia a rotura como valor de referencia a la hora de realizar los correspondientes cálculos de resistencia de un elemento fabricado con dicho tipo de materiales.

2.4.1 Teoría del esfuerzo normal máximo.

Esta teoría establece que la rotura se producirá cuando las tensiones aplicadas sean igual o mayores a la resistencia última del material registrada durante un ensayo de tracción o compresión[4].

En un caso bidimensional la teoría queda representada gráficamente como:

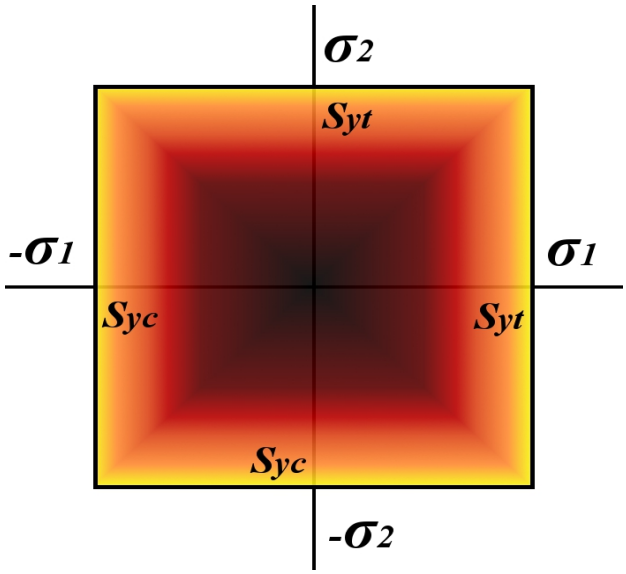

Figura 8: Esfuerzo Normal Máximo.

Y por tanto el material se fracturará cuando las tensiones aplicadas queden fuera del área encerrada por la figura 8.

Como se ha comentado en apartados anteriores esta teoría también es aplicable en el caso de materiales dúctiles, teniendo en cuenta que se debe tomar su límite a fluencia del material S_y como valor de referencia en lugar de su resistencia última S_t.

En el caso del estudio de un esfuerzo de cortadura pura, aplicando esta teoría se obtendrá que la tensión a cortadura máxima soportada viene dada por $\tau_{max}= S_u$ (S_y para el caso de materiales dúctiles).

En el aparto correspondiente al cálculo de resistencia para materiales dúctiles siguiendo el criterio de Von Misses se concluyo que esta aseveración no es cierta, para el caso de un esfuerzo de cortadura pura la tensión cortante máxima soportada es:

$$\tau_{max} = 0{,}577\, S_Y$$

Cifra que también se ha verificado experimentalmente por tanto la teoría del esfuerzo normal máximo puede arrojar resultados fuera del margen de seguridad y por consiguiente es recomendable no utilizarla[1].

2.4.2 Teoría de Coulomb-Morh.

Esta teoría deriva de la teoría de Coulomb-Morh aplicada a los materiales dúctiles, con la salvedad de que debido a las características de los materiales frágiles, que carecen de límite a fluencia, su cálculo se realiza teniendo en cuenta las tensiones a rotura de tracción y compresión.

$$\frac{\sigma_1}{S_{ut}} - \frac{\sigma_3}{S_{uc}} = 1$$

Donde $\sigma_1 \geq 0$, $\sigma_3 \leq 0$ y $S_{uc} \geq 0$.

En el resto de casos la rotura ocurrirá cuando:

$\sigma_1 = S_u$	si $\sigma_1 > \sigma_3 > 0$
$\sigma_3 = -S_{uc}$	$0 > \sigma_1 > \sigma_3$

En un análisis bidimensional del problema la teoría de Coulomb-Morh para materiales frágiles se representa como:

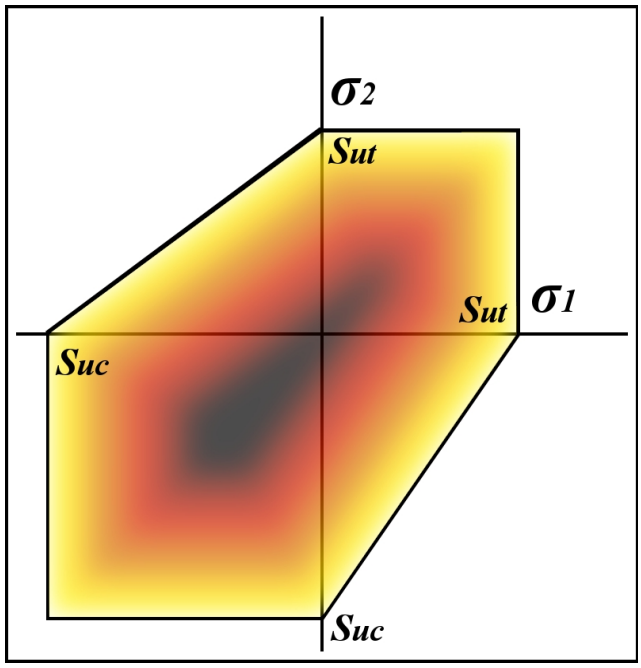

Figura 9: Teoría Coulomb-Morh.

Por tanto, el material se fracturará cuando las tensiones aplicadas queden fuera de área encerrada por la figura representada. En la figura 9, asociada al análisis bidimensional, puede apreciarse claramente cómo el área asociada a las tensiones de tracción es sensiblemente inferior a las tensiones de compresión.

2.5 Resistencia de materiales sometidos a cargas cíclicas.

2.5.1 Introducción.

En el caso de materiales o elementos mecánicos sometidos a cargas estáticas el criterio seguido a la hora de comprobar su punto de rotura o fallo se focaliza en garantizar que las tensiones sufridas durante la vida útil de elemento no sobrepasen las tensiones máximas permitidas por el material que lo compone.

Sin embargo en el caso de que la carga sea variable en el tiempo garantizar no sobrepasar las tensiones límites de cada situación no garantizan la integridad estructural del elemento objeto de estudió, pudiéndose dar el caso de que un elemento sometido a cargas variables falle bajo una carga muy inferior a la que en teoría podría soportar si estuviera sometido exclusivamente a esfuerzos estáticos.

Esto es debido a que la aplicación de una carga variable puede conllevar la propagación de pequeños defectos presentes en el material (internos o superficiales), dando como resultado la aparición y propagación de fisuras que a la larga generarán una disminución significativa del área efectiva sobre la que se distribuye la fuerza, es decir, aumento de la tensión así como la aparición de fenómenos de concentración de tensiones.

En la imagen siguiente se muestra como debido a los fenómenos de fatiga han aparecido grietas en el cuadro de una bicicleta de montaña, sin sufrir deformaciones permanentes:

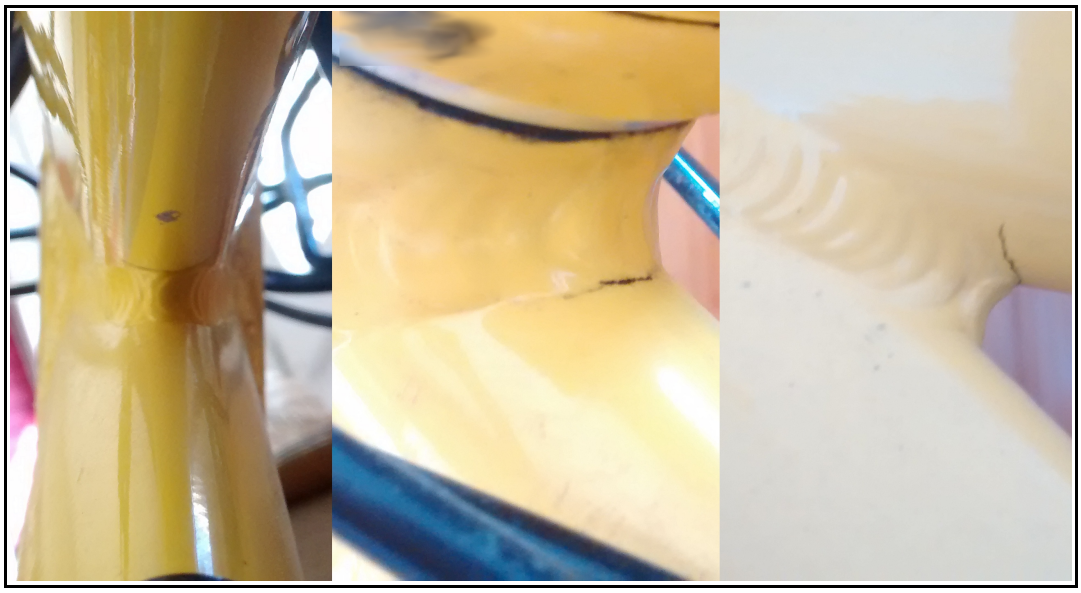

Figura 10: Fisura por fatiga.

En la figura 10 se aprecia la presencia de grietas en la parte superior e inferior del tubo de aluminio. Las grietas han aparecido tras años de uso y se han propagado hasta hacerse visibles sin necesidad de utilizar técnicas de detección de fisuras, en zonas susceptibles de sufrir concentraciones de tensiones debido al cambio de relieve en la superficie del cuadro de la bicicleta. Las soldaduras también son susceptibles de sufrir tensiones residuales, especialmente en este caso ya que se trata de un aluminio 7005 sin tratamiento térmico.

La circunstancia de que un material pueda fallar por debajo de su límite a rotura o fluencia hace obligatorio el estudio del comportamiento de los materiales sometidos a cargas fluctuantes para poder evitar la aparición de este tipo de roturas bajo cargas variables o bien garantizar la integridad de la estructura durante un tiempo razonable, dado que no siempre será posible garantizar la vida infinita de un elemento sometido a cargas dinámicas.

A este hecho se le conoce con el nombre de Fenómeno de Fatiga, y su estudio será imprescindible en todos los materiales y elementos mecánicos sometidos a cargas cíclicas dado que en la mayoría de los casos, si bien será necesario realizar el estudio de resistencia de materiales sometidos a cargas estáticas, las condiciones más restrictivas vendrán dadas por los resultados obtenidos del estudio de los fenómenos de fatiga en

elemento objeto de estudio.

2.5.2 Diagrama S-N. Resistencia a la fatiga y límite de fatiga.

Dado la inmensa cantidad de formas y elementos posibles susceptible de sufrir fenómenos de fatiga es preciso estandarizar un método mediante el cual poder estudiar dicho fenómeno a partir de una probeta normalizada, para poder aislar así los efectos debidos a la geometría propia del elemento y centrarse en las propiedades del material.

Para ello el método ideado es la utilización de una probeta cilíndrica con uno de los sus extremos fijado en un torno y a la que se le aplica una carga vertical (perpendicular a su eje de rotación) en el extremo libre mientras esta gira. De esta manera se consigue someter al la probeta a una carga fluctuante dado que al girar se irán alternando las tensiones de tracción y de compresión en el interior de la barra.

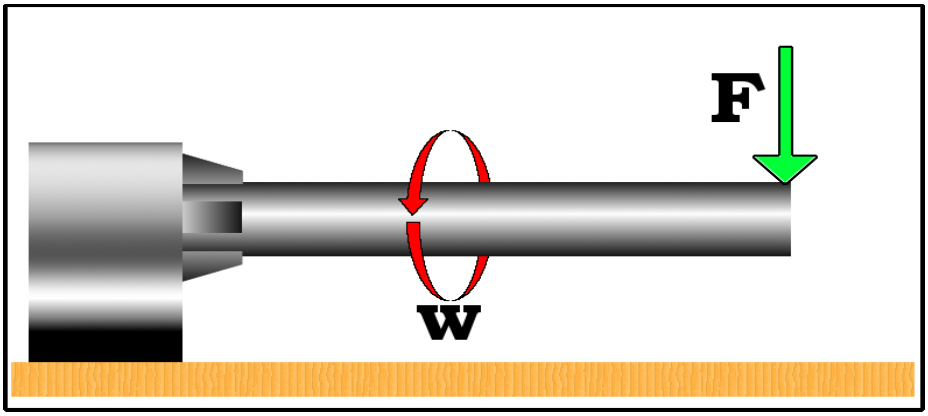

Figura 11: Ensayo de viga rotatoria.

Esto permite controlar el número de ciclos (las revoluciones) y las tensiones a la se somete la barra. Variando la carga será posible obtener el número de ciclos soportados en función de la tensión y por consiguiente establecer la relación entre la tensión y los ciclos soportados.

Linealizando la curva obtenida, mediante el uso de logaritmos, la gráfica obtenida será la siguiente:

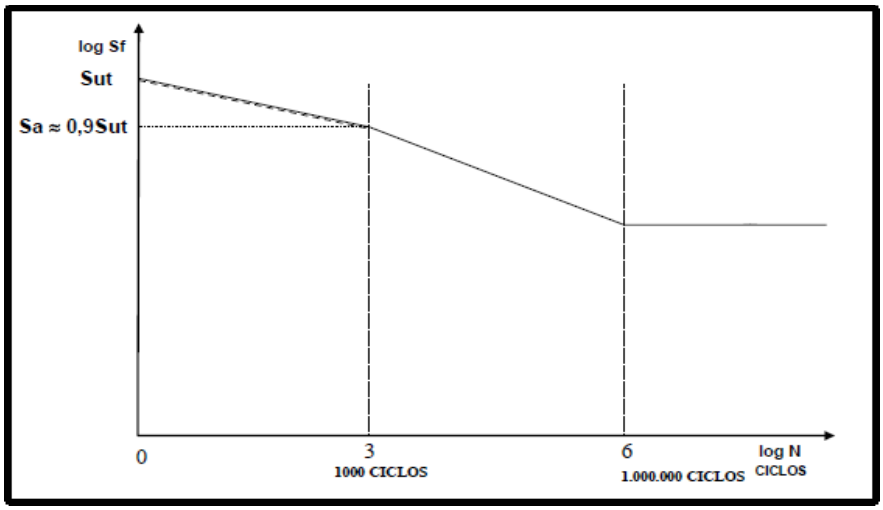

Figura 12: Curva S-N.

Así, a partir del estudio de la viga rotatoria, con un mínimo de 8 ensayos variando las cargas es posible obtener el denominado diagrama de fatiga o S-N[1].

En la gráfica se aprecian tres zonas claramente diferenciadas, donde la pendiente es significativamente distinta.

La primera zona, correspondiente al rango de tensiones S_u-0.9S_u, se aprecia como la pieza sometida a N ciclos falla bajo una carga próxima a su resistencia última, de lo cual se deduce que los defectos del material se propagan rápidamente cuando laa cargas cíclicas generan tensiones cercanas a S_u fallando aproximadamente antes de los mil ciclos. Dicha zona recibe el nombre de Ciclo bajo.

La segunda zona, para tensiones igual o inferiores a 0.9S_u hasta tensiones iguales a S_e abarca desde los mil ciclos hasta el millón de ciclos y es conocida como zona de fatiga Ciclo alto. En ella se puede apreciar cómo el material, a medida que aumentan los ciclos, falla bajo valores de tensiones sensiblemente menores a la resistencia última del material.

La última zona representa la zona de Vida infinita del material, que se alcanza para un número de ciclos igual o mayor a un millón. La tensión por debajo de la cual la probeta sometida al ensayo rotatorio nunca llega a romperse se conoce como límite de fatiga sin corregir S'_e.

Muchos materiales presentarán una ligera pendiente en la zona representada para el millón de ciclos en adelante, por tanto los materiales no siempre dispondrán de una carga por debajo de cual se garantice su vida infinita estando sometidos a cargas cíclicas, materiales como el aluminio y sus aleaciones carecen de límite fatiga [1].

En el caso de aceros y hierro y de no disponer de sus características sobre su límite de fatiga es posible usar el siguiente criterio [1]:

- Hierro y aceros forjados:

$S_e'=0.5S_u$	$S_{ut} \leq 1400MPa$
$S_e'=700Mpa$	$S_{ut} \geq 1400MPa$

- Aceros colados:

$S_e'=0.45S_u$	si $S_{ut} \leq 600MPa$
$S_e'=275$ MPa	si $S_{ut} \geq 600Mpa$

2.5.3 Coeficientes de corrección del límite de fatiga.

El límite de fatiga S'_e mostrado en el apartado anterior se corresponde con el ensayo normalizado a fatiga para una probeta sometido a un ensayo de viga rotatoria. Debido a los múltiples factores que intervienen en los fenómenos de fatiga en muchas ocasiones no será posible disponer de la curva S-N obtenida mediante un ensayo normalizado para el elemento objeto de estudio. Este obstáculo es salvable gracias a la aplicación de diversos coeficientes que corrigen el límite de fatiga obtenido en el ensayo de viga rotatoria y permiten extrapolar los resultados de dicho ensayo al cálculo de fatiga en distintos elementos.

El resultado de aplicar dichos coeficientes a S'_e permitirán calcular el valor teórico del límite de fatiga del elemento estudiado. A dicho valor se le conoce como el límite de fatiga corregido S_e. Este valor servirá para calcular la vida infinita del

elemento sometido a cargas cíclicas.

Los factores que condicionan el límite de fatiga son:

- Factor de acabado superficial, K_a
- Factor de tamaño, K_b.
- Factor de confiabilidad, K_c.
- Factor de temperatura, K_d.
- Factor de concentración de tensiones, K_e.
- Factor de efecto diversos, K_f.

La ecuación que relaciona el límite de fatiga del ensayo de la viga rotatoria con el límite de fatiga, denominada ecuación de Marin, es:

$$S_e = K_a . K_b . K_b . K_c . K_d . K_e . K_f . S_e^{'}$$

2.5.3.1 Factor de acabado superficial K_a.

En el ensayo normalizado de la viga rotatoria, ésta debe ser previamente pulida y además recibe un pulido final en dirección axial [1]. En el caso de no ser pulida la pieza se observa como el límite de fatiga se ve modificado. Esto es debido a la concentración de tensiones que puede surgir debido a las rugosidades del acabado de la pieza.

La forma de obtener K_a es a partir de los dato recolectados tras realizar múltiples ensayos mediante la siguiente correlación estadística[1]:

Acabado superficial	*Coeficiente a (MPa)*	*Exponente b*
Pulido	1	0
Acabado fino (esmerilado, rectificado,)	1.58	-0.085
Mecanizado sin acabar/estirado en frío	4.51	-0.265
Laminado en caliente	57.7	-0.718
Forjado	272	-0.995

$$K_a = a . S_{ut}^{b}$$

2.5.3.2 Factor de tamaño K_b.

Este factor no influye en las probetas sometidas a tracción pura, en cuyo caso su valor será K_b=1. En el caso de que los esfuerzos a los que se someta la pieza sean de torsión y flexión el factor vendrá dado por las siguientes ecuaciones:

- Para diámetros comprendidos entre 2,79mm y 51mm.

$$K_b = \left(\frac{d}{7.62} \right)^{-0,1133}$$

- Para diámetros comprendidos entre 52mm y 250mm:

$$K_b = 1,189 \, . \, d^{-0,097}$$

En el caso de que la sección del elemento no sea circular será preciso determinar un diámetro efectivo que permita calcular el K_b, de tal manera que el área equivalente obtenida permita el cálculo de la resistencia a fatiga de una probeta de sección no circular.

El diámetro efectivo se calcula a partir del área del 95% del esfuerzo. Esta área esta definida por las tensiones que superan el valor del 95% de la tensión máxima que sufre una probeta en cada punto de la sección analizada a medida que el esfuerzo va fluctuando.

$$A_{0,95} = \frac{\pi}{4} [d^2 - (0,95 \, . \, d)^2]$$

Diámetro efectivo para un área rectangular:

$$d_e = 0,808 \, . \, (h \, . \, b)^{\frac{1}{2}}$$

2.5.3.3 Factor de confiabilidad K_c.

Debido a la dispersión asociada a los fenómenos de fatiga es necesario aplicar

un coeficiente de seguridad que permita predecir las posibilidades de que la pieza estudiada sea capaz de soportar al menos los ciclos deseados. Dicha dispersión se adapta la curva de una distribución normal. La siguiente tabla muestra los valores de K_c en función de la confiabilidad requerida:

Confiabilidad	Factor de confiabilidad K_c
0,5	1
0,9	0,897
0,95	0,868
0,99	0,814
0,999	0,753
0,9999	0,702
0,99999	0,659
0,999999	0,620
0,9999999	0,584
0,99999999	0,551
0,999999999	0,520

2.5.3.4 Factor de temperatura K_d.

La temperatura altera la propiedades mecánicas de un material, por tanto su límite a rotura y a fluencia será diferente en función de la temperatura así como su respuesta al fenómeno de fatiga. Tanto es así, que los aceros que trabajan a altas temperaturas no disponen de límite de fatiga.

Calcular K_d puede resulta complejo, siendo recomendable siempre que sea posible calcularla de manera experimental mediante pruebas de laboratorio. Para el caso de resolución de problemas estos cálculos son válidos [1]:

$K_d = 1$	$T \leq 450ºC$
$K_d = 1- 5.8 \cdot 10^{-3} \cdot (T – 450)$	$450ºC \leq T \leq 550ºC$

2.5.3.5 Factor de concentración de tensiones K_e.

Este factor aparece como resultado de las distintas discontinuidades que pueden presentar loes elementos en su cuerpo, como puede ser el caso de roscas, taladros, etc. Este tipo de elementos pueden favorecer la aparición de fenómenos de concentración de tensiones y por tanto deben ser tenidos en cuenta a la hora de ponderar el límite de fatiga de los materiales.

$$K_e = \frac{1}{R_f} \quad , \quad q = \frac{R_f - 1}{K_t - 1}$$

Donde R_f es el factor de reducción de la resistencia, resultado de dividir el limite a fatiga de probetas sin discontinuidad entre el límite a fatiga de la misma probeta con discontinuidad' y "q" es el factor de sensibilidad a las ranuras, que depende del material.

K_t es el factor de concentración de tensiones teórico y se calcula a través de la configuración geométrica o mediante utilización de las correspondientes tablas que serán específicas del elemento y la discontinuidad que presenten.

2.5.3.6 Factor de efectos diversos K_f.

Este factor se utilizada para modificar el límite de fatiga teniendo en cuenta los distintos factores no tenidos en cuenta en los coeficientes anteriores.

En la mayoría de los casos será necesario hacer ensayos de laboratorio para cuantificarlo adecuadamente.

Dentro de todos los posibles factores varios susceptibles de afectar al límite de fatiga alguno de los más frecuentes son:

- Tensiones residuales propias del proceso de fabricación. Como pueden ser el forjado, el laminado, etc. Si la tensión residual en la superficie es de compresión dichas tensiones pueden incrementar el límite a fatiga [4].

- Características direccionales del material. Operaciones como el laminado o el forjado puede generar una deformación del grano del materia a lo largo de una dirección en particular, esto puede propiciar que el límite a fatiga en al dirección de la deformación sea entre un 10% y un 20% superior al límite de la dirección perpendicular a dicha deformación [4].

- Defecto internos como pudieran ser el óxido durante la fundición, burbujas o partículas extrañas.

- Cementado. Este tratamiento térmico en el que se endurece la superficie del elemento mecánico puede reducir su límite de fatiga.

- La corrosión. Este fenómeno altera el acabado superficial (corrosiones leves) y puede generar fenómenos de concentración de tensiones. Si la corrosión es severa además de la concentración de tensiones puede verse disminuida la sección efectiva de la pieza.

- Los procesos de galvanizado pueden llegar a reducir el límite de fatiga hasta en un 35% [4].

- Templado superficial. Como resultado de un proceso de templado el acero puede presentar un límite a fatiga en su superficie superior al límite de fatiga alojado en capas mas internas del elemento mecánico objeto de estudio.

2.5.4 Tensiones Fluctuantes.

Las tensiones fluctuantes son aquellas en la que como indica su nombre, se producen variación de su módulo y/o sentido.

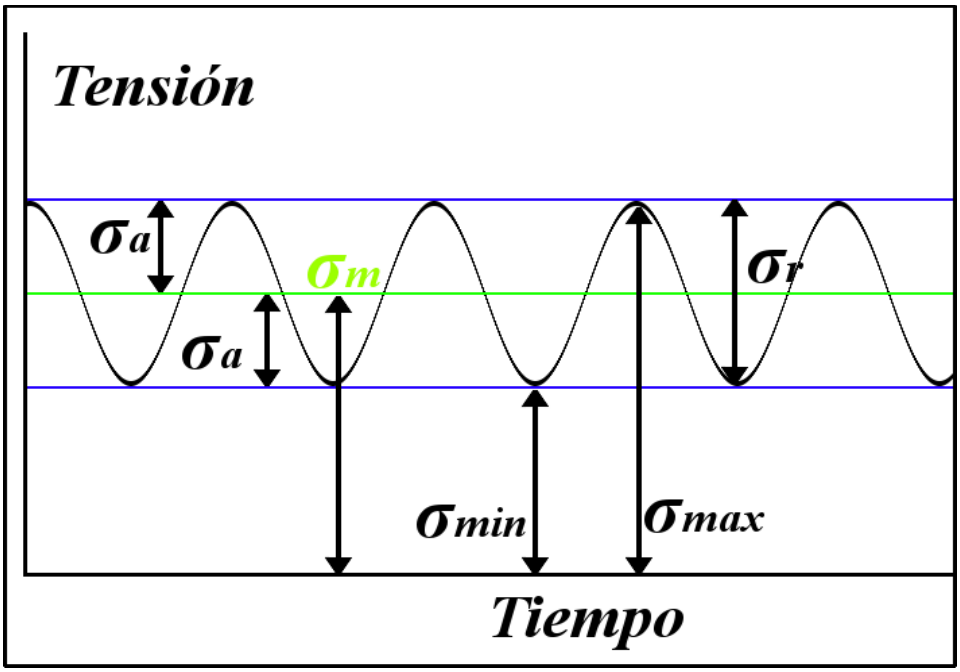

Figura 13: Tensión fluctuante.

Esta figura representa una situación genérica de un caso de tensiones fluctuantes senoidales con respecto del tiempo. En ella se pueden observar los siguientes parámetros:

- σ_a = amplitud de la tensión, tension alternante.
- σ_{min} = tensión mínima
- σ_{max} = tensión máxima
- σ_r = recorrido de la tensión.
- σ_m = tensión media

Donde:

$$\sigma_m = \frac{\sigma_{max} + \sigma_{min}}{2} \qquad \sigma_a = \frac{\sigma_{max} - \sigma_{min}}{2} \qquad \sigma_r = \sigma_{max} - \sigma_{min}$$

Se considera el estado de tensiones resultante como la contribución de un esfuerzo constante que produce la tensión media y otro esfuerzo que varía con respecto del tiempo, que origina la tensión alternante.

Por tanto a la hora de calcular la vida útil de un elemento mecánico habrá que

tener en cuenta la contribución de ambos estados de esfuerzos simultaneados.

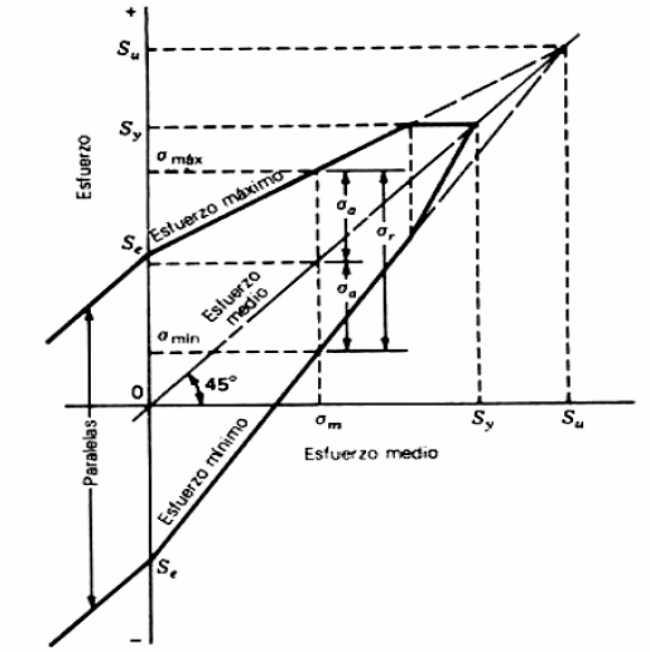

Figura 14: Tension media-máxima [1].

En esta figura se puede la variación de la tensión total en función de la tensión media.

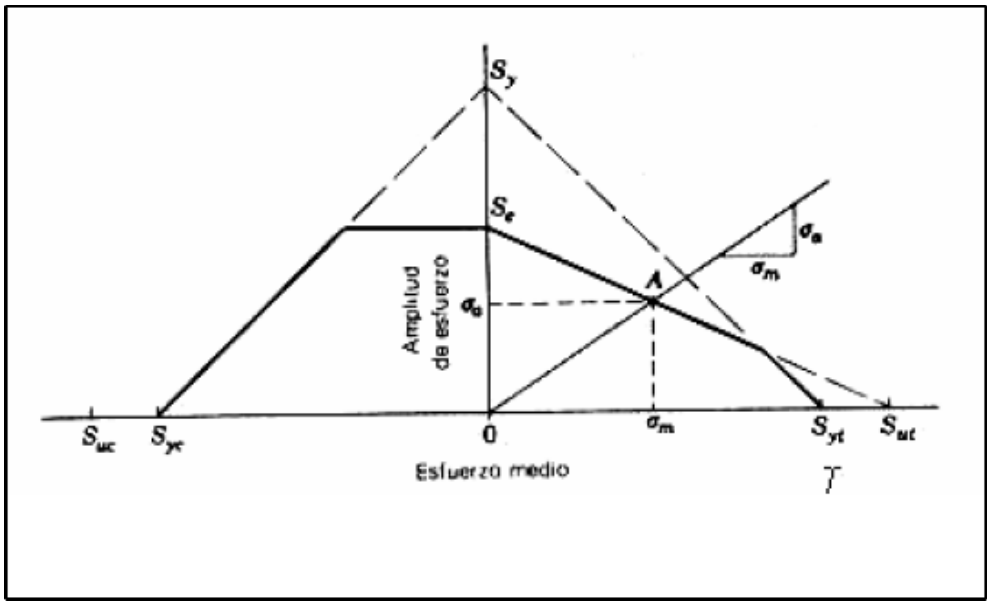

Figura 15: Tensión media-alternante [1].

Esta imagen se corresponde con el gráfico obtenido para los resultados del estudio de la resistencia de un material en el que juega con la tensión media y la alternante y en el que se produce la rotura para un número N de ciclos determinado.

A partir de múltiples ensayos y la nube de datos arrojados durante los mismos se correlacionan los datos para obtener las distintas teorías. Estas correlaciones aportan resultados aceptables, sin embargo los proceso de fatiga presentan una dispersión elevada y por tanto es recomendable la utilización de coeficientes de seguridad elevados [1].

2.5.4.1 Criterio de Goodman.

El criterio de Goodman establece que la rotura para un número determinado de ciclos se produce cuando:

$$\frac{\sigma_a}{S_f}+\frac{\sigma_m}{S_{ut}}=1$$

El criterio de Goodman proporciona los valores máximos de la tensión alternante, por tanto se suele expresar en términos de resistencia alternante:

$$\frac{S_a}{S_f}+\frac{S_m}{S_{ut}}=1$$

Este criterio se adecua correctamente al comportamiento real de los materiales bajo cargas fluctuantes y dado que el expresarlo mediante un relación supone una gran ventaja suele ser una de las teorías más usadas en la práctica [1].

2.5.4.2 Criterio de Soderberg.

Esta teoría indica que en caso de materiales dúctiles el fallo para cargas no fluctuantes se produce cuando la tensión media es igual al límite de fluencia del material. Se expresa como:

$$\frac{S_a}{S_f}+\frac{S_m}{S_{yt}}=1$$

2.5.4.3 Criterio de Gerber.

Esta teoría establece una relación parabólica entre la tensión alternante y la tensión media, tomando como vértice de la parábola el punto (0, S_f), siendo su expresión analítica:

$$\frac{S_a}{S_f}+\left(\frac{S_m}{S_{ut}}\right)^2=1$$

2.5.5 Límite de fatiga para tensiones variables.

En el caso de necesitar calcular la vida de un elemento mecánico, del que se conocen los valores de resistencia a rotura S_u y límite a fatiga S_e, sometido a esfuerzos variables habría que determinar el punto de corte entre la recta trazada por los puntos (S_{ut},0) y (σ_m, σ_a) y las curvas relativas a los criterios de Goodman o Gerber o (S_y,0) en el caso de usar el criterio de Soderberg. Una vez determinado S_f, es posible determinar el número de ciclos soportados por la pieza.

S_f puede ser determinado gráficamente o bien a partir de las siguientes expresiones:

- Goodman : $\frac{\sigma_a}{S_f}+\frac{\sigma_m}{S_{ut}}=1$, $S_f=\frac{\sigma_a}{1-\frac{\sigma_m}{S_{ut}}}=\frac{S_{ut}}{S_{ut}-\sigma_m}\cdot\sigma_a$

- Gerber: $\frac{\sigma_a}{S_f}+\left(\frac{\sigma_m}{S_{ut}}\right)^2=1$, $S_f=\frac{\sigma_a}{1-\left(\frac{\sigma_m}{S_{ut}}\right)^2}=\frac{{S_{ut}}^2}{S_{ut}^2-\sigma_m^2}$

- Soderberg: $\frac{\sigma_a}{S_f}+\frac{\sigma_m}{S_{yt}}=1$, $S_f=\frac{\sigma_a}{1-\frac{\sigma_m}{S_{yt}}}=\frac{S_{yt}}{S_{yt}-\sigma_m}\cdot\sigma_a$

Por tanto si fuera necesario calcular el límite a fatiga de un elemento sometido a cargas fluctuantes mediante las ecuaciones anteriores y la curva S-N del material, en el caso de utilizar el criterio de Goodman, el resultado sería:

$$\log S_f(10^6, \sigma_m) = \log\left(\frac{S_{ut} - \sigma_m}{S_{ut}} S_e\right)$$

3 Resortes sometidos a torsión.

3.1 Introducción.

Los muelles sometidos a torsión se utilizan en aplicaciones en las que se requiere de un elemento capaz se suministrar un par determinado como respuesta a una variación angular, como es el caso de las bisagras de puertas, mandos de cambio y frenos de bicicletas, gatillos de todo tipo, etc.

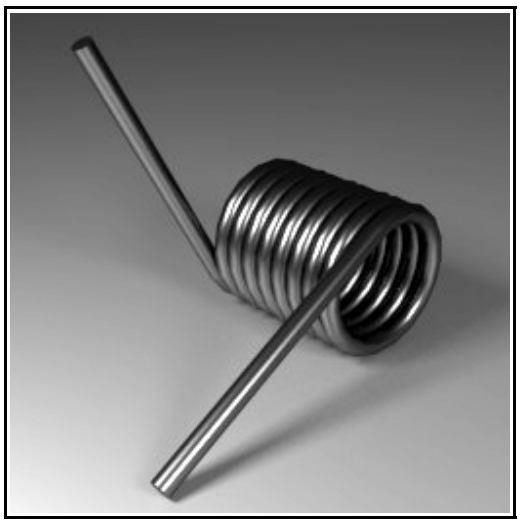

Figura 16: Resorte de torsión.

Los muelles a torsión trabajan sometidos a una tensión normal a su sección originada por un momento flector, es decir, bajo esfuerzos de compresión y tracción, a diferencia de los muelles de compresión-tracción que trabajan sometidos a esfuerzos de torsión (en los muelles de compresión el esfuerzo cortante puede ser despreciado, dado que es mucho menos significativo que el torsor).

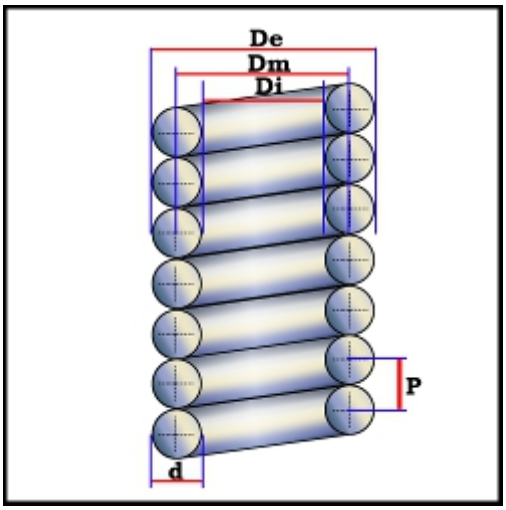

Figura 17: Cotas de resorte.

La figura 17muestra el la sección de un resorte a través de un corte de un plano que contiene a su eje, donde D_e es el diámetro exterior, D_m el diámetro medio, D_i el diámetro interior, d el diámetro del alambre y P es el paso.

3.2 Propiedades mecánicas de los resortes sometidos a torsión.

Los muelles a torsión trabajan bajo la acción de un momento flector que producirá en el seno se sus espiras un esfuerzo de tensión. Esta forma de trabajar de los resortes sometidos a torsión, a diferencia de los muelles de compresión/tracción, implica que las tensiones residuales surgidas durante el proceso de fabricación, en el que se enrolla su espiral, desempeñan un papel importante el la capacidades mecánicas del resorte.

Estas tensiones propician que el muelle sea más fuerte si trabaja de tal manera que el sentido del momento aplicado coincide con el sentido de arrollamiento de las espiras. Como consecuencia a la que tensión residual se opone a la dirección del trabajo, los resortes de torsión pueden diseñarse para trabajar con unos límites de tensión que igualen o excedan el límite de fluencia del material utilizado [4].

La tensión resultante al aplicar un momento a un resorte puede obtenerse empleando la teoría de la viga curva:

$$\sigma = K \frac{Mc}{I}$$

Donde K es el coeficiente de concentración de tensiones, no siendo en este caso considerado como un coeficiente de la reducción de la resistencia. El valor de K viene determinado por la forma del alambre y de la posición deseada para la fibra neutra. Siendo su diagrama de esfuerzos para una sección circular el siguiente:

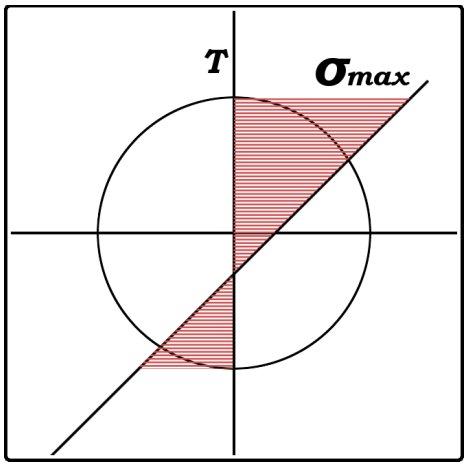

Figura 18: Tensiones en sección circular.

El valor de K, la constante de Wahl, para el caso de muelles con alambre de sección circular viene dado por la siguiente expresión:

$$K_i = \frac{4C^2 - C - 1}{4C(C-1)} \qquad K_o = \frac{4C^2 + C - 1}{4C(C+1)}$$

Donde C es el índice del muelle los subíndices "i" y "o" hacen alusión al exterior (outside) o al interior (inside) del alambre del muelle.

Para un muelle de sección circular el índice C, el índice de curvatura del resorte, es la relación existente del diámetro medio de la espira entre el diámetro del alambre:

$$C = \frac{D}{d}$$

El momento de inercia de una sección circular respecto de su centro es:

$$I = \frac{\pi d^4}{64}$$

Por tanto para el caso de una sección circular la tensión máxima viene dada por

la siguiente expresión:

$$\sigma = K\frac{32\mathrm{M}}{\pi d^3}$$

Donde K dependerá del sentido del momento aplicado como se ha mostrado anteriormente y por tanto las tensiones no sólo dependerán del módulo del momento aplicado, también dependerán de de su sentido.

Una vez conocida la tensión máxima soportada por el muelle en función del momento es necesario calcular la deformación asociada a dicho momento. Pare ello se hace uso de la energía asociada a dicha deformación:

$$U = \int \frac{M^2 dx}{2\mathrm{EI}}$$

El muelle sometido está sometido a un momento flector, originado por una fuerza F a una distancia "r" respecto del eje del muelle. Aplicando el teorema de Castigliano se puede calcular el giro asociado al momento aplicado:

$$r\theta = \frac{\partial U}{\partial F} = \int_0^{\pi DN} \frac{\partial}{\partial F}\left(\frac{F^2 r^2 dx}{2\mathrm{EI}}\right) = \int_0^{\pi DN} \frac{Fr^2 dx}{EI}$$

En el caso de un muelle de alambre circular el giro vendrá dado por la siguiente relación:

$$\theta = \frac{64\mathrm{FrDN}}{d^4 E}$$

Y la constante del muelle será:

$$k = \frac{M}{\theta} = \frac{d^4 E}{64\mathrm{DN}}$$

Que expresada en radianes queda:

$$k = \frac{M}{\theta} = \frac{d^4 E}{10{,}18\, DN}$$

Se ha comprobado experimentalmente que la constante 10,18 debe corregirse para obtener resultados más fiables [4], por tanto las constate vendrá da por:

$$k = \frac{M}{\theta} = \frac{d^4 E}{10{,}8\, DN} [Nm/rad]$$

$$k = \frac{M}{\theta} = \frac{d^4 E}{3888\mathrm{DN}} [Nm/grad]$$

Y por tanto el giro expresado en radianes vendrá dado por la siguiente expresión:

$$\Delta\alpha = \frac{10{,}8 \cdot M \cdot D \cdot N}{d^4 \cdot E}$$

O bien expresada en grados:

$$\Delta\alpha = \frac{3888 \cdot M \cdot D \cdot N}{d^4 \cdot E}$$

Por tanto la constante del muelle y el ángulo recorrido no dependen del sentido del momento aplicado, pero será necesario tener en cuenta dicho sentido a la hora de garantizar que el momento ejercido no genere tensiones máximas superiores a las toleradas por el material del resorte de torsión

Otro factor que dependerá del sentido de aplicación del momento y del giro producido por éste será el diámetro interior de las espiras del muelle, que decrecerá o crecerá dependiendo de si el sentido del giro producido por el momento es a favor o en contra del sentido de enrollamiento del resorte, dicho diámetro interior vendrá dado por:

$$D'_i = \frac{D_i \cdot N}{N + \Delta\alpha / 360}$$

Por último queda calcular la masa del muelle. Este cálculo dependerá del tipo de terminación generada en los extremos de muelle, que pueden ser de muy diversos tipos. Para el caso de muelles de hilo de sección circular con extremos rectos la masa del muelle vendrá dada por la siguiente expresión:

$$m = \delta \cdot \frac{\pi \cdot d^2}{4} \cdot (\pi \cdot D \cdot N + L_{extremos})$$

Donde δ es la densidad del material y $L_{extremos}$ es la suma de las longitudes de los extremos del resorte.

3.3 Vida útil para un resorte de torsión bajo cargas variables.

El siguiente paso, una vez conocidas la variables y características de muelle bajo esfuerzos estáticos, es determinar las condiciones necesarias para que el resorte de torsión sea capaz de tener un periodo de vida determinado bajo las condiciones de uso dadas.

La mayoría de los resortes estarán sometidos en su día a día a cargas flutuantes y por tanto, para la mayoría de los casos, es necesario dimensionarlos de tal forma que tengan vida infinita.

A la hora de calcular y dimensionar elementos mecánicos sometidos a fuerzas fluctuantes puede surgir un problema, que en la mayoría de los casos reales será imposible determinar de manera inequívoca las cargas implicadas.

Sirva el ejemplo de la suspensión de un automóvil. En este caso no será posible conocer la tensión media o la alternante ya que éstas cambiarán permanentemente en función de las condiciones del firme o del estado de carga del vehículo. En la mayoría de casos la tensión media de los muelles de un utilitario podrá ser estimada a partir de la masa del vehículo, sin embargo la tensión alternante variará mucho dependiendo de la carretera, no es lo mismo circular por una autopista que por una zona urbana repleta de badenes, por ejemplo.

El el caso de un vehículo destinado al transporte de mercancías ya no sólo variarán las tensiones alternantes sufridas por los muelle o ballestas , sino que también la tensión media variará en función de la carga.

Por tanto el muelle cuyo espectro de tensiones se quiera que quede dentro de las tensiones que no comprometan la vida infinita del mismo deberá ser calculado bajo la condiciones más desfavorables posibles, de tal forma que las tensiones sufridas no sobrepasen el límite a fatiga del material.

Otro ejemplo, en este caso para un resorte sometido a torsión, es el resorte que

acciona el desviador (cambio de los platos) en una bicicleta. Este tipo de dispositivos permiten al usuario realizar los cambios de plato en una bicicleta.

Para el caso de una bicicleta de montaña se disponen de tres platos (si bien la tendencia actual es incluir uno o dos platos), por tanto el desviador tendrá tres posiciones, cada una de ellas asociadas a un plato.

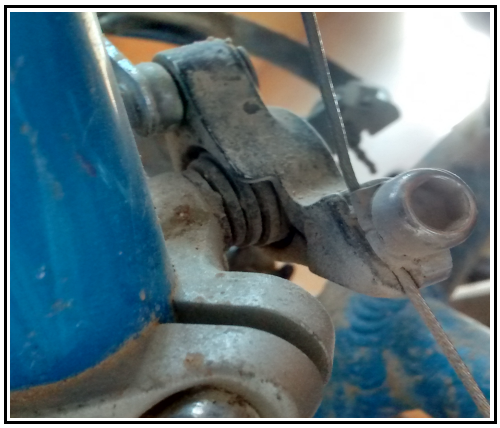

Figura 19: Desviador de bicleta.

La figura 19 muestra el muelle se torsión que acciona el desviador de la bicicleta, alojado en el interior de una bielas del paralelogramo que compone el mecanismo.

En este caso las tensiones mínima y máxima del resorte serán conocidas dado que estarán asociadas a la posición del plato pequeño y grande respectivamente.

Igualmente las tensiones alternantes serán fácilmente calculables.

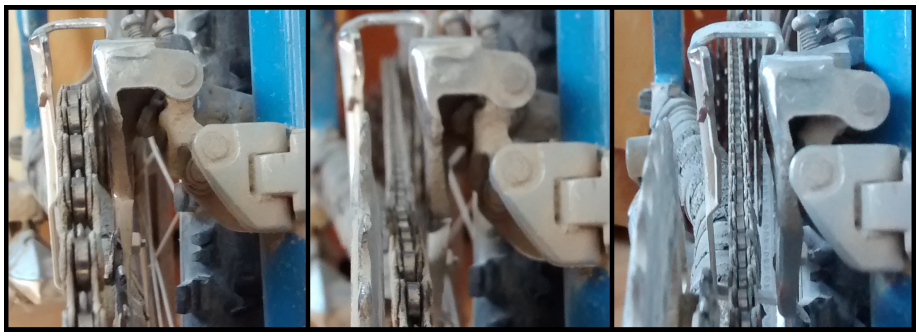

Figura 20: Posiciones del desviador.

La figura 20 muestra las tres posibles posiciones que puede adoptar el desviador de la bicicleta, cada una de ellas generará un estado de tensiones diferente.

Suponiendo el caso de una bielas con tres platos existirían 2 situaciones posibles:

- Cambio del plato pequeño al mediano y viceversa. La tensión máxima sería la asociada al plato mediano y la mínima al plato pequeño. Por tanto la tensión media y alternante podrían calcularse como:

$$\sigma_m = \frac{\sigma_{mediano} + \sigma_{pequeño}}{2} \qquad \sigma_a = \frac{\sigma_{mediano} - \sigma_{pequeño}}{2}$$

Donde $\sigma_{mediano}$ sería la tensión en el muelle para el plato mediano y $\sigma_{pequeño}$ la tensión asociada al plato pequeño.

- Cambio de plato grande al mediano y viceversa. La tensión máxima sería la asociada al plato grande y la mínima al plato mediano. Por tanto la tensión media y alternante podrían calcularse como:

$$\sigma_m = \frac{\sigma_{grande} + \sigma_{mediano}}{2} \qquad \sigma_a = \frac{\sigma_{grande} - \sigma_{mediano}}{2}$$

Siendo σ_{grande} la tensión del muelle asociada al plato grande.

Parece en principio un caso sencillo, pues en esta ocasión es relativamente fácil medir los ángulos de trabajo del muelle y conocer las tensiones medias y alternantes. Sin embargo es imposible determinar cómo el usuario final del producto va a hacer uso del mismo, no se puede determinar con exactitud con qué frecuencia se va a producir el cambio entre los platos grande y mediano o mediano y pequeño.

Este hecho repercutirá en la vida útil del muelle, ya que si se supone linealidad entre el giro del muelle y el tiro del cable (esto no es exactamente así, lo que es constante el el desplazamiento lateral del desviador, el ángulo recorrido es ligeramente

superior entre los platos pequeño y mediano) , se puede estimar que la tensión alternante será la misma en ambos casos, sin embargo la tensión media será mayor para los cambios entre el plato mediano y grande. Por tanto si se quiere determinar las características del muelle que garanticen la vida infinita del muelle habrá que suponer que todos los cambios producidos se corresponden con la situación más desfavorable.

3.3.1 Vida infinita para un resorte sometido a torsión.

El programa desarrollado generará un muelle capaz de tener una vida infinita con respecto al fenómeno de fatiga.

Para ello se debe garantizar que la tensión máxima sea igual o menor a la que garantiza vida infinita bajo esfuerzos fluctuantes, que recordemos usando el criterio de Goodman viene dada por:

$$\log S_f(10^6, \sigma_m) = \log\left(\frac{S_{ut} - \sigma_m}{S_{ut}} S_e\right)$$

La vida infinita se considera que se alcanza a partir del millón de ciclos[1], la resistencia a rotura S_{ut} es una propiedad del material usado, vendrá especificada mediante la información del fabricante o bien se obtendrá a partir de bibliografía y el límite de fatiga corregido S_e se obtendrá del límite sin corregir (a través de la información del fabricante o bibliografía) y aplicando los correspondientes factores de la ecuación de Marin.

Por tanto se dispone de una ecuación y dos incógnitas y será necesario fijar un criterio que permita relacionar la vida infinita bajo cargas fluctuantes S_f y la tensión media.

La tensión máxima nunca debe superar S_f si se quiere garantizar vida infinita luego esta será la tensión máxima tomada como admisible. El siguiente paso será determinar la tensión mínima que vendrá dada por las restricciones del mecanismo donde vaya alojado el muelle. Así que se tomará el valor de precarga o carga inicial de

trabajo como el referencia a la hora de calcular la tensión mínima.

Dado el carácter lineal de la reacción entre la deformación y el momento aplicado se puede establecer una relación entre la tensión y el ángulo recorrido, por tanto podremos expresar σ_{min} como una fracción de S_f, es decir

$$x \cdot S_f \quad \text{donde } \alpha \quad x = \frac{\alpha_{precarga}}{\alpha_{max}} = \frac{M_i}{M_f}$$

Y por tanto la tensión media en función de la precarga y la carga máxima sera:

$$\sigma_m = \frac{S_f + x \cdot S_f}{2} = \frac{(1+x) \cdot S_f}{2}$$

E introduciendo este valor en la ecuación para calcular S_f y aplicando las propiedades de los logaritmos sería posible determinar el límite de fatiga para resorte a torsión.

$$\log S_f = \log\left(S_{ut} - \frac{(1+x) \cdot S_f}{2}\right) + \log\left(\frac{S_e}{S_{ut}}\right)$$

$$\log\left(\frac{S_f}{S_{ut} - \frac{(1+x) \cdot S_f}{2}}\right) = \log\left(\frac{S_e}{S_{ut}}\right)$$

Despejando S_f se obtiene:

$$S_f = \frac{S_e}{1 + \frac{(1+x) \cdot S_e}{2 \cdot S_{ut}}}$$

El resorte tendrá vida infinita siempre que se cumpla que la tensión máxima sea igual o menor que el limite de fatiga calculado.

Para el caso de precarga nula la determinación de los ángulos inicial y final es

trivial pero para el caso de que el muelle posea una precarga habrá que determinar el ángulo entre el extremo del resorte considerado como fijo y el ángulo del otro extremo cuando el resorte se encuentra totalmente libre de carga a partir de la ecuación de la recta de pendiente “k”.

La relación entre el número de espiras N y los ángulos de trabajo de los extremos del muelle será tratada más adelante, en el apartado referente al desarrollo del macro y su funcionamiento.

Los resortes sometidos a compresión (suponiendo el caso de muelle helicoidal cilíndrico) sólo tienen una zona característica, el total de su cuerpo está formado por un alambre doblado entrono a un eje. Sin embargo en el caso de un resorte de torsión esto no será así ya que el resorte estará formado por tres partes, el extremo superior, el inferior y la parte helicoidal.

Debido a esto es necesario identificar en qué zona del resorte aparecerán las zonas críticas con respecto del fenómeno de fatiga.

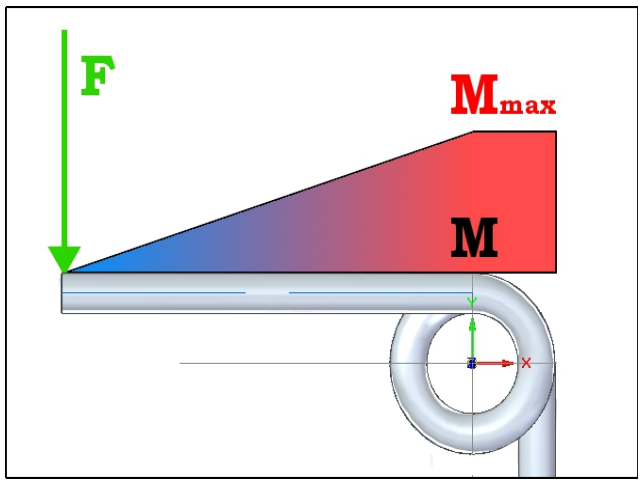

Figura 21: Momentos en resorte.

En la figura 21 se representa un caso genérico en el que a un resorte de torsión helicoidal con extremos rectos se le aplica una fuerza. El área coloreada representa el momento flector que sufre el resorte. El momento flector para el caso de una fuerza puntual viene dado por:

$$M_f = (L_e - x) \cdot F$$

Siendo L_e la longitud del extremo y "x" la distancia respecto del eje de la hélice para el eje de abscisas.

Por tanto el momento flector máximo se alcanzará en la sección del hilo del muelle que separa el extremo recto y el comienzo de la zona helicoidal y se mantendrá constante a lo largo de toda la espiral.

En el apartado relativo al cálculo de las características mecánicas de los resortes a torsión se describió la tensión máxima sufrida para un hilo de sección circular como:

$$\sigma = K_c \frac{32M}{\pi d^3}$$

Donde "K_c" es la constante de Wahl, "M" el momento flector y "d" el diámetro del alambre.

La constante de Wahl dependerá de si tomamos en consideración la cara interna de la hélice o la externa. En este caso la cara que se deberá tener en cuenta será la exterior, que será la sometida a tensiones de tracción (los esfuerzos de compresión pura no producen fatiga).

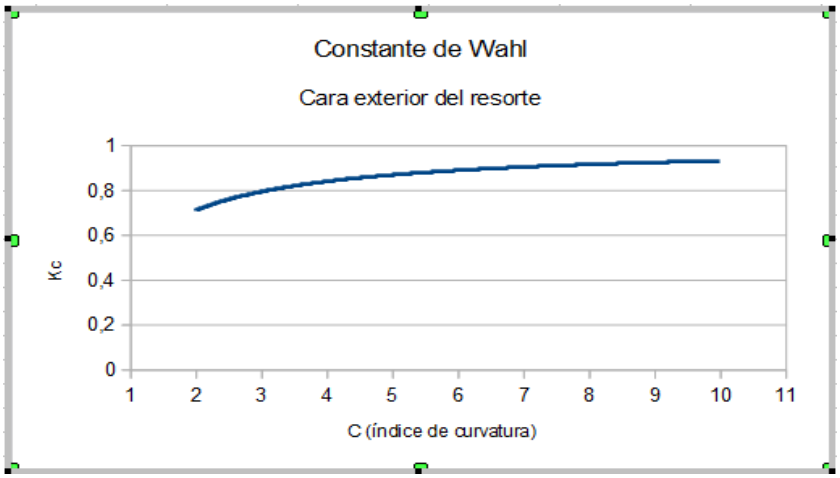

Figura 22: Constante de Wahl, cara externa.

En la gráfica relativa a la constante de Wahl respecto del índice de curvatura, para la cara externa del resorte, se aprecia como este factor reduce siempre la tensión máxima sufrida.

Para el caso de la sección recta del hilo del resorte, los extremos, el momento máximo sera el mismo que para un eje de sección circular sometido a flexión pura:

$$\sigma_{max} = \frac{32M}{\pi d^3}$$

Por tanto la tensión máxima en el resorte, que se tomará como referencia a la hora de dimensionar el muelle, será la sufrida en el comienzo de dichos extremos, en su zona más próxima a la zona helicoidal del resorte, suponiendo que la sección del hilo es constante a lo largo de toda su longitud.

Luego la tensión máxima sufrida para un resorte de torsión helicoidal que garantice la vida infinita del resorte deberá ser:

$$\sigma_{max} = \frac{32M}{\pi d^3} \leqslant S_f = \frac{S_e}{1 + \frac{(1+x)\cdot S_e}{2\cdot S_{ut}}}$$

De este resultado es sencillo deducir cuándo la tensión máxima capaz de

resistir será óptima, hecho que ocurrirá cuando "x" sea igual a menos uno, es decir, cuando la tensión mínima sea igual y de signo opuesto a la máxima y por tanto la tensión media sea nula, de tal manera que S_f sería igual al límite de fatiga corregido del material, S_e.

En el caso de que el momento o la fuerza aplicada tendiera a desenrollar la hélice la tensión que se debería tomar como referencia será la de la zona interna de la hélice.

En este caso la constate de Wahl vendrá definida por la siguiente gráfica:

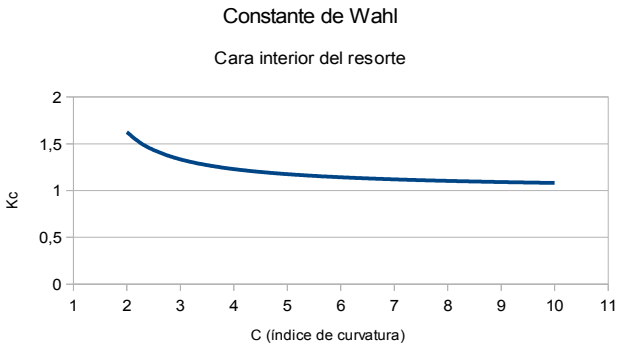

Figura 23: Constante de Wahl, cara interna.

En este caso la constante de Wahl aumentará la la tensión máxima sufrida en el área de la sección circular sometida a tracción y convertirá a la zona helicoidal en la parte del muelle que deberá ser tomada como referencia a la hora de dimensionar el resorte e implicará la necesidad de utilizar un hilo más grueso y/o resistente.

Por este motivo es recomendable, siempre que sea posible, diseñar el resorte de tal manera que los momentos a los que es sometido coincidan con el sentido de arrollamiento de su hélice.

4 Visual Basic.

4.1 Introducción.

Dado que Solid Edge carece de un entorno de programación propio, es necesario la utilización de software de terceros para el desarrollo de la macroinstrucción, siendo Visual Basic, en su versión 2010 Express el entorno de trabajo elegido y VB.Net el lenguaje usado.

El motivo de elegir este entorno de trabajo ha sido la recomendación de la guía de ayuda a la programación incluida en Solid Edge, que está orientada a trabajar a dicho entorno (si bien es cierto que una gran parte de los ejemplos dentro de la guía han sido realizados en VBA) y por tanto, dado que se parte del absoluto desconocimiento tanto de programación de macros como del resto de lenguajes recomendados para tal fin, la elección de VB.Net facilitará los primeros pasos en el desarrollo del proyecto.

A la hora de programar las operaciones y funciones relativas a las propias acciones realizables dentro de Solid Edge, como por ejemplo dibujar un boceto o acotar una pieza, la principal fuente de información ha de ser la guía de ayuda a la programación incluida en Solid Edge, en la que se incluye toda la información imprescindible para el correcto funcionamiento de cada una de las operaciones disponibles dentro de Solid Edge.

Dada la extensa información incluida en dicha guía, orientada principalmente a trabajar con Visual Basic.Net (muchos de los ejemplos están programados en VBA, que si bien es muy similar tiene algunas diferencias con respecto VB.Net), no se hace imprescindible disponer unos amplios conocimientos del lenguaje de programación utilizado y es posible dar los primeros pasos sólo con ayuda de la guía.

Sin embargo esto no quiere decir que no sea recomendable el conocer el lenguaje de programación y su entorno de trabajo. Estar familiarizado con Visual Basic facilitará sobremanera la compresión del código descrito en la guía y dotará al programador de una mayor flexibilidad y eficiencia en la elaboración del código.

La guía de programación de tareas repetitivas de Solid Edge, Solid Edge ST7 SDK, se centra en mostrar al programador la manera correcta de declarar las variables y funciones relativas al funcionamiento de Solid Edge, pero el programa generado también precisará de otra serie de funcionalidades y elementos propios de la programación informática, como puede ser el tratamiento de errores durante la compilación del código o la utilización de la herramienta, agregar funciones matemáticas para calcular parámetros, etc.

Es aquí, en la personalización e implementación de nuevas funcionalidades donde cobra especial relevancia disponer de ciertos conocimientos en el lenguaje utilizado. Esto permitirá al programador complementar la funcionalidad de Solid Edge y agregar interfaces amigables y optimizadas que permitan al usuario de Solid Edge disponer de un entorno lo más productivo posible y una experiencia de trabajo satisfactoria.

Por tanto, aún no siendo Visual Basic.Net el objetivo principal del proyecto, sino más bien el vehículo del mismo, es necesario explicar algunas sus características más relevantes y funcionalidades utilizadas.

Esto permitirá el correcto desarrollo del código y dotar al conjunto de operaciones estrictamente de diseño de la capacidad de calcular las distintas variables que rigen el comportamiento de los resortes, así como añadir una interfaz gráfica que facilite la utilización de la herramienta y se adapte a las necesidades del usuario.

En los siguientes apartados se van a describir algunos de los conceptos básicos relativos a la programación en Visual Basic así como la funcionalidad de su entorno de trabajo. Los apartados están estrictamente relacionados con los conocimientos necesarios para desarrollar la herramienta objeto del proyecto, dado que el estudio avanzado del lenguaje de Visual Basic.Net es una tarea extensa y queda fuera del objetivo principal del proyecto.

Los conocimientos necesarios en Visual Basic para el desarrollo del macro son:

- Variables: declaración de variables, tipos de variables y características de las mismas, alcance de las variables y vida de las variables.
- Los operadores: de asignación, aritméticos, binarios, de concatenación y lógicos.
- Las estructuras de control: estructuras de decisión y bucles.
- Los procedimientos y funciones.

4.2 Conceptos fundamentales en Visual Basic.

Si bien el objetivo principal del proyecto no es Visual Basic, es necesarió introducir algunos conceptos que facilitarán la comprensión del código y el desarrollo de la herramienta.

Será necesario comprender la manera correcta de declarar las variables, gestionar los posibles errores y complementar las operaciones propias de Solid Edge con un código que sea capaz de calcular los distintos parámetros del resorte antes de convertirlos en las entradas que Solid Edge tomará para definir el objeto diseñado mediante el marco. También será necesario diseñar un modo de entrada de esos datos, para permitir al usuario introducir las características del muelle deseado.

Si bien hay disponible opciones comerciales dentro de la oferta del catálogo de Microsoft, Visual Basic 2010 Express ofrece todas las funcionalidades requeridas para el correcto desarrollo de macros para Solid Edge, siendo esta versión (en el momento de desarrollo del proyecto) la recomendada por la guía de programación incluida en Solid Edge.

4.3 Las variables en Visual Basic.Net

Una variable es una entidad usada en programación que permite el almacenamiento de información durante el funcionamiento del programa. Es necesario asignar el tipo de variable utilizada antes de que su uso se produzca dentro del código, si bien no es necesario asignarle un valor antes de dicho momento [3].

4.3.1 Nombre de las variables

El nombre de la variable puede estar compuesto por hasta un total de 1023 caracteres y admite tanto letras como números y también admite el uso del guión bajo "_".

Durante la declaración de variables conviene tener en mente que Visual Basic no distingue entre mayúsculas y minúsculas, por tanto interpretará como iguales dos o más variables cuya única diferencia en el nombre sea el uso de mayúsculas o minúsculas en alguna de sus letras.

No es posible utilizar las palabras propias del código como el nombre para una variable como pudieran ser "Dim", "If", "Sub", etc. En el caso de necesitar nombrar de esta manera a una variable será necesario introducirla dentro de corchetes [ ].

4.3.2 Tipos de variables

Atendiendo a las características y necesidades de las variables y constantes utilizadas será necesario asignarles el tipo de datos que van a albergar. Visual Basic ofrece la posibilidad de no declarar el tipo de variables utilizadas, siendo posible que el programa determine el tipo de variable, sin embargo esta práctica no es recomendable.

Los tipos de datos disponibles son los siguientes:

Tipo	Tamaño	Valores
Boolean	2 bytes	True / False

Date	8 bytes	0:00:00 1 de enero de 0001 a 11:59:59 31 de diciembre de 9999
Decimal	16 bytes	Número decimal, positivo o negativo. 29 dígitos
Double	8 bytes	Número decimal positivo o negativo Rango de -1.79769313486231570E+308 a 1.79769313486231570E+308
Integer	4 bytes	Número entero Rango desde -2,147,483,648 hasta 2,147,483,648
Long	8 bytes	Número entero Rango desde -9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,808
Object	4 bytes	Cualquier tipo se puede almacenar en una variable de tipo Object
Short	2 bytes	Número entero
Single	4 bytes	Número decimal Rango desde -3.4028235E+38 hasta -3.4028235E+38
String	variable	Cadena de caracteres Unicode
Char	2 bytes	Un carácter Unicode
Byte	1 byte	Un valor positivo sin signo,para contener datos binarios

La correcta asignación del tipo de variable cobrará especial relevancia a la hora de optimizar el uso de memoria en el sistema, dado que aunque pudiera parecer despreciable en códigos pequeños con pocas variables y/o constantes, la memoria ocupada de manera innecesaria podría suponer un problema en el caso de uso de matrices con muchos elementos[3]. Por tanto se hace necesario elegir el tipo de variable que mejor se adecue a las necesidades reales del programa, no tiene sentido por ejemplo, utilizar variables de tipo decimal si el programa sólo va a trabajar con

números enteros de dos dígitos en donde una variable de tipo “Short” sería una elección mucho más adecuada.

Como se mostrará mas adelante, el tipo de variables para el código relativo a la automatización de tareas en Solid Edge vendrán impuestas por las necesidades del programa, no dando lugar a la elección de las mismas.

4.3.3 Visibilidad de las variables.

El concepto de visibilidad de una variable hace referencia a la parte del código en el que estará accesible la variable declarada. El alcance dependerá de la parte del código en la que la variable haya sido declarada y la palabra clave que la acompañe.

- Ámbito a nivel de bloque: la variable declarada dentro de un bloque sólo será utilizable por el código del mismo y debe ir acompañada por la palabra clave “Dim”.

- Ámbito a nivel de módulo : la variable se declara fuera de cualquier procedimiento o función y por tanto la variable estará disponible en todo el módulo. Un código se declara como a nivel de módulo cuando esta insertado entre las palabras clave Module y End Module.

- Ámbito a nivel de procedimiento: será la variable declarada dentro del código insertado en un procedimiento, es decir el código existente entre las palabras clave Sub y End Sub.

4.3.4 Nivel de acceso de las variables.

El nivel de acceso de las variables permite configurar las partes del código en las que se podrá leer o modificar las variables declaradas en el código del programa.

Dependiendo del nivel requerido se podrán declarar las variables atendiendo a las siguiente categorías:

- Public: las variables serán accesibles desde cualquier parte del código en el que hayan sido declaradas. No es posible declarar como Public una variable que vaya a ser utilizada dentro de una función o de un procedimiento.

- Friend: las variables serán accesibles desde cualquier parte del ensamblado en el que hayan sido declaradas. No es posible declarar como Friend una variable que vaya a ser utilizada dentro de una función o de un procedimiento.

- Protected: la variables declaradas como Protected sólo estarán disponibles dentro de una Clase.

- Protected Friend: las variables declaradas como Protected Friend presentan un nivel de acceso que combina las propiedades de los dos niveles anteriores,por tanto será accesible desde el ensamblado en que hayan sido declaradas y todas las clases que heredan de la clase en la que han sido declaradas.

- Private: las variables declaradas como privates sólo estarán disponibles para la clase,la estructura o el módulo en el que hayan sido declaradas. No es posible declarar como Private una variable que vaya a ser utilizada dentro de una función o de un procedimiento.

4.3.5 Ciclo de vida de las variables.

El ciclo de vida de una variable permite establecer hasta qué momento una variable estará disponible dentro del código.

4.4 Operadores.

Los operadores en Visual Basic son el conjunto de palabras claves o símbolos que permiten realizar operaciones y comparaciones entre los distintos componentes del código del programa.

El uso de los operadores permitirá, entre otras funciones, realizar las operaciones matemáticas necesarias para calcular los distintos parámetros que describen a los resortes de torsión helicoidales.

Atendiendo a su funcionamiento pueden ser clasificado en seis grupos diferentes:

- Lógicos: los operadores lógicos permiten añadir o modificar las condiciones de las estructuras de bucle o condición. El grupo de los operadores lógicos está compuesto por las palabras clave “and”, “or”, “xor”, “not” , “andalso” y “orelse”.

- Operadores binarios. Estos operadores son exclusivos de las variables de números enteros (Integer, Short, Long y Byte) [3].

- Operadores aritméticos. Permitirán realizar realizar las distintas operaciones matemáticas incluidas en el código.

- Operadores de comparación. Permitirán hacer comparaciones entre las distintas variables declaradas en el código. Se usarán típicamente en las estructuras tipo “If” o en estructuras de bucle. Dan como resultado un valor de tipo booleano.

- Operador de asignación. Sirve para fijar la información albergada en una variable. La información ha de ser coherente con el tipo de variable , es decir, si la variable es de tipo “Integer” se le debe asignar un número entero o si es de tipo “String” se le deberá asignar una cadena de caracteres. Dicho operador viene representado por el símbolo igual, = .

- Operador de concatenación. Sirve para concatenar cadenas de caracteres.

4.5 Estructuras de control.

4.5.1 Introducción

Las estructuras de control permiten al programador establecer qué orden o bajo qué circunstancias debe de ejecutarse un bloque del código. Ésto permitirá seleccionar una opción, que conlleve una acción determinada, entre los los distintos casos posibles para la tarea que se quiera llevar a cabo. También permitirá realizar una acción un número determinado de veces.

La justificación de la inclusión de este apartado en el documento surge de la necesidad de la necesidad de realizar operaciones de iteración asocias a la resolución de los problemas de fatiga y se centrará exclusivamente en el tipo de estructuras utilizadas.

4.5.2 Estructura de control If.

Esta estructura permite habilitar o deshabilitar un bloque en función de que se cumpla o no una determinada condición impuesta durante la redacción de la estructura "If". El valor devuelto será de tipo booleano (True, False).

Se disponen de cuatro tipos de sintaxis para la estructura "If" [3], que pueden ser sintetizadas una sola para ilustrarlas:

If Condición Then

Primera acción
….
N ésima acción

Else
Primera acción
….
N ésima acción

End If

Es decir *"Si se cumple la condición, entonces" "ejecutar acción 1, acción 2,...) "Si no" "Ejecutar alternativa 1, ejecutar alternativa2, ….", " Terminar*

estructura"

Por tanto se pueden agregar tantas acciones como sean necesaria, tantas alternativas como sean necesaria (si no son necesarias se puede eliminar la palabra "Else" en cuyo caso si no se cumple la condición el programa al ejecutarse se "saltará" el bloque).

Las acciones y alternativas a su vez pueden incluir estructuras "If", hecho que se conoce como anidamiento [3].

4.5.3 Estructura de bucle "While".

La estructura de bucle permite realizar un proceso tantas veces como sea necesario. También permite fijar el número de veces que se necesita realizar una acción.

En el caso de la aplicación o macro desarrollado en el proyecto será necesario añadir una estructura de bucle en varias ocasiones, como por ejemplo a la hora de calcula el factor de tamaño K_b o el diámetro normalizado del hilo.

La estructura de bucle utilizada será la estructura "While" que permite realizar una operación mientras se siga cumpliendo una condición fijada.

La sintaxis es similar a la sintaxis de la estructura "If", como se muestra a continuación:

```
While Condición_Impuesta
    Acción deseada
End While
```

Sólo se ejecutará si la condición se cumple desde el principio, si no el programa se saltará el bloque [3].

4.6 Funciones y procedimientos.

En Visual Basic es necesario que todas las instrucciones vayan incluidas dentro de un procedimiento o función, esto permitirá al programador generar bloques que podrán ser utilizados más de una vez por el código, para lo que simplemente será necesario llamarlos.

La división del código en bloques facilitará el desarrollo del mismo, ya que este hecho permitirá acotar los errores del código a un bloque en concreto, siendo por tanto una práctica muy recomendable.

Visual Basic dispone de cinco tipos distintos de procedimientos [3]:

- Procedimiento de evento. Son los procedimientos activados cuando se produce una acción concreta, como puede ser pulsar una tecla o presionar con el puntero del ratón un botón.

- Funciones. Ejecutan un bloque y devuelven un valor como resultado.

- Procedimientos Sub. Ejecutarán un bloque cuando sean llamados. En el caso del macro para el cálculo y diseño de resortes las instrucciones propias de Solid Edge irán alojadas en un procedimiento Sub.

- Procedimientos operador. Modifican el funcionamiento de un operador.

- Procedimientos property. Modifican las propiedades de un objeto creado en la aplicación.

4.7 Excepciones.

A la hora de desarrollar un programa es importante añadir un método que permita disponer a la aplicación de una alternativa en caso de que se produzca un error durante su ejecución y dotar a este método de una herramienta que permita a la aplicación transmitir la información pertinente acerca del error ocurrido.

De entre los múltiples métodos disponibles para el caso del desarrollo del macro para Solid Edge se hará uso del método "Try", que permitirá al programa ejecutar las instrucciones relativas a Solid Edge y mostrará a través de una ventana de consola el código relativo al error ocurrido.

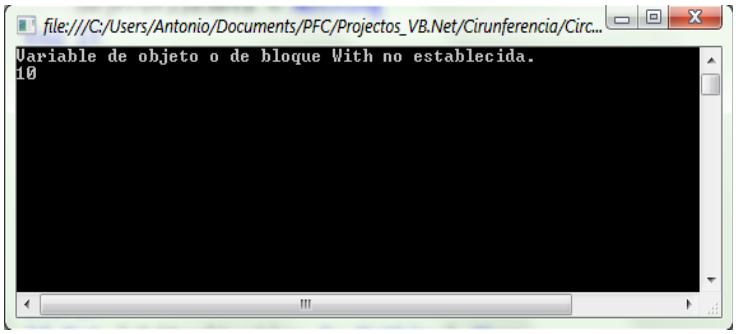

Figura 24: Excepción del macro.

Al producirse una excepción o error aparecerá una ventana como la de la figura 24.

La sintaxis del bloque "Try" se describe a continuación:

```
Try
Instrucciones a realizar
Catch
Instrucciones a realizar si se produce un error.
Finally
Instrucciones a realizar se produzca o no un error.
End Try
```

5 Solid Edge.

5.1 Introducción.

En este capítulo se describen las características básicas del software para diseño asistido por computador elegido, Solid Edge, así como los pasos necesarios para la realización y utilización de un Macro para Solid Edge.

Solid Edge es un programa informático de diseño asistido por computador (CAD en sus siglas en inglés), principalmente orientado al diseño de piezas en 3D.

Solid Edge dispone de varios entornos diferenciados de trabajo (entorno pieza, chapa, plano, conjunto y soldadura) pudiéndose considerar su entorno "pieza" como el principal. También permite realizar simulaciones, renderizados y animaciones.

La manera de trabajar en Solid Edge es análoga a la manera en la que se fabricaría una pieza el la realidad, en donde a partir de un bloque o forma inicial se irían aplicando una secuencias de operaciones (torneado, taladrado, mecanizado, roscado, etc) hasta llegar al resultado final. En Solid Edge la manera de trabajar en el entorno pieza sería análoga, generalmente a partir de un perfil en dos dimensiones se genera un objeto en tres dimensiones por medio de una operación de extrusión (recta, de revolución, helicoidal, etc) mediante lo cual se obtiene la forma básica de la pieza, a la cual se le realizará una secuencia de operaciones de adhesión o retirada de material hasta llegar al estado definitivo.

La herramienta a desarrollar modelará de manera automática, dentro del entorno de Solid Edge, un muelle una vez conocidas las variables, sin que sea necesario la utilización de las herramientas propias de trabajo incluidas en Solid Edge por parte del usuario.

A la hora de realizar un macro es necesario el conocimiento de las distintas herramientas y métodos de trabajo necesarios a la hora de diseñar los elementos dentro del entorno de Solid Edge.

A que a la hora de realizar un macro la manera de proceder será análoga a la forma en la que se diseñaría una pieza o se realizarían una serie de operaciones en una pieza, siendo necesario programar todos los pasos en el mismo orden en el que se harían si la pieza, o las operaciones, se estuvieran realizando a través de la interfaz de trabajo dentro del entorno de Solid Edge.

Por tanto, antes de iniciarse en la programación para la realización de macros en Solid Edge, es necesario tener los conocimientos pertinentes en su manejo dentro de su entorno. Para realizar un macro que automatice un proceso dentro de Solid Edge (o cualquier otro software de diseño asistido por computador), el primer paso es conocer la manera de proceder a la hora de realizar dicha pieza u operaciones con las herramientas de diseño incluidas por defecto en el programa, siendo incluso más importante a la hora de programar el conocimiento de Solid Edge que el del lenguaje de programación elegido.

Solid Edge incluye varios entornos de trabajo bien diferenciados que se adaptan a las diferentes necesidades existentes dentro de la disciplina del diseño asistido por computador, así como un amplio repertorio de complementos disponibles a través de la web del producto.

La versión utilizada del software en este caso es Solid Edge ST7 bajo licencia académica, que incluye todas las funcionalidades y entornos de trabajo disponibles para la versión comercial, es decir, el entorno Pieza, Soldadura, Conjunto, Chapa y Plano.

Es posible desarrollar macros específicos para cada uno de estos entornos, si bien en este caso el proyecto se centrará en el entorno Pieza a la hora de desarrollar la herramienta que calculará y diseñará los resortes.

El programador necesitará estar familiarizado con el entorno de trabajo correspondiente para la correcta creación del macro desarrollado.

Dentro del entorno conjunto se encuentra una herramienta que sirve como referencia para el desarrollo del macro objeto del proyecto, el Engineering Reference, ubicado el la barra de herramientas del lateral derecho:

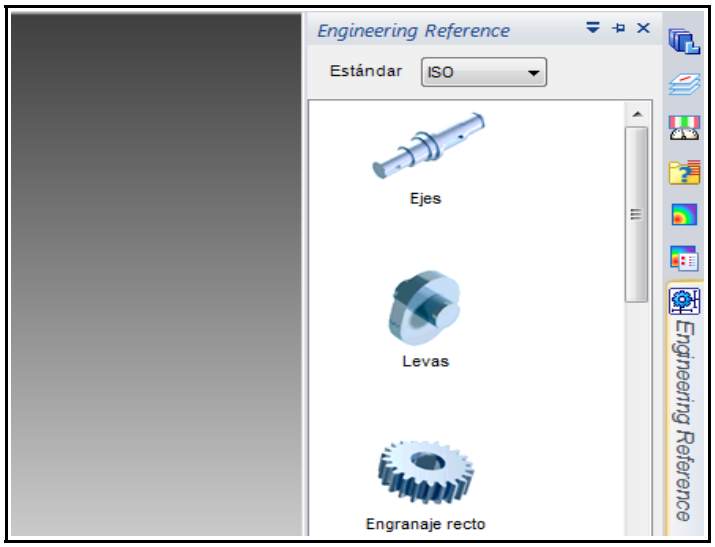

Figura 25: Engineering Reference.

Engineering Reference permite al usuario disponer de una herramienta capaz de diseñar algunos de los elementos mecánicos más utilizados en ingeniería a partir de los requisitos introducidos mediante el uso de una interfaz gráfica, no siendo necesario por parte del usuario realizar ningún tipo de operación de diseño. Una vez introducidos los requisitos, Solid Edge de manera automática creará el modelo en tres dimensiones que se ajuste a las necesidades del usuario. Engineer Reference permite obtener el diseño de piezas normalizadas para engranajes rectos, engranajes cónicos, levas, ejes, resortes de compresión, resortes de tracción, engranaje de tornillo sin fin, ruedas dentadas, vigas, columnas, poleas y poleas síncronas.

Esta herramienta facilita sobremanera el diseño de este tipo de piezas, que en ocasiones puede ser especialmente laborioso, como podría ser el caso del diseño de engranajes normalizados y por tanto resulta de gran utilidad para el usuario.

Engineering Reference es una herramienta de gran utilidad de cara al diseño orientado al mundo de ingeniería e ilustra las bondades de la aplicación de macros dentro del entorno del diseño asistido por computador y para Solid Edge en particular.

Dentro de Engineering Reference no está incluida la posibilidad de diseñar y modelar resortes helicoidales de torsión, por tanto la herramienta objeto del proyecto

viene a satisfacer una necesidad no cubierta por esta utilidad.

5.2 Programación de macros para Solid Edge.

5.2.1 Introducción.

En informática la palabra Macro hace referencia a una macroinstrucción que, en definitiva, es una secuencia de instrucciones almacenadas para posibilitar la ejecución organizada de una orden. Es decir, cuando se ejecuta un macro dentro de una aplicación informática se obtiene como resultado la de una serie acciones realizadas de manera secuencial conforme al orden preestablecido en el código del Macro.

La utilización de macros en las aplicaciones informáticas permite dotar de nuevas herramientas o utilidades a las mismas, así como la automatización de tareas, hecho que puede aumentar significativamente la productividad en la realización de tareas repetitivas.

En la mayoría de los casos, aunque el software en cuestión admita la utilización de macros, será necesario la utilización de software de terceros para la realización de los mismo, así como conocimientos necesarios del lenguaje de programación utilizado.

Dentro de la instalación de programa de Solid Edge es posible encontrar ejemplo de macros de libre distribución creados para Solid Edge. Dichos macros se ubican en la carpeta de instalación del programa "Archivos de Programa/Solid Edge ST7/ Custom" [6].

Estos macros incluyen tanto el archivo ejecutable como el código del programa preparado para abrirlo en su correspondiente entorno de programación.

Además de estos ejemplos Solid Edge incluye diversas ayudas que permiten al usuario nobel aprender a trabajar con las distintas herramientas incluidas, así como familiarizarse con las nuevas utilidades y funcionalidades incluidas en cada nueva versión.

Dentro de esa ayuda está disponible la Guía de Programación en Solid Edge,

Solid Edge ST7 SDK, accesible únicamente a través de internet, lo cual hará imprescindible una conexión a internet, especialmente durante los primeros pasos en la programación de macros.

La guía está orientada a trabajar en Visual Basic.Net, si bien dado las similitudes con C# el usuario experto en este lenguaje no debería experimentar grandes dificultades si usa este último en detrimento de VN.Net para el desarrollo de los macros.

La guía incluye todo lo necesario para iniciarse en la programación de macro, incluyendo la descripción de cómo activar y utilizar todas y cada una de las herramientas y funcionalidades de Solid Edge. La información incluida se centra exclusivamente en las herramientas propias de Solid Edge, y aunque incluye una pequeña introducción de cómo empezar a trabajar en Visual Basic es recomendable disponer de cierto grado de conocimiento del lenguaje de programación elegido que, aunque no siendo imprescindible, facilitará la tarea de aproximación al mundo de la programación de macros para Solid Edge.

No menos importante será el conocimiento del funcionamiento de Solid Edge mediante el método tradicional, es decir, mediante la interfaz del programa. El paso previo imprescindible, sin el cual la programación de macros no tiene sentido, es el correcto dominio del entorno de trabajo de Solid Edge por parte del programador. Antes de realizar cualquier tipo de operación mediante el uso de macros será imprescindible conocer la manera de realizar dicha acción mediante las herramientas propias de Solid Edge.

El correcto conocimiento de las herramientas de Solid Edge, entre otras cosas, permitirá a programador diferenciar los errores cometidos en el código de los errores cometidos en la manera de realizar una operación en Solid Edge

Por este motivo el primer paso previo, siempre que sea posible (puede no serlo en caso de añadir una nueva funcionalidad a Solid Edge) será el realizar el proceso que se quiera automatizar mediante un macro a través de la interfaz propia de Solid Edge y estudiar la manera óptima de realizar dicho proceso. Hecho ésto será el momento de comenzar a escribir el código del macro.

5.2.3 Macros.

Dentro del amplio abanico de herramientas de que dispone Solid Edge en la interfaz de su entorno de trabajo, se ofrece la posibilidad de complementar su funcionalidad por medio de macros, bien fijándolos dentro de su barra de tareas, o bien ejecutándolos a través de archivo ".exe" del Macro en cuestión.

Para generar automatizaciones o macros en Solid Edge es posible utilizar cualquier software de terceros capaz de trabajar con el modelo de automatización Microsoft COM, por tanto, como se ha comentado anteriormente, es posible generar macros en múltiples lenguajes que soporten dicha característica.

COM (siglas en inglés de Component Object Model) es un sistema que permite la comunicación entre procesos y la creación de objetos[9].

La generación de macros tiene como objetivo principal complementar la funcionalidad del entorno de trabajo de Solid Edge, permitiendo dotar al mismo de nuevas funcionalidad, automatizar todo tipo de secuencia de procesos y operaciones ya incluidos en Solid Edge e incluso simplificar el uso de las herramientas propias de Solid Edge.

Por tanto queda patente el potencial de la programación de macros como herramienta aplicada al diseño asistido por computador, dado el amplio abanico de posibilidades que ofrece.

Sin embargo la aproximación al la programación de macros para Solid Edge puede resultar un tanto desalentadora dada la dificultad para encontrar bibliografía al respecto, pues la mayoría de títulos disponibles se centran en el modelado mediante la interfaz gráfica de Solid Edge. La guía incluida de ayuda a la programación, si bien incluye toda la información estrictamente necesaria, no siempre resultará de fácil comprensión, a lo que hay que sumar el hecho de que aglutina toda la información disponible a lo largo de todas las versiones existentes de Solid Edge lo que en ocasiones será causa de confusión dado que las variables de la operación que se esté

automatizando no se definan exactamente igual en el ejemplo y la versión de Solid Edge utilizada.

Este hecho relega la creación de macros a usuarios avanzados (aun cuando los primeros pasos puedan no ser muy complicados) cuyo objetivo, principalmente, sea el de generar aplicaciones comerciales que satisfagan las necesidades específicas de los clientes. Para el resto de casos el método de trabajo "tradicional" será mucho más indicado, dado que Solid Edge ofrece un entorno de trabajo amigable, de fácil comprensión y la bibliografía disponible para su aprendizaje es abundante.

5.2.3 Macros para Solid Edge.

El primer paso para automatizar un proceso dentro de Solid Edge es conocer la secuencia de pasos necesarios dentro del entorno de trabajo del programa, para posteriormente tratar de reproducir la misma secuencia de operaciones dentro del código del Macro.

Para ello, una vez conocido el método a seguir para diseñar la pieza, el primer paso es iniciar el entorno de trabajo elegido para escribir el código del macro, en este caso Visual Basic.

Visual Basic ofrece varios modelos de plantillas a partir de los cuales comenzar a trabajar, también ofrece la posibilidad de seleccionar nuestra propia plantilla si fuera necesario. La plantilla "Aplicación de consola" será suficiente a la hora de iniciarse en la programación de macros para Solid Edge.

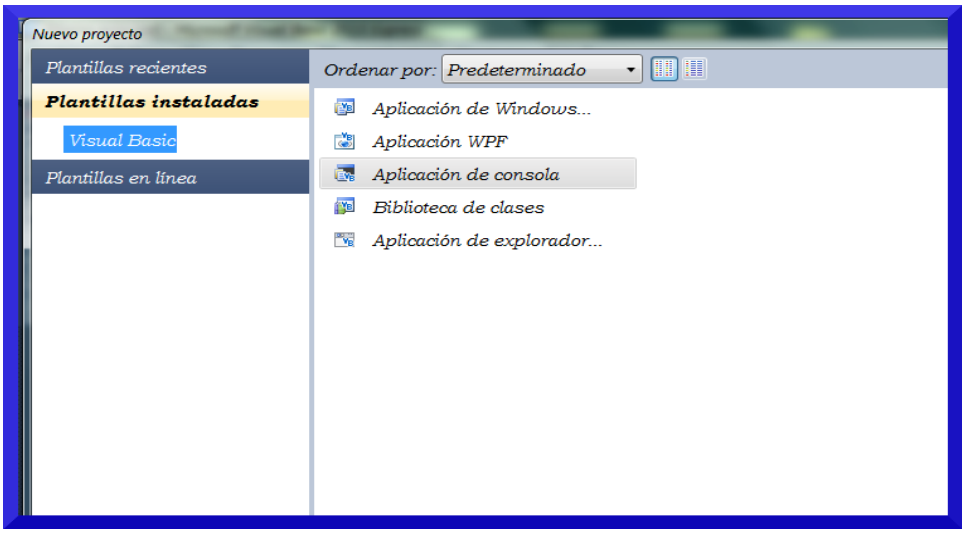

Figura 26: Plantillas de Visual Basic.

Una vez abierto el entorno de trabajo, con nuestra plantilla seleccionada el primer paso a seguir será guardar nuestro proyecto y habilitar dentro de la carpeta creada para tal fin la creación de una carpeta en la que guardar la solución (el archivo ".exe" del macro generado).

Seguidamente se debe agregar las correspondientes referencias a las librerías de Solid Edge. Para ello en el menú hay que desplegar la pestaña de "proyecto" y seleccionar agregar referencia.

Si bien este paso no es imprescindible hacerlo hasta justo el momento antes de compilar el código, resulta altamente recomendable llevarlo a cavo, de lo contrario Visual Basic no será capaz de reconocer los elementos propios de Solid Edge y los reconocerá como errores, lo que dificultará sobremanera la detección de problemas en nuestro código. Además, una vez agregadas la referencias, dado que Visual Basic reconocerá los elemento de Solid Edge, se dispondrá de la ayuda ofrecida por IntelleSense, herramienta de gran utilidad, especialmente para programadores noveles, dado que no siempre será sencillo encontrar el código necesario para la operación que se desee realizar. IntellSense mostrará todas las opciones disponibles para cada elemento del código a medida que se vaya redactando el código, pero no todas opciones producirán un resultado coherente, así que debe ser usado con cautela.

Las referencias necesarias, obviamente, dependerán del entorno de Solid Edge (Pieza, Plano, etc) en el que se vaya a trabajar, así como de las operaciones a realizar. Durante los primeros pasos el la programación de macros puede resultar útil agregar todas las referencias (esto no generará ningún error en la compilación), si por el contrario obviamos agregar alguna referencia necesaria para nuestro macro no será posible compilarlo, dado que Visual Basic no será capaz de reconocer los elementos propios de Solid Edge y por tanto no generará el archivo ejecutable. Las referencias se pueden agregar en cualquier momento antes de generar la solución. También existe la opción de retirar todas las referencias sin usar una vez finalizado el código (menú/proyecto/propiedades/referencias).

Una vez completado este paso se deberá escribir el código que lleve a cabo las operaciones necesarias para completar la tarea.

El siguiente paso a seguir será generar el archivo ejecutable. Para ello hay que seleccionar en el menú "generar solución" dentro de la pestaña desplegable "generar", tras lo cual el archivo ejecutable aparecerá dentro de la carpeta "bin/Debug" bajo el nombre que se haya asignado.

Una vez creado el macro ya podrá utilizarse dentro de Solid Edge. La manera de ejecutar el macro dentro de Solid Edge es abrir el programa en el entorno correspondiente para el que el macro ha sido creado y desplegar el menú. Con el menú desplegado estará disponible la opción "Ejecutar Macro". Dentro será necesario buscar la carpeta en la que se guardó el proyecto, en donde estará disponible el archivo ".exe" para su ejecución.

Si el macro ha sido realizado correctamente, el usuario podrá observar cómo se desencadena de manera automática la secuencia de operaciones que se hubiera llevado a cabo si la pieza (u operación) se hubiera realizado utilizando la interfaz de Solid Edge.

El proceso de automatización de tareas repetitivas en Solid Edge mediante la utilización de macros puede suponer un reto importante para el programador novel, además ,si bien es cierto que no se precisa un excelso dominio del lenguaje de programación elegido, es altamente recomendable poseer unas nociones básicas del

mismo.

Dentro de la guía de ayuda a la programación se encuentra el código (en diversos lenguajes de programación, siendo el más extendido VBA) de todas los operaciones disponibles dentro del entorno de trabajo de Solid Edge, y su uso será necesaria para correcta declaración de variables y creación del código.

Dada la gran cantidad de posibles operaciones disponibles en Solid Edge el dominio de la automatización de tareas en Solid Edge requiere de un largo periodo de aprendizaje, siendo posible necesitar años para desenvolverse con soltura.

Queda patente pues, que la creación de macros para Solid Edge solo supone un ventaja en aquellos procesos que la repetitividad sea su característica más evidente y sea preciso realizarlos frecuentemente, como por ejemplo cálculo de tornillos, conversión a PDF o acotación, todos ellos ejemplos de macros para Solid Edge ya disponibles[6]. En resto de casos el método "tradicional" será mucho más eficiente.

5.2.4 Ejemplo de programación de un macro para Solid Edge.

A continuación se ilustran los primeros pasos para la creación de un macro en que se automatizan una serie de operaciones típicas de Solid Edge, comparándolo con la manera de proceder en el diseño de una pieza mediante la utilización de la interfaz de Solid Edge en su entorno pieza.

Es posible incluir en el código la acción de iniciar Solid Edge y abrir el entorno deseado [5], pero en este el ejemplo comenzará una vez el entorno Pieza de Solid Edge esté ejecutándose.

En primera instancia se crea un boceto. En el boceto se añade un círculo de radio quince milímetros y centrado en el origen de coordenadas (0,0) del plano de referencia elegido.

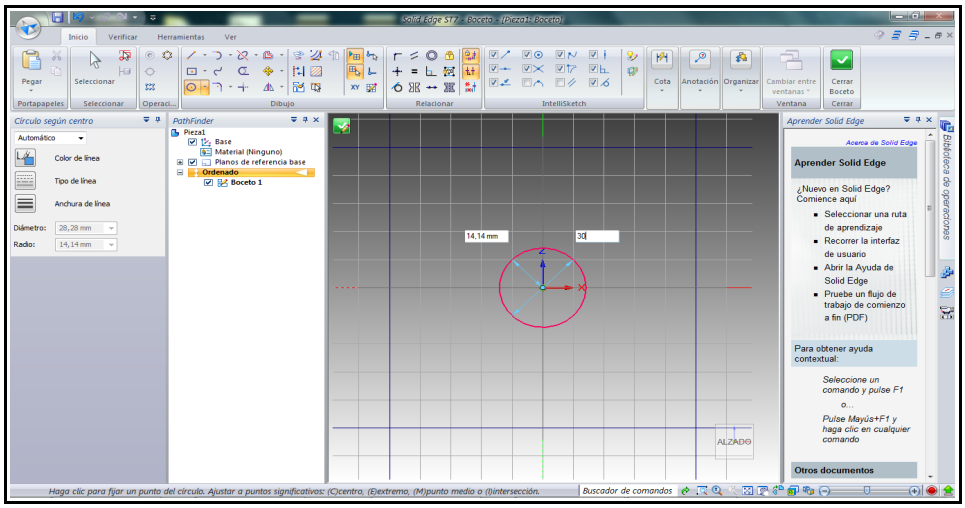

Figura 27: Boceto circular.

Se presiona aceptar y se obtiene el círculo de radio quince milímetros centrado en el punto seleccionado.

Para conseguir este mismo resultado mediante un macro el primer paso, como se ha comentado en el apartado anterior, es abrir el entorno de programación y agregar las correspondientes referencias (Menú/Proyecto/Agregar Referencia) y se desplegará la siguiente ventana:

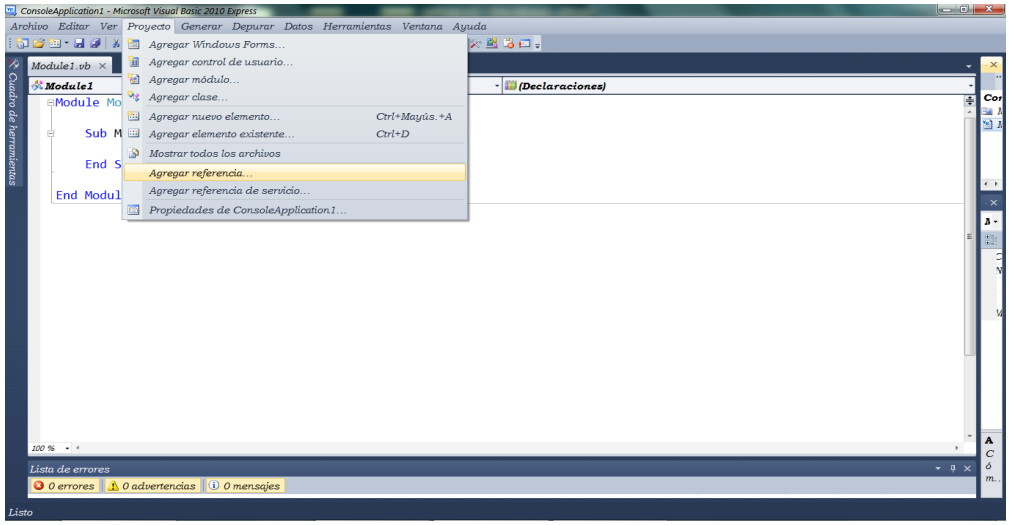

Figura 28: Agregar referencias.

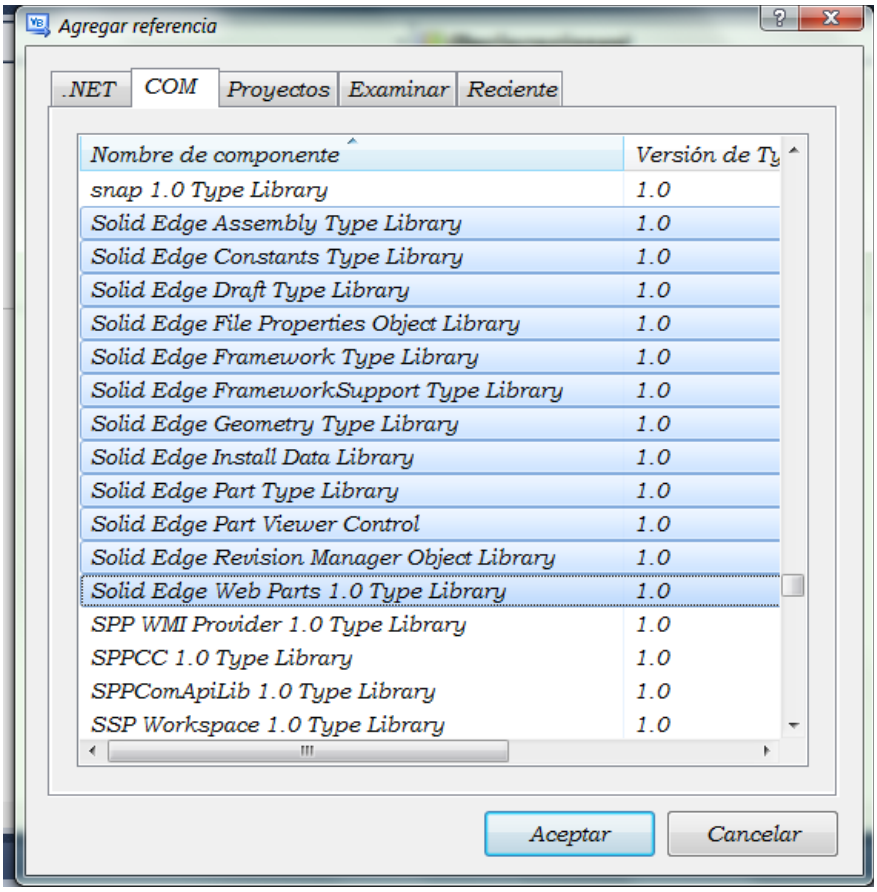

Figura 29: Referencias de Solid Edge.

Dentro de la pestaña "COM" aparecerán las correspondientes librerías de Solid Edge. Será necesario desplazar la barra de movimiento vertical para buscarlas. También es posible ayudarse pulsando las teclas "so" en el teclado que localizarán todas la librerías que comiencen por dichas letras.

El siguiente paso es comenzar a escribir el código. En la plantilla selecciona aparece por defecto parte del código escrito (dependerá de la plantilla seleccionada). Para este ejemplo, y en general para los primeros paso en el aprendizaje de la programación no es necesario modificarlo.

Antes de comenzar a escribir el código del macro es necesario añadir las siguientes líneas de comandos, que permiten a Visual Basic importar y reconocer las constantes usadas por Solid Edge, o por otras aplicaciones.

```
Imports System.Runtime.InteropServices
```

```
Imports SolidEdgeConstants
```

Una vez hecho esto se puede proceder a escribir el código propiamente dicho correspondiente al proceso de automatización.

```
Module Module1
    Sub Main()
  Dim Documento As SolidEdgeFramework.Documents
  Dim Aplicacion As SolidEdgeFramework.Application
        Dim Pieza As SolidEdgePart.PartDocument
        Dim PerfilSets As SolidEdgePart.ProfileSets
        Dim PerfilSet As SolidEdgePart.ProfileSet
        Dim Perfiles As SolidEdgePart.Profiles
        Dim Perfil As SolidEdgePart.Profile
        Dim PlanosReferencia As SolidEdgePart.RefPlane
        Dim Bocetos As SolidEdgePart.Sketchs
        Dim Boceto As SolidEdgePart.Sketch
        Dim Circulos As SolidEdgeFrameworkSupport.Circles2d
        Dim Circulo As SolidEdgeFrameworkSupport.Circle2d
        Dim x As Double
        Dim y As Double
        Dim r As Double
```

El nombre de las variable puede ser elegido sin embargo se debe tener en mente el tipo de objeto que representa, y las características del mismo, dado que éste vendrá impuesto por Solid Edge y las variables deberán ser declaradas conforme aparece en la documentación incluida en Solid Edge.

Es recomendable usar como nombre de la variable el tipo de objeto que representan, lo que facilitará la compresión del código. En este ejemplo se ha elegido el uso del español para facilitar apreciar la relación del código escrito con la acción correspondiente en Solid Edge, con los botones de las herramientas también en español.

Con las variables declaradas puede comenzarse la escritura del código correspondiente a la creación un círculo dentro de un boceto contenido en el plano

seleccionado. Se asigna el valor adecuado a cada variable y se activa el documento:

```
Aplicacion = Marshal.GetActiveObject("SolidEdge.Application")
        Documento = Aplicacion.Documents
        Pieza = Aplicacion.ActiveDocument
        PerfilSets = Pieza.ProfileSets
        PerfilSet = PerfilSets.Add()
        Perfiles = PerfilSet.Profiles
```

El siguiente paso es seleccionar el el plano sobre el que se genera el perfil e iniciar la herramienta Boceto:

```
PlanosReferencia = Pieza.RefPlanes
        Bocetos = Pieza.Sketches
        Boceto = Bocetos.Add()
        Perfil = Perfiles.Add(PlanosReferencia.Item(3))
```

Con el plano seleccionado y la herramienta Boceto activada el siguiente paso es dibujar el círculo de centro (0,0) y radio quince milímetros:

```
Circulos = Perfil.Circles2d
        Circulo = Circulos.AddByCenterRadius(0, 0, 0.015)

    End Sub
    End Module
```

Tras lo cual se añaden las instrucciones “ End Sub” y “End Module” para cerrar el código del macro.

El macro ya estaría preparado para probarlo. Para ello primero es necesario generar la solución a través del menú de herramientas de Visual Basic (Menú/Generar/Generar NombreMacro). Suponiendo que la sintaxis y ortografía del código sea correcta el archivo ejecutable del macro aparecerá en la carpeta que

previamente haya sido seleccionada. Cabe destacar que el hecho de que no existan errores de compilación no implica el correcto funcionamiento del macro, éste debe ser verificado en el propio entorno de Solid Edge.

Si se ejecuta el macro generado se comprueba como efectivamente se genera una circunferencia en el plano adecuado y con las dimensiones preestablecidas como se muestra en la figura:

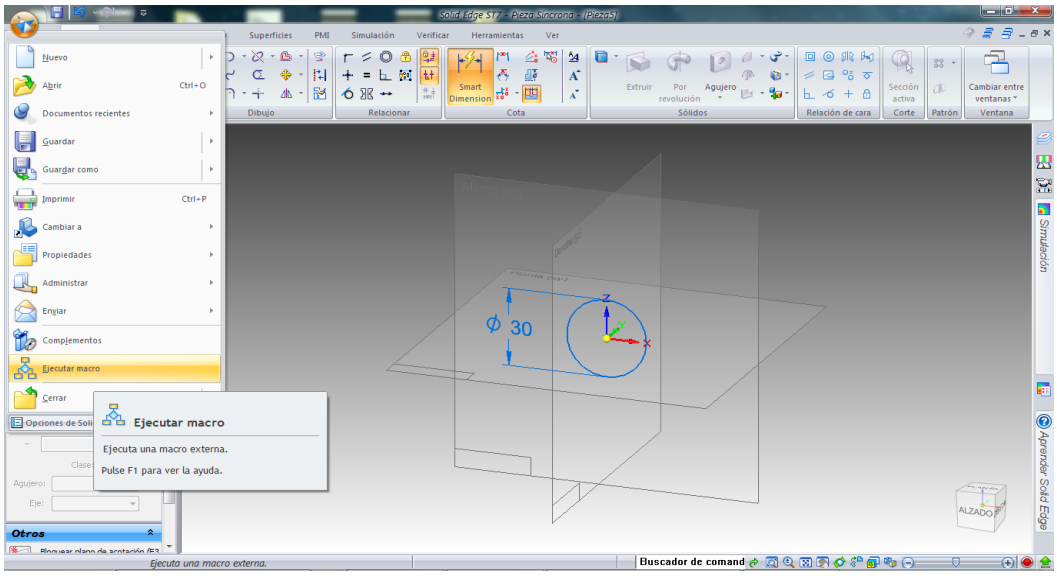

Figura 30: Macro para circunferencia.

Para establecer los paralelismos existentes entre el trabajo a través de la interfaz de Solid Edge y la generación de piezas y/u operaciones mediante programación de automatización de tareas se va proceder a realizar una serie de operaciones sobre este perfil generado, mediante ambos métodos.

El siguiente paso es generar una extrusión recta a partir del perfil circular creado en el paso previo. Para ello, en la interfaz de usuario de Solid Edge, es necesario seleccionar la herramienta Extrusión, el perfil utilizado para la extrusión y el sentido y longitud de la extrusión.

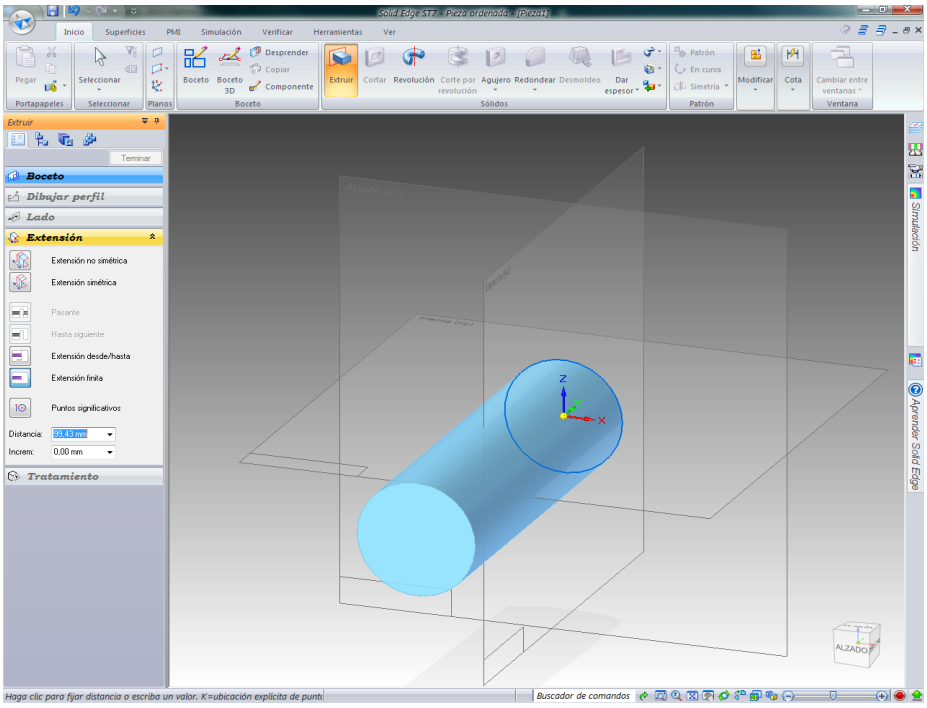

Figura 31: Extrusión recta.

Para ello se ha seleccionado el boceto inicial como base para la extrusión y se le ha asignado la dirección y longitud deseada.

La manera de proceder redactando el código para el macro será análoga, habrá que seleccionar el perfil generado anteriormente , llamar a la herramienta de extrusión e introducir la información necesaria para definir las propiedades de la extrusión.

Se continúa escribiendo el código a partir de las últimas instrucciones de la operación Boceto del apartado anterior, antes de cerrar el mismo mediante las instrucciones " End Sub" y "End Module"

El primer paso, nuevamente, es declarar las nuevas variables utilizadas en esta parte del código. Es recomendable declarar todas las variables utilizadas al comienzo de cada bloque, pero para ilustrar este ejemplo de la manera más sencilla posible se declaran las variables al inicio de cada operación para facilitar la compresión del funcionamiento del macro.

Las variables necesarias para esta operación de Extrusión, así como los parámetros necesarios para definir la misma están disponibles dentro de la guía de

programación de automatizaciones de Solid Edge.

```
Dim Modelos As SolidEdgePart.Models
Dim Modelo As SolidEdgePart.Model
Dim aPerfiles As Array
```

No serán necesarias más variables para la operación de extrusión. "SolidEdgePart.Models" permite, entre otras cosas, el acceso a las diferentes tipos de operaciones mediante las cuales se puede generar un objeto en tres dimensiones y "Array" es una matriz que en este caso permite albergar los elementos que forman parte de la operación de extrusión.

Al generar la extrusión y concluirla la línea que define la circunferencia del boceto desaparece. Para ello es necesario ocultar la visibilidad del perfil generado:

```
Perfil.Visible = False
```

Ésto deshabilitará la visibilidad del perfil sin eliminarlo. El siguiente paso es almacenar la información del perfil generado dentro de la matriz aPerfiles y definir el tipo de modelo que se va a generar con él, en este caso un modelo por extrusión:

```
aPerfiles=Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
aPerfiles.SetValue(Perfil, 0)
Modelos = Pieza.Models

Modelo = Modelos.AddFiniteExtrudedProtrusion( aPerfiles.Length, aPerfiles,_
SolidEdgePart.FeaturePropertyConstants.igRight, 0.1)
```

En la última linea del código (dividida en dos mediante el uno de un guión bajo "_") se genera la operación de extrusión y se fija el valor de los diferentes parámetros que definen sus propiedades. Para ello se declara el valor de "Modelo" como un modelo generado a partir de un extrusión finita. Los términos declarados entre paréntesis son los que definen las características de la extrusión, el perfil seleccionado, el sentido de la extrusión respecto de la base seleccionada

(SolidEdgePart.FeaturePropertyConstants.igRight) y la longitud de extrusión expresada en metros (0.1).

Una vez completado este paso se puede proceder a generar la solución y ejecutar el macro del mismo modo que para paso anterior, el boceto de la circunferencia. Hecho ésto se deberá obtener el siguiente resultado:

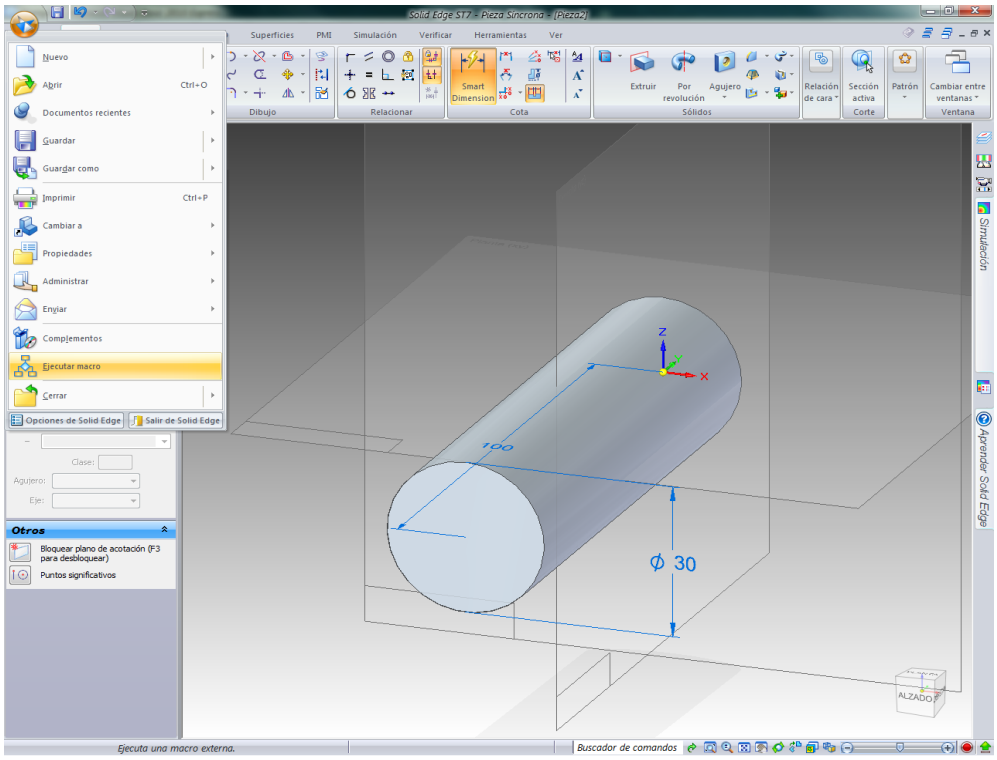

Figura 32: Ejecutar macro.

El código en Visual Basic.Net del macro completo sería pues:

```
Imports System.Runtime.InteropServices
Imports SolidEdgeConstants
Module Module1
        Sub Main()
    Dim Documento As SolidEdgeFramework.Documents
        Dim Aplicacion As SolidEdgeFramework.Application
      Dim Pieza As SolidEdgePart.PartDocument
      Dim PerfilSets As SolidEdgePart.ProfileSets
      Dim PerfilSet As SolidEdgePart.ProfileSet
      Dim Perfiles As SolidEdgePart.Profiles
```

```
        Dim Perfil As SolidEdgePart.Profile
        Dim PlanosReferencia As SolidEdgePart.RefPlane
        Dim Bocetos As SolidEdgePart.Sketchs
        Dim Boceto As SolidEdgePart.Sketch
        Dim Circulos As SolidEdgeFrameworkSupport.Circles2d
        Dim Circulo As SolidEdgeFrameworkSupport.Circle2d
        Dim x As Double
        Dim y As Double
        Dim r As Double
      Dim Modelos As SolidEdgePart.Models
        Dim Modelo As SolidEdgePart.Model
       Dim aPerfiles As Array

          Aplicacion = Marshal.GetActiveObject("SolidEdge.Application")
            Documento = Aplicacion.Documents
            Pieza = Aplicacion.ActiveDocument
            PerfilSets = Pieza.ProfileSets
            PerfilSet = PerfilSets.Add()
            Perfiles = PerfilSet.Profiles
          PlanosReferencia = Pieza.RefPlanes
            Bocetos = Pieza.Sketches
            Boceto = Bocetos.Add()
            Perfil = Perfiles.Add(PlanosReferencia.Item(3))
          Circulos = Perfil.Circles2d
            Circulo = Circulos.AddByCenterRadius(0, 0, 0.015)
          Perfil.Visible = Falsea
            Perfiles=Array.CreateInstance(GetType(SolidEdgePart.Profile),1)
aPerfiles.SetValue(Perfil, 0)
            Modelos = Pieza.Models
            Modelo = Modelos.AddFiniteExtrudedProtrusion(
            _aPerfiles.Length, aPerfiles, _
            SolidEdgePart.FeaturePropertyConstants.igRight, 0.1)

    End Sub
    End Module
```

Con este ejemplo en el que se realizan algunas de las operaciones más habituales en el entorno Solid Edge queda patente cómo la manera de trabaja a la hora de programar automatización de tareas repetitivas en Solid Edge es similar a la manera de trabajar con su interfaz gráfica. Queda patente también la necesidad de conocer el manejo de las distintas herramientas de Solid Edge antes de comenzar a desarrollar macros, dado que a fin de cuentas el macro reproducirá la secuencia de operaciones que el usuario realizaría mediante la interfaz de Solid Edge si éste diseñara la pieza

tradicionalmente.

Siguiendo este método, mediante una secuencia de operaciones, será posible diseñar cualquier pieza del mismo modo en que sería posible diseñarla mediante la interfaz gráfica. Ésto representa una gran ventaja a la hora de diseñar un gran número de piezas, dado que una vez realizado el macro será posible fijarlo en la barra de tareas de Solid Edge y utilizarlo como si fuera una herramienta más, permitiendo al usuario repetir las operaciones y/o piezas propias del macro sin perder tiempo en realizarlas manualmente, lo que supone una gran ventaja en términos de productividad.

Por otra parte la finalidad del macro es la de automatizar el proceso de diseño permitiendo si fuera necesario cierta flexibilidad sobre los parámetros que definen la pieza diseñada, por ejemplo modelar un tornillo, para el que se puede elegir su métrica, longitud, etc.

Mediante la agregación de interfaces gráficas o mediante la consola del sistema es posible permitir al usuario introducir el valor de las variables que definen las operaciones realizadas por el macro, del mismo modo en el que es posible realizarlo mediante la interfaz de Solid Edge. Sin embargo el objetivo de este ejemplo era el de ilustrar las analogías existentes entre el método “tradicional” de trabajo en Solid Edge y la programación de macros y constatar la necesidad de conocer el funcionamiento de Solid Edge antes de adentrarse en la programación de macros.

Queda patente cómo para iniciarse en la programación de macros no se precisan elevados conocimientos de programación, pero poseer ciertas nociones facilitará al programador relacionar el código con los elementos propios de SoliEdge.

6 Macro para el cálculo y diseño de resortes de torsión.

6.1 Introducción.

A continuación se describe el desarrollo y funcionamiento del elemento entorno al cual gira todo el documento. Dicho elemento es una aplicación informática que sirviéndose del entrono de trabajo y herramientas de Solid Edge ST7 calculará y diseñará un resorte de torsión helicoidal con extremos rectos. La aplicación costará de dos partes bien diferenciadas.

La primera parte será una interfaz gráfica mediante la cual el usuario podrá introducir los requisitos que debe cumplir el resorte, además de la introducción de datos la ventana emergida permitirá realizar los correspondientes cálculos de las propiedades del resorte y mostrará los resultados obtenidos.

La segunda parte de la aplicación será la encargada de interactuar con las herramientas de Solid Edge y diseñar de manera automática en el entorno Pieza de Solid Edge el resorte obtenido numéricamente en la ventana de la aplicación.

6.2 Interfaz del Macro.

A la hora de iniciar un nuevo proyecto Visual Basic ofrece varias plantillas preconfiguradas a fin de facilitar el trabajo. Para los primeros pasos a la hora de programar macros la plantilla “Aplicación de Consola” es una buena alternativa, pues permite al programador centrarse en los aspectos estrictamente relativos a la automatización de las operaciones que permite Solid Edge.

En el caso de desear crear alguna automatización con parámetros configurables por parte del usuario final del macro una aplicación de consola pude no ser la más indicada, pues puede llegar a complicar la interactuación del usuario con la aplicación.

Por este motivo, para facilitar el manejo de la herramienta y la muestra de los resultados numéricos asociados a las variables del resorte, se ha preferido desarrollar una aplicación a partir de una plantilla “Windows Form” que mejorará y facilitará la experiencia del usuario con el macro, permitiendo el manejo de la herramienta a través de botones y la introducción de los datos a través de casillas, en lugar de tener que introducir toda la información a través de la consola del sistema.

6.2.1 Creación de la ventana del programa.

Este apartado se centra en el aspecto estético de la aplicación y la distribución de los elementos que conforman la interfaz del programa.

Una vez abierta la plantilla “Windows Form” aparecerá en el entorno de trabajo la ventana por defecto sobre la que se comenzará a trabajar.

En ella se podrán a través del catálogo de herramientas disponibles en el lateral izquierdo del entorno de trabajo (Cuadro de Herramientas) será posible agregar los distintos elementos que mostrarán la información relativa al cálculo del resorte y permitirán al usuario introducir los requisitos que debe de cumplir el resorte. Una vez introducida la información el usuario podrá calcular, reiniciar, diseñar (dentro de Solid Edge) y cerrar la ventana.

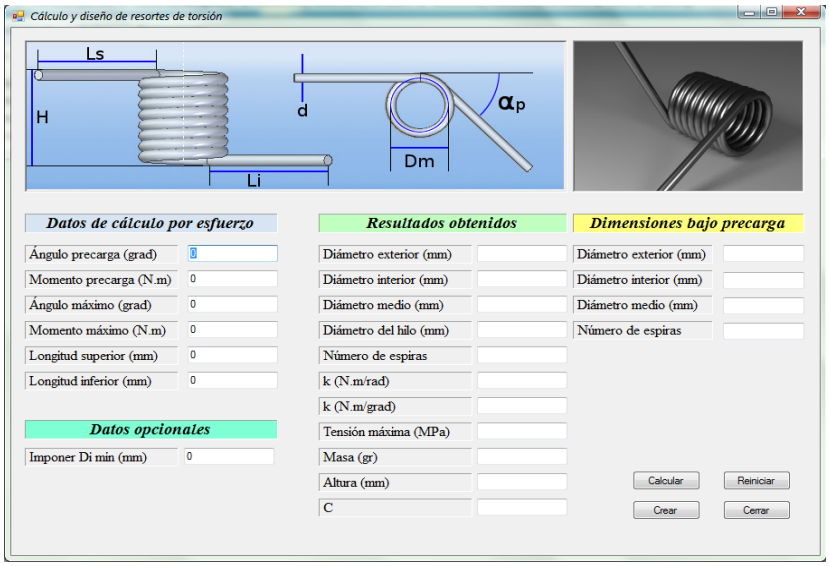

Figura 33: Interfaz del macro de cálculo de resortes.

La figura 33 muestra el diseño final de la ventana del macro. Las dimensiones de la ventana, 900 x 650 píxeles, tamaño que permite mostrar toda la información sin necesidad de añadir barras de desplazamiento horizontal o vertical. El tamaño de la ventana no debería suponer ningún tipo de problema dado que los requisitos mínimos de Solid Edge hacen necesaria pantallas con una resolución mínima de 1280x1024 pixeles, por tanto la ventana del macro permite mostrar toda la información disponible en cualquier ordenador capaz de ejecutar Solid Edge.

En ella se muestran bajo el rótulo “Datos de cálculo” los datos que el usuario debe de introducir para el cálculo del resorte :

- Ángulo inicial de trabajo. Este no se corresponde (salvo que la precarga sea nula) al incremento de ángulo respecto de la posición del extremo del resorte sin carga, sino que se toma como referencia el eje horizontal mostrado en la imagen. Esto es debido a que el número de vueltas (que impondrá los ángulos entre los extremos del resorte) es desconocido hasta que el muelle es calculado. Expresado en grados.

- Momento de precarga o inicial. Es el momento aplicado al resorte en la posición inicial del recorrido. Expresado en [N.m].

- Ángulo máximo. Es ángulo máximo alcanzado durante el recorrido del resorte

respecto de la horizontal. Al igual que el ángulo inicial en ángulo respecto de la posición inicial del resorte sin cargas aplicadas se podrá determina una vez calculado el resorte. Expresado en grados.

- Momento máximo. Es el momento alcanzado al final del recorrido del resorte. Será el parámetro utilizado para el cálculo de la tensión máxima sufrida por el resorte. Expresado en [N.m].

- Longitudes superior e inferior. La longitud de los extremos rectos del resorte a lo largo del eje horizontal de referencia de la imagen respecto del centro de la hélice del resorte. Expresados en milímetros.

- Diámetro interior mínimo: permite al usuario imponer una restricción con respecto al diámetro interior mínimo de la hélice. Este parámetro es opcional.

En el rótulo central "Resultados obtenidos" se mostrarán los resultados obtenidos:

- Diámetro exterior. Es el diámetro exterior de la hélice del resorte, resultado de sumarle al diámetro medio de la hélice el diámetro del hilo normalizado. Expresado en milímetros.

- Diámetro interior. Es el diámetro interior de la hélice del resorte, resultado de restarle al diámetro medio de la hélice el diámetro del hilo normalizado. Expresado en milímetros.

- Diámetro medio: es diámetro de la hélice generada por la línea imaginaria trazada por el centro de la sección circular del hilo. Expresada en milímetros.

- Diámetro del hilo o alambre. El diámetro del hilo normalizado, elegido entre los diámetros disponibles dentro del catálogo. Los cálculos y por tanto los resultados obtenidos se realizan a partir del diámetro ya normalizado. Expresado en milímetros.

- Constante del resorte “k”. La relación entre el momento aplicado y el incremento de ángulo asociado al mismo. Expresada en [N.m/rad] y [N.m/grados].

- Tensión máxima. La tensión máxima sufrida por el resorte, calculada a partir del diámetro del hilo normalizado. Expresada en [MPa].

- Altura. Altura del resorte a lo largo de su eje. Expresada en milímetros.

- C o índice de curvatura. Relación entre el diámetro medio y el diámetro del hilo. Adimensional.

El rótulo derecho “Dimensiones bajo carga” muestra las dimensiones del resorte bajo las condiciones de precarga. Serán los valores que se usarán para diseñar el resorte en Solid Edge:

- Diámetro exterior. Expresado en milímetros.
- Diámetro interior. Expresado en milímetros.
- Diámetro medio. Expresado en milímetros.
- Número de espiras.

La ventana del macro permite al usuario diseñar un resorte que proporciona el momento deseado dentro del rango de ángulos de trabajo pare el que el muelle será diseñado.

Por tanto el usuario podrá calcular el resorte a partir del ángulo de inicio de trabajo o de precarga, el par de inicio de trabajo o de precarga, el momento máximo y su ángulo asociado. Como resultado el programa proporciona las dimensiones y características mecánicas del resorte capaz de trabajar con las condiciones impuestas y garantizar la vida infinita del resorte, así como las dimensiones asociadas a su estado de precarga, a partir de las cuales se diseñará el resorte en Solid Edge.

El motivo de ofrecer unas posibilidades de configuración tan restringidas tiene

como finalidad evitar sobrepasar la propiedades mecánicas del material usado. Debido a que los ángulos y los momentos se consideran impuestos por el entorno en el que muelle vaya a trabajar las variables con las que se puede diseñar el muelle para un material dado será el diámetro del hilo, el diámetro de la espiral y el número de espiras o vueltas.

El número de vueltas estará condicionado pos los ángulos de trabajo y los momentos inicia y final del recorrido (en el caso de un momento inicial nulo el número de vueltas deberá ser igual a un número entero más el ángulo inicia), a diferencia de los muelles de compresión o tracción donde se podrá modificar el número de vueltas con mayor libertad.

El diámetro de hilo vendrá limitado por dos factores (para un mismo material), el diámetro mínimo vendrá condicionado por el momento máximo, dado que deberá garantizar que no se sobrepasa la tensión máxima admitida. También estará condicionado por los valores normalizados de diámetro de alambre disponible y por tanto habrá que seleccionar uno dentro de los disponibles en el catálogo, inmediatamente superior al obtenido mediante la utilización de las ecuaciones pertinentes (inmediatamente superior).

El hecho de que el número de vueltas tenga que cumplir unas condiciones determinadas y que el diámetro del hilo también hace complicado que el usuario sin conocimientos técnicos pueda obtener un resorte funcional si se le permite elegir una o varias de las dimensiones del resorte. En el caso de ofrecer la posibilidad de jugar con las dimensiones del resorte muy posiblemente el usuario se vería en la necesidad de hacer uso de las correspondientes ecuaciones aplicadas al cálculo de resortes de torsión y la herramienta perdería gran parte de su sentido.

En el caso de imponer un diámetro mínimo el programa calculara un diámetro interior igual o inmediatamente superior compatible con los ángulos de trabajo impuestos.

Estas restricciones permiten obtener resultados siempre dentro del rango de la seguridad frente a vida infinita del elemento.

6.2.2 Funcionamiento de la interfaz.

La ventana de la aplicación está formada por una serie de elementos que permiten mostrar información útil al usuario, introducir los parámetros de cálculo, obtener los resultados y activar la secuencia de operaciones de Solid Edge que dará como resultado el diseño del resorte con las dimensiones y geometría adecuadas.

En este caso la ventana está formada por dos "Picturebox" (elemento que contiene las imágenes) que permiten mostrar imágenes, veintiséis "Label" que contienen el nombre (el texto escrito en la ventana de la aplicación) de las variables y el texto de los rótulos, veintiún "Textbox" que permiten introducir y mostrar los valores de las variables y cuatro botones que permiten desencadenar las acciones asociadas a cada uno de ellos.

6.2.2.1 Funcionamiento del botón "Calcular".

Cuando el usuario presione este botón desencadenará el proceso de cálculo del resorte a partir de los datos introducidos. Alberga todo el código encargado de calcular las dimensiones y características mecánicas del resorte.

El primer paso es determinar la constante del resorte a través de los momentos introducidos y los ángulos inicial y final:

$$k = \frac{M_f - M_i}{a_f - a_i}$$

El código asociado a esta operación es:

```
k_grad = (Val(TextBox4.Text) - Val(TextBox2.Text)) / (Val(TextBox3.Text) - Val(TextBox1.Text))
k_rad = k_grad * ((Math.PI) / 180)
```

Donde "TextBox4.text", "TextBox2.text", "TextBox3.text", "TextBox1.text" se

corresponden con los valores de momento máximo, momento de precarga, ángulo final y ángulo inicial respectivamente. Se calcula “k” en Newton por metro partido de radianes y de grados.

El siguiente paso implica el cálculo del diámetro del hilo, el límite de fatiga y la tensión máxima fluctuante soportada. El diámetro y las tensiones dependen el uno de las otras y viceversa, es decir, para calcular el diámetro hace falta conocer el límite de fatiga y para conocer el límite de fatiga es preciso es preciso conocer el diámetro (el límite de fatiga depende entre otras cosas del factor de tamaño K_b calculado a través del diámetro).

El límite de fatiga para cargas fluctuantes venía dado por:

$$S_f = \frac{S_e}{1 + \frac{(1+x)\cdot S_e}{2\cdot S_{ut}}}$$

Para resolver este problema es necesario realizar un proceso iterativo, en el que a partir de un diámetro de hilo calculado para un supuesto valor de K_b (se partirá de un valor de K_b=1) se determinará el límite de fatiga corregido $S_{e.}$, con el valor obtenido de se calculará el diámetro del hilo.

A partir del nuevo diámetro del hilo se calcula K_b y se repite este proceso tantas veces como sea necesario hasta que la diferencia de valores entre iteraciones consecutivas sea suficientemente buena.

El el caso de hilos normalizados el proceso iterativo se simplifica, habrá que realizar iteraciones hasta obtener dos veces seguidas el mismo valor del diámetro normalizado o sea igual diámetro normalizado calculado 2 iteraciones previas (puede darse el caso de que el diámetro d_{n+3} implique el diámetro d_{n+2} y que este a su vez implique el diámetro d_{n+3} con cual el programa se quedaría atrapado en un bucle sin fin, la condición “Or” evitará este problema).

El código asociado a este proceso es el siguiente:

```
While c4 = 0
        Su = 1400000000
        Se = 700000000
        Sec = Se * Kb
        xa = Val(TextBox2.Text) / Val(TextBox4.Text)
        Sf = Sec / (1 + ((1 + xa) * (Sec / (2 * Su))))
        M = Val(TextBox4.Text)
        da = ((32 * M) / ((Math.PI) * Sf)) ^ (1 / 3)
        c1 = 0
        c2 = 0
        c3 = 0
        While c3 = 0
          If da < DiametroNormalizado(46) Then
            While c2 = 0

              If (da > DiametroNormalizado(c1)) And (da < DiametroNormalizado(c1 + 1)) Then
                c2 = 1
                c3 = 1
                d(c5) = DiametroNormalizado(c1 + 1)
                TextBox14.BackColor = Color.White
                TextBox4.BackColor = Color.White
              Else
                c1 = c1 + 1
              End If
            End While
            Kb = ((d(c5) * 1000) / 7.62) ^ (-0.1133)
          Else
            MessageBox.Show("Momento demasiado grande, reintroduzca los datos")
            c3 = 1
            TextBox14.BackColor = Color.OrangeRed
            TextBox4.BackColor = Color.OrangeRed
          End If
        End While
        If d(c5) = d(c5 - 1)  Or d(c5) = d(c5 –2) Then
          c4 = 1
        Else
          c5 = c5 + 1
        End If
      End While
```

La primera operación que realiza este código es el cálculo de la tensión S_f fluctuante a partir del límite de fatiga corregido S_e a falta de aplicar el factor corrector del tamaño K_b.

Con este valor de S_f se calcula por primera vez el diámetro de alambre necesario para no superar dicha tensión.

Tras calcular el diámetro del hilo se realiza un bucle que busca el diámetro normalizado inmediatamente superior al calculado, acto seguido se determina K_b con este diámetro.

A partir de K_b se calcula el límite de fatiga corregido y se reinicia el proceso, es decir se vuelve a calcular S_f, con este valor se calcula el diámetro del hilo, se busca el normalizado y se calcula nuevamente K_b y S_e.

Al final de cada bucle completo se compara el diámetro obtenido en el bucle realizado con el diámetro del ciclo anterior.

Este proceso se repite hasta que los diámetros normalizados de dos iteraciones consecutiva coinciden. Una vez ocurre ésto la tensión máxima y el diámetro del hilo ya quedan fijadas.

El siguiente paso es determinar el diámetro de la hélice y el número de espiras. El código asociado a dicho proceso es el siguiente:

```
Na = 5
 N = Na + (ai / 360)
 Dm = (ND / N)
 C = (Dm / df)
 If C > 5 Then
    While C > 5
       Na = Na + 1
       N = Na + (ai / 360)
       Dm = (ND / N)
       C = (Dm / df)
    End While
 End If
 If C < 3 Then
    While C < 3
       Na = Na - 1
```

```
        N = Na + (ai / 360)
        Dm = (ND / N)
        C = (Dm / df)
    End While
End If
```

Se parte de la base que el sistema de referencia elegido para que el usuario introduzca los datos tiene los extremos de la hélice alineados, es decir, se parte de una hélice de un número de vueltas entero, al que posteriormente se le añadirá el ángulo necesario para adecuarse a la solución del problema.

En primera instancia se calcula el parámetro "DN" que se corresponde con el producto del diámetro de la hélice por el número de espiras.

Una vez calculado "DN" es necesario iniciar otro proceso iterativo y fijar un rango de C o indice de curvatura deseado. El número de vueltas debe cumplir dos condiciones, la primera ser un número entero más una parte decimal (es decir separamos la parte entera de la decimal) y la segunda condición es que multiplicado por D debe ser igual a "DN"

Se parte de un valor de N igual a cinco al que se le suma el angulo inicial del resorte (obtenido a través de la ecuación de la recta Momento frente a ángulo) y se realiza un bucle que suma o resta una unidad por iteración a N hasta obtener un valor de C comprendido en el rango fijado, es decir, entre 5 y 3 (este valor se puede editar para obtener un diámetro determinado si fuera necesario).

También es posible imponer un diámetro interior mínimo. Dicha acción es realizada por el siguiente bucle:

```
'Determinar D y N para un diámetro interior mínimo
    Na = 100
    While (Dm - df) < (Val(TextBox22.Text) / 1000)
        Na = Na - 1
        N = Na + (ai / 360)
        Dm = (ND / N)
        C = (Dm / df)
```

```
End While
```

Este bucle toma el valor de la casilla 22 y se ejecuta hasta que el valor obtenido del diámetro interior es el primero capaz de satisfacer la limitación impuesta teniendo en cuenta las condiciones que debe de cumplir el número de espiras N.

Una vez completado este bucle ya se conocen todas las dimensiones del resorte, sólo queda calcular las dimensiones del resorte bajo precarga y mostrar los resultados en pantalla. El código para el resto de cálculos es:

```
'Determinar longitud total de los extremos
    LongS = Val(TextBox5.Text) / 1000
    LongI = Val(TextBox6.Text) / 1000
    L = LongS + LongI

    'Determinar masa
    M = 7.8 * ((Math.PI) * (df ^ 2)) * 0.25 * ((Math.PI) * Dm * N + L)
    TextBox15.Text = Val(M)
    'Mostrar Altura y tensión máxima
    TextBox14.Text = Val((Sf * ((da / df) ^ 3)) / 1000000)
    TextBox16.Text = N * df * 1000
    'Asignar dimensiones en precarga
    Nc = N + ((Val(TextBox2.Text) / k_grad) / 360)
    Dmc = Dm * (N / Nc)
    Dic = Dm - df
    Dec = Dm + df
    TextBox18.Text = Val(Dec)
    TextBox19.Text = Val(Dic)
    TextBox20.Text = Val(Dmc)
    TextBox21.Text = Val(Nc)
```

Al igualar los “Textboxt.Text” a “Val (variable)” se introducen los valores numéricos en las casillas correspodientes.

6.2.2.2 Funcionamiento de los botones “Reiniciar”, “Crear” y “Cerrar”.

En el bloque asociado al botón “Calcular” está incluido todo el código relativo a los cálculos del resorte y por tanto se hacía necesario dedicarle un apartado en exclusiva, sin embargo el código asociado al resto de los botones no reviste de

demasiada complejidad, por tanto se describe su comportamiento en un mismo apartado.

- Botón “Reiniciar”. Este botón restablecerá el valor de todas las casillas “Textbox” asignándolas el valor igual a cero. El código asociado a dicha acción es el siguiente:

```
Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click

Textbox1.text = 0
Textbox2.text = 0
…...
Textbox21.text= 0
End Sub
```

- Botón “Crear”. Este botón desencadenará la ejecución de la secuencia de operaciones dentro de Solid Edge. Será necesario que la operación “Calcular” se haya realizado previamente. Una vez presionado, Solid Edge modelará el resorte y la ventana del macro se cerrará.

```
'Acción del botón Crear
  Private Sub Button4_Click(sender As System.Object, e As System.EventArgs) Handles Button4.Click
    End
  End Sub
```

El bloque del código asociado a este botón no contiene código de las instrucciones de las herramientas de Solid Edge. El bloque llamará al bloque que contiene dichas instrucciones. El código de las herramientas y operaciones específicas de Solid Edge será descrito más adelante.

- Botón “Cerrar.Este botón cierra la ventana y el macro a través de la instrucción “End”.

6.3 Automatización del diseño del resorte en Solid Edge.

6.3.1 Introdución.

Este apartado se centra en describir el código que produce la secuencia de operaciones en Solid Edge que da como resultado el diseño de un resorte que se ajusta a los valores y dimensiones obtenidas mediante la interfaz gráfica del resorte.

A partir de los resultados obtenidos en el apartado anterior, de los cuales se tomarán las variables que definirán los parámetros de las operaciones de Solid Edge, se procederá a redactar el código relativo a las correspondientes operaciones.

El primer paso será determinar la secuencia de operaciones necesarias en Solid Edge para generar el objeto deseado, en este caso se necesita generar una operación que dé como resultado un prisma de base circular (un de los extremos) . El siguiente paso será generar una hélice que parta de una de las bases del primas circular generado en la operación inicial y por último será necesario añadir otro primas circular que parta del extremo libre de la hélice generada, perpendicular a su sección.

Por tanto la secuencia de operaciones, que se ilustran en las figuras 34, 35 ,36 y 37 sería la siguiente:

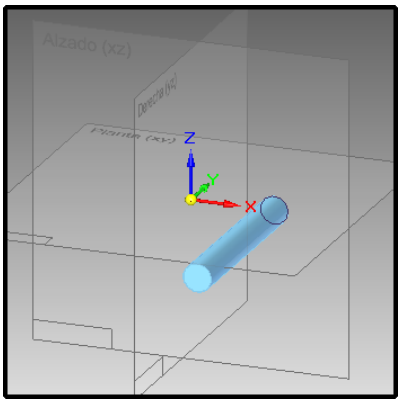

Figura 34: Extrusión recta.

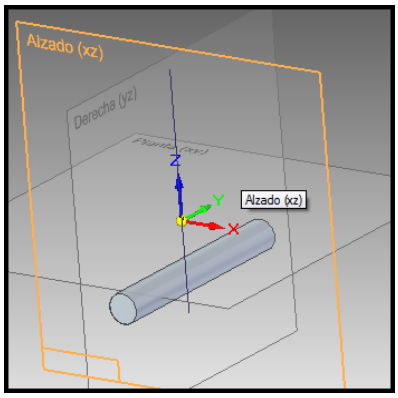

Figura 35: Eje de revolución.

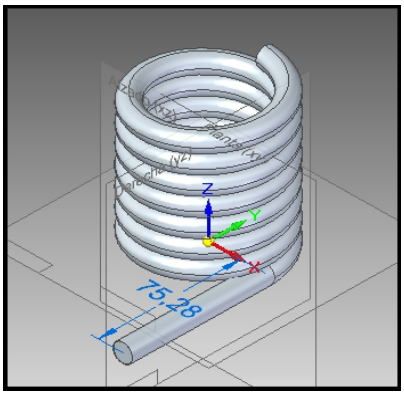

Figura 36: Extrusión helicoidal.

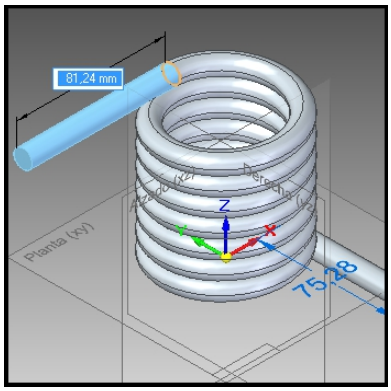

Figura 37: Extrusión superior.

En la figura 34 se muestra como se genera una extrusión recta a partir de un perfil circular previamente dibujado. Será necesario definir el plano de trabajo, la posición del centro de la circunferencia y la dirección y sentido de la extrusión, así como su longitud.

En la figura 35 se muestra una línea dibujada que pasa por el origen de referencia. Esta línea hará la veces de eje de revolución.

En la figura 36 se muestra la hélice generada a partir de un perfil circular coincidente con la base de la extrusión recta inicial y en la que ha utilizado la línea creada en la figura 35 a modo de eje. En esta operación habrá que definir el objeto utilizado como eje, el plano sobre el que se dibuja la sección, la posición y dimensiones del perfil circular, el paso, el número de vueltas y el sentido de giro de la hélice.

En la figura 37 se genera la última extrusión, de base coincidente con el final de la hélice y con la longitud deseada. Nuevamente será necesario definir el plano de trabajo, posición y dimensiones de la sección, sentido de la extrusión y longitud de la misma.

Una vez conocidos los pasos necesarios se puede proceder a automatizar esta secuencia de operaciones de tal manera que el diseño obtenido se corresponda con la solución numérica obtenida en la ventana del macro.

6.3.2 Extrusión recta. Extremo inferior del resorte.

Esta operación ha sido descrita anteriormente como parte de un ejemplo de creación de macros para Solid Edge en el apartado 5.2.4, por tanto se describe sólo las peculiaridades propias de la finalidad del diseño. El código completo del macro puede ser consultado el capítulo 9, Anexo.

A diferencia del ejemplo mostrado anteriormente el perfil generado no puede estar en el centro de los planos de referencia, si no que la posición del centro de la circunferencia debe de coincidir con el radio medio del resorte calculado. Por tanto la posición de la circunferencia de ser ($D'_m/2$, 0) dado que para modelar el resorte se toma la posición inicial de trabajo.

El diámetro de la sección circular debe ser igual al diámetro normaliza del hilo obtenido durante los cálculos y la longitud de la extrusión deberá coincidir con la longitud del extremo inferior, introducida por el usuario. El código relativo a la operación de dibujo del boceto con el perfil circular y la extrusión recta es el siguiente:

```
' Crear boceto
sketches = objPart.Sketches.
sketch = sketches.Add()
objProfile = objProfiles.Add(objRefplanes.Item(3))
objcircles2d = objProfile.Circles2d
objcircle2d = objcircles2d.AddByCenterRadius(x, 0, r)
' Cerrar perfil
objProfile.End( _
SolidEdgePart.ProfileValidationType.igProfileClosed)
' Ocultar perfil
objProfile.Visible = False
' Crear matriz para albergar la información del perfil
aProfiles = Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
aProfiles.SetValue(objProfile, 0)
objModels = objPart.Models
' Generar la extrusión recta.
```

```
objModel = objModels.AddFiniteExtrudedProtrusion( _
aProfiles.Length, _
aProfiles, _
SolidEdgePart.FeaturePropertyConstants.igRight, _
LongI)
```

La diferencia más significativa respecto del ejemplo es el hecho de que las variables que definen la circunferencia y la longitud de la extrusión no están fijadas, lo cual permitirá variar dichas dimensiones en función de los cálculos realizados y los resultados obtenidos.

La circunferencia viene definida por "objcircle2d = objcircles2d.AddByCenterRadius(x, 0, r) " donde "x" será el radio de la hélice y "r" el radio del hilo del resorte y la longitud "LongI" de la extrusión se define en el último termino de la instrucción de la extrusión.

6.3.3 Extrusión recta. Extremo superior.

La base de la extrusión del extremo superior debe coincidir con el final de la hélice generada, para lo que será necesario crear un plano coincidente con la cara del extremo final de la hélice. Para ello se tomará como referencia el número de vueltas o espiras que conforman la hélice. Para generar dicho plano también se hará uso de un eje, que a su vez servirá como referencia para la extrusión helicoidal lo cual permitirá ahorrar un paso.

El plano de referencia deberá pasar por el eje de la hélice y formar el mismo ángulo con respecto el plano de referencia del que parte extrusión del extremo inferior que la cara final de la hélice. De esta manera la sección del extremos superior de la hélice estará contenida en plano auxiliar y se garantiza la continuidad del hilo a lo largo de toda pieza.

El código para generar el plano es el siguiente:

```
objRPlanes = objPart.RefPlanes
objPPlane(1) = objRefplanes.Item(3)
objRPParallel(1) = objRPlanes.AddAngularByAngle(ParentPlane:=objPPlane(1), _
                                    Angle:=((2) * (Nc) * (Math.PI)),_
NormalSide:=SolidEdgePart.ReferenceElementConstants.igPlaneRotateLeft, Pivot:=objEje, _
PivotOrigin:=SolidEdgePart.ReferenceElementConstants.igPlaneAlignY, Local:=False)
```

El primer paso ha sido elegir un plano de referencia (objRPPlanet (1) = objRefplanes.Item(3)) que se corresponde con el plano XZ, sobre ese plano se ha generado un plano con el ángulo deseado , que se corresponde con Nc. El eje utilizado es el mismo que se usará más tarde en la extrusión helicoidal (figura35).

Una vez creado el plano el proceso a realizar será similar al de la extrusión recta inferior, las dos únicas diferencias vendrán de la necesidad de usar el plano auxiliar generado y de ubicar el boceto circular que sirve de base en la posición adecuada, es decir, su centro debe estar a una altura igual al paso por el número de vueltas y su distancia respecto del eje ha de ser el radio medio de la hélice.

Para utilizar el nuevo plano creado para la extrusión el código es el siguiente:

```
objProfil = objProfiles.Add(objRPParallel(1))
```

El resto del código es similar al mostrado en el apartado relativo al la extrusión inferior.

6.3.4 Extrusión helicoidal. Cuerpo del resorte.

La siguiente operación será la que genere la hélice. La hélice deberá comenzar en la base de la extrusión anterior, por tanto el plano de referencia empleado será el mismo que le usado en la operación anterior. Igualmente la circunferencia deberá ser la misma, así como su posición.

Los pasos a seguir son los mismos que para el caso de la extrusión recta, con la salvedad de que será necesario, además de generar la circunferencia, generar el eje que servirá de referencia para la extrusión helicoidal. El eje tendrá que ser identificado como

tal, no sirviendo de referencia una simple línea. En este caso aprovechando el eje creado para el plano de referencia oblicuo de la operación de la extrusión del extremo superior no será necesario crear un nuevo eje, ya que en este caso son coincidentes.

```
' Generación del boceto con la circunferencia y el eje de revolución
bocetos = objPart.Sketches
boceto = bocetos.Add()
objProfile = objProfiles.Add(objPlanoRef.Item(3))
'Crear circunferencia
objcirculos2d = objProfile.Circles2d
objcirculo2d = objcirculos2d.AddByCenterRadius(x, 0, r)
' Cerrar el perfil
objProfile.End(SolidEdgePart.ProfileValidationType.igProfileClosed)
' Deshabilitar la visivilidad del perfil generado
objProfile.Visible = False
' Crear matriz para almacenar la información del perfil creado
aProfiles = Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
aProfiles.SetValue(objProfile, 0)
' Obtener referencia para la colección de modelos
objModels = objPart.Models
' Generación de la extrusión helicoidal
objModel = objModels.AddFiniteBaseHelix(HelixAxis:=objEje, _
AxisStart:=SolidEdgePart.FeaturePropertyConstants.igStart, _
NumCrossSections:=1, CrossSectionArray:=aProfiles, _
ProfileSide:=SolidEdgePart.FeaturePropertyConstants.igRight, _
 Height:=0, Pitch:=d + 0.0000001, _
 NumberOfTurns:=Nc, _
 HelixDir:=SolidEdgePart.FeaturePropertyConstants.igRight)
```

En la operación de extrusión helicoidal se define el eje usado de referencia, su orientación, el número de secciones usadas, el perfil circular usado, la altura (fijada en Height=0 no es tomada como referencia), el paso, el número de vueltas, Nc, y el sentido de giro.

“Nc” es la variable que asigna el número de vueltas con el resorte sometido a la precarga establecida por el usuario. De igual manera que en el entorno de Solid Edge es posible definir una extrusión helicoidal a partir de la altura y número de vueltas o número de vueltas y paso, también será posible a través del código de la instrucción. En este caso se ha elegido el número de vueltas y el paso por ser la opción más sencilla y directa.

En el caso del paso o “Pitch” el valor asignado coincide con el diámetro normalizado del hilo, debido a que en general y para la mayoría de aplicaciones las espiraras de este tipo de resortes suelen ser compactas el macro se centrará en generar resortes con espiras compactas

. Al diámetro del hilo se la ha añadido una pequeña cantidad. Esto es debido a que en el caso de usar como paso justamente el diámetro del hilo se solapan los perfiles de las secciones de las espiras consecutivas, como se muestra en la imagen:

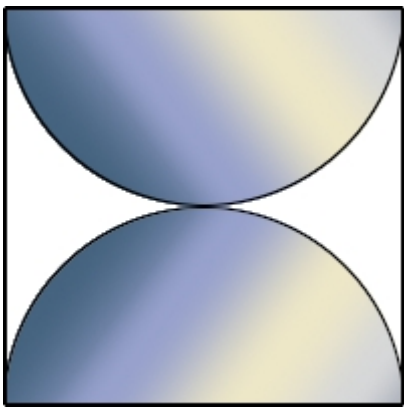

Figura 38: Secciones solapadas.

Esta figura representa la sección circular de dos espiras consecutivas, como se aprecia en la imagen si la distancia entre los centros (el paso) es exactamente igual a su diámetro los perímetros de las circunferencias se solapan en un punto (o una línea si se considera toda la hélice), lo que a efectos prácticos equivaldría a una soldadura o unión entre las espiras del muelle.

Esto trasladado a Solid Edge da como resultado lo mostrado en las siguientes figuras:

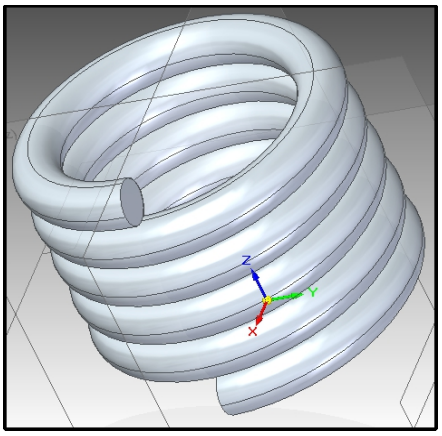

Figura 39: Error en extrusión.

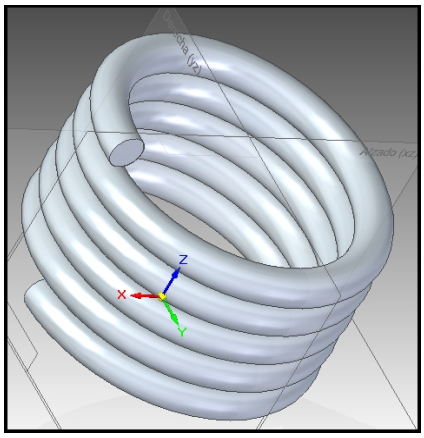

Figura 40: Extrusión sin error.

En estas dos imágenes se puede comparar la diferencia de resultado. En la imagen de la izquierda (paso igual al diámetro) aparecen cuatro divisiones a lo largo de toda hélice resultado del solapamiento de los perfiles de las espiras, en la imagen de la derecha se le ha sumado al paso una centésima de milímetro y se evita la división del la superficie de la hélice en cuarto áreas. Esta hecho sucede tanto trabajando con las herramientas de Solid Edge como realizando la extrusión a través de un macro.

En el caso de seguir disminuyendo el paso llegará un momento en el que se produzca un error y la operación no pueda continuar. Aparecerá entonces un mensaje de error en el PathFinder [6].

Para el caso del macro al diámetro del hilo se ha añadido una diezmilésima parte de un milímetro (por defecto Solid Edge muestra las unidades en milímetros, pero a la hora programar el macro las unidades por defecto serán metros). Dicha cantidad es la mínima que evita este problema y no interferirá a la hora de acotar en Solid Edge a excepción de muelles con mas de cien espiras, pues por defecto sólo trabaja con centésimas de milímetro.

En cualquier caso todos los cálculos realizados serán hechos con el diámetro normalizado del hilo, por tanto la cantidad añadida para evitar este problema no repercute en ningún momento en los resultados obtenidos.

6.3.5 Ejemplos del funcionamiento del macro.

Con las operaciones de extrusión automatizadas y ligadas a los cálculos de la interfaz del programa ya es posible comprobar el correcto funcionamiento del macro para el cálculo de resortes helicoidales de torsión.

El primer paso será abrir Solid Edge y ejecutar el archivo "exe" creado al generar la solución del macro en Visual Basic.

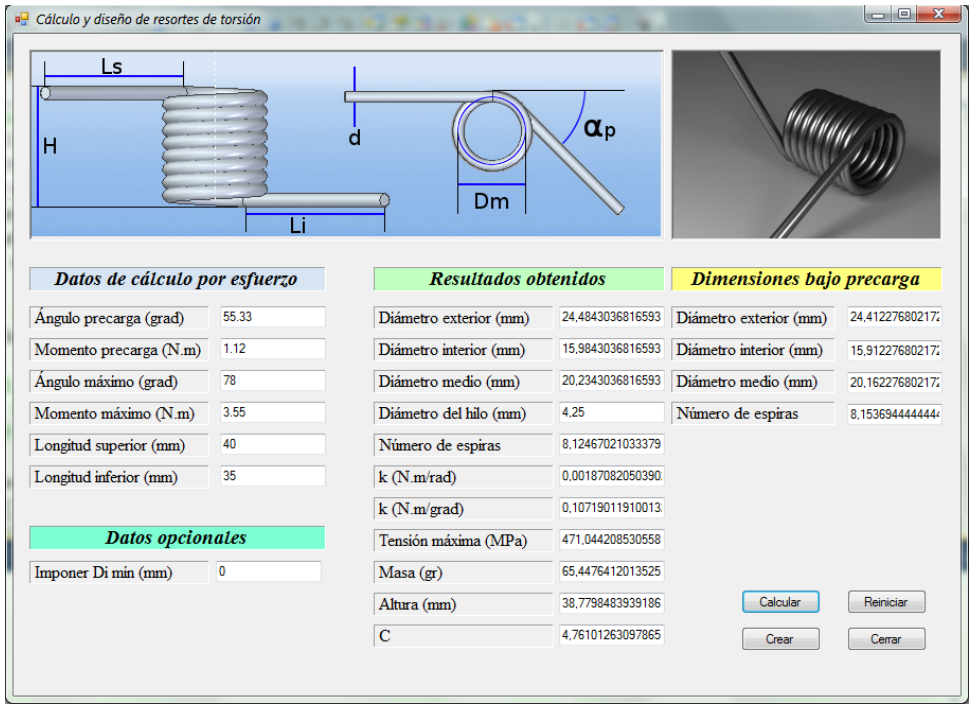

Figura 41: Interfaz-ejemplo1

El ejemplo se realiza para un acero ASTM A229, estirado en frio adecuado para resortes de uso general [8].

Su resistencia a rotura es de 2020 MPa y su límite a fatiga sin corregir S'_e es de 700 MPa.

Se selecciona "Crear" y se ejecutan las instrucciones relativas a Solid Edge, dando como resultado la siguiente imagen:

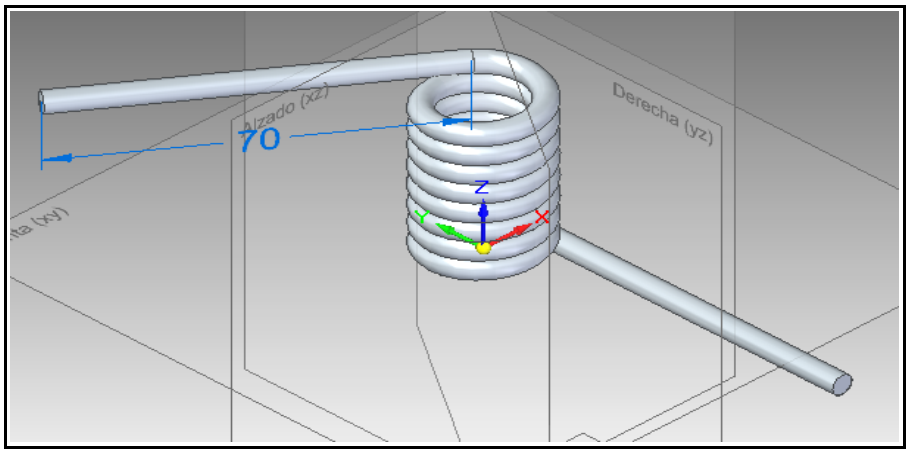

Figura 42: Resultado del Macro.

Y sus cotas:

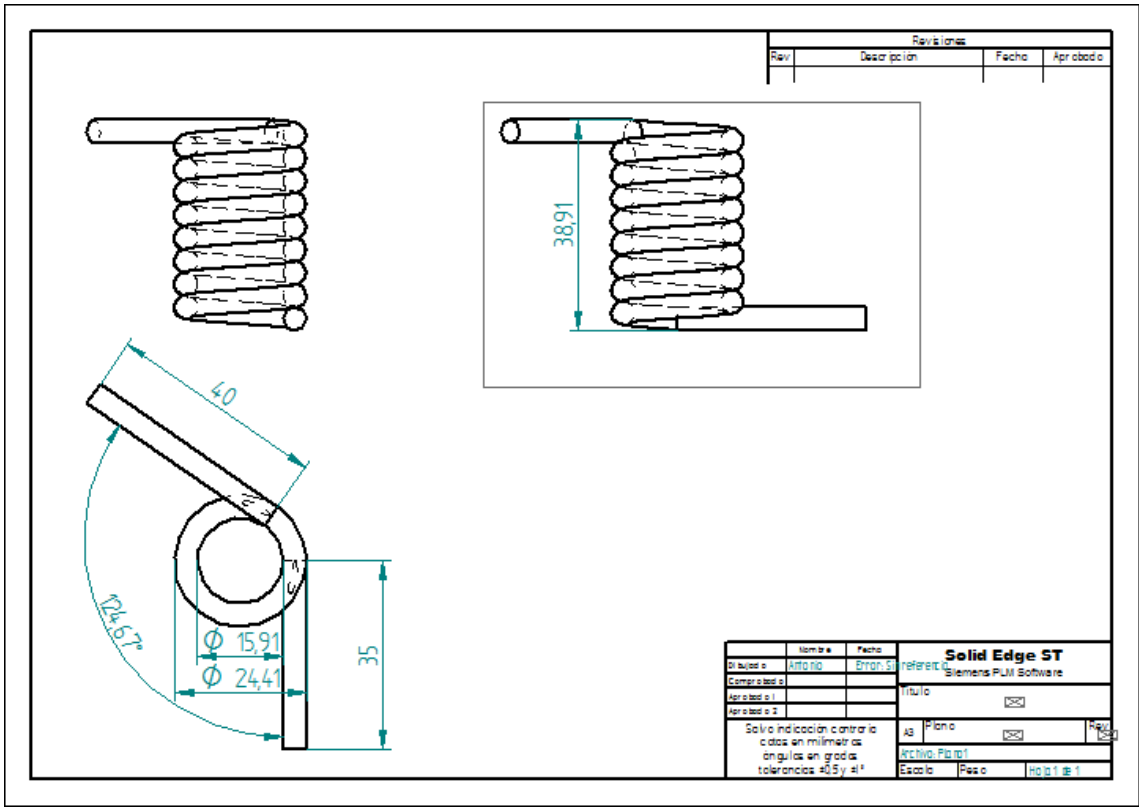

Figura 43: Cotas Resorte

En la figura 43 se puede comprobar como las medidas de la pieza dibujada se corresponden con los resultados calculados por el programa.

Para comprobar el ángulo a partir de las espiras de la ventana de resultado basta con hacer la conversión de vueltas a grados.

Nc = 8,15369 es decir 8 vueltas completas más 0,15369 vueltas, cada vuelta se

corresponde con 360º por tanto el ángulo con respecto a la vertical es de 55,3284º, es decir 124,6716º que se corresponde con el plano mostrado (teniendo en cuenta el redondeo de Solid Edge).

El siguiente paso es realizar los cálculos para compararlos con los obtenidos mediante el programa.

$$k = \frac{M_{max} - M_{pre}}{a_f - a_i} = \frac{3,55 - 1,12}{78 - 55,33} = 0,1071\ Nm/grad$$

Siendo el límite de fatiga sin corregir S'_e de 700 MPa se calcula la primera estimación de S_f, teniendo en cuenta que para la primera iteración se supone un K_b =1,12 y que acabado superficial del material es "acabado fino" y un factor de confiabilidad del 50% el resultado será:

$K_a = 1,58 \,.\, (2020)^{-0,085}$

$S_e = S'_e \,.\, 0,82737$

$$S_f = \frac{S_e}{1 + \frac{(1+x)\cdot S_e}{2\cdot S_{ut}}} = \frac{579,162}{1 + \frac{(1+(1,12/3,55))\cdot 579,162}{2\cdot 2020}} = 487,27\,MPa$$

$$d = \sqrt[3]{\frac{32\cdot M_{max}}{\pi\cdot S_f}} = \sqrt[3]{\frac{32\cdot 3,55}{\pi\cdot 420540000}} = 0,004202\,m$$

Por tanto d es igual a 4.5 mm si se selecciona un diámetro normalizado.

$$K_b = \left(\frac{d}{7.62}\right)^{-0,1133} = 1,0697$$

S_e= 700 . 0,82737 . 1,06149= 619,55 MPa

$$S_f = \frac{S_e}{1 + \frac{(1+x)\cdot S_e}{2\cdot S_{ut}}} = \frac{619,55}{1 + \frac{(1+(1,12/3,55))\cdot 619,55}{2\cdot 2020}} = 515,55\,MPa$$

$$d = \sqrt[3]{\frac{32 \cdot M_{max}}{\pi \cdot S_f}} = \sqrt[3]{\frac{32 \cdot 3{,}55}{\pi \cdot 515550000}} = 0{,}00412\, m$$

Con este diámetro de hilo se vuelve a obtener 4.25mm como diámetro normalizado y dado que 4.25 es el diámetro para el que se ha calculado el facto de tamaño K_b al volver a realizar la iteración se volvería siempre al mismo resultado, por tanto el hilo debe tener un diámetro normalizado de 4.25mm, como se muestra el la ventana del programa.

En el siguiente ejemplo se impone un diámetro interior mínimo, conservando el resto de valores:

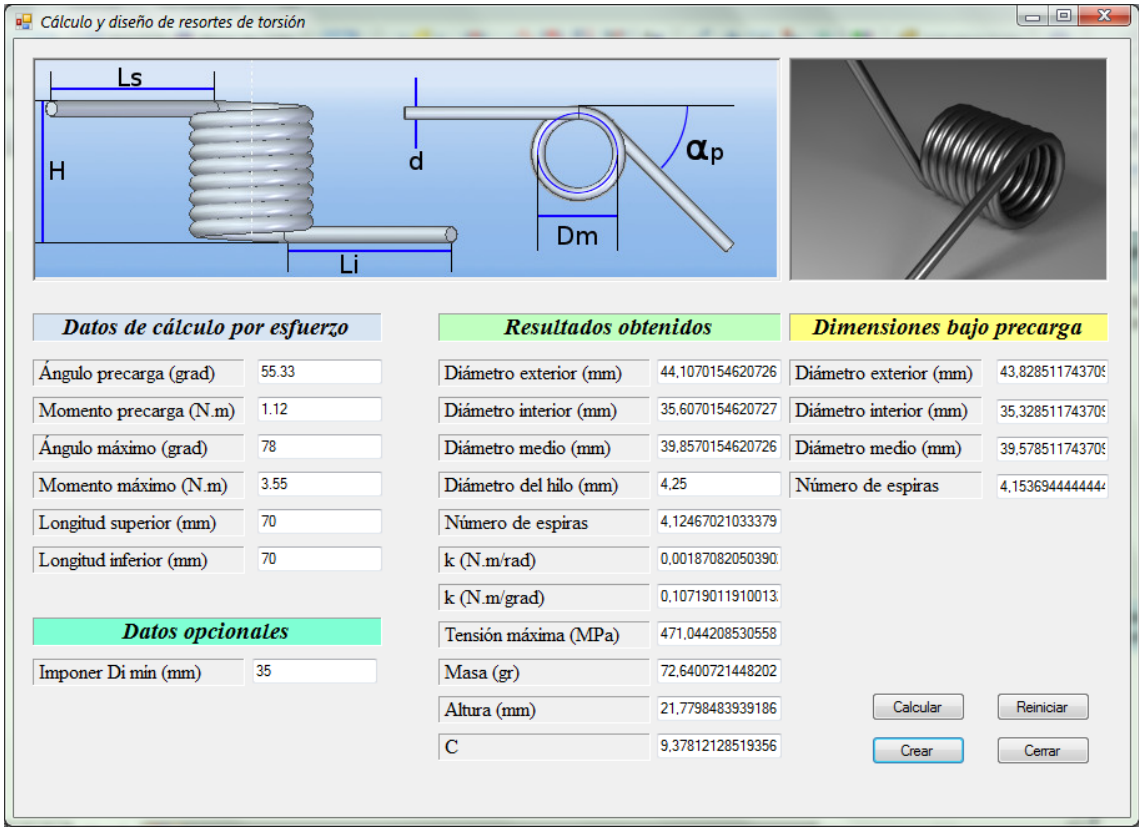

Figura 44: Ejemplo 2, pantalla de cálculos

Los cálculos relativos a la tensión son iguales que para el ejemplo anterior. El diámetro mínimo interior impuesto es de 35 milímetros, el programa realiza iteraciones hasta que encuentra el número de vueltas que cumple con los ángulos impuestos y

permite un diámetro interior lo más cercano posible a la limitación impuesta, en este caso el diámetro interior es de 35,328 mm. Como puede apreciarse en el siguiente plano:

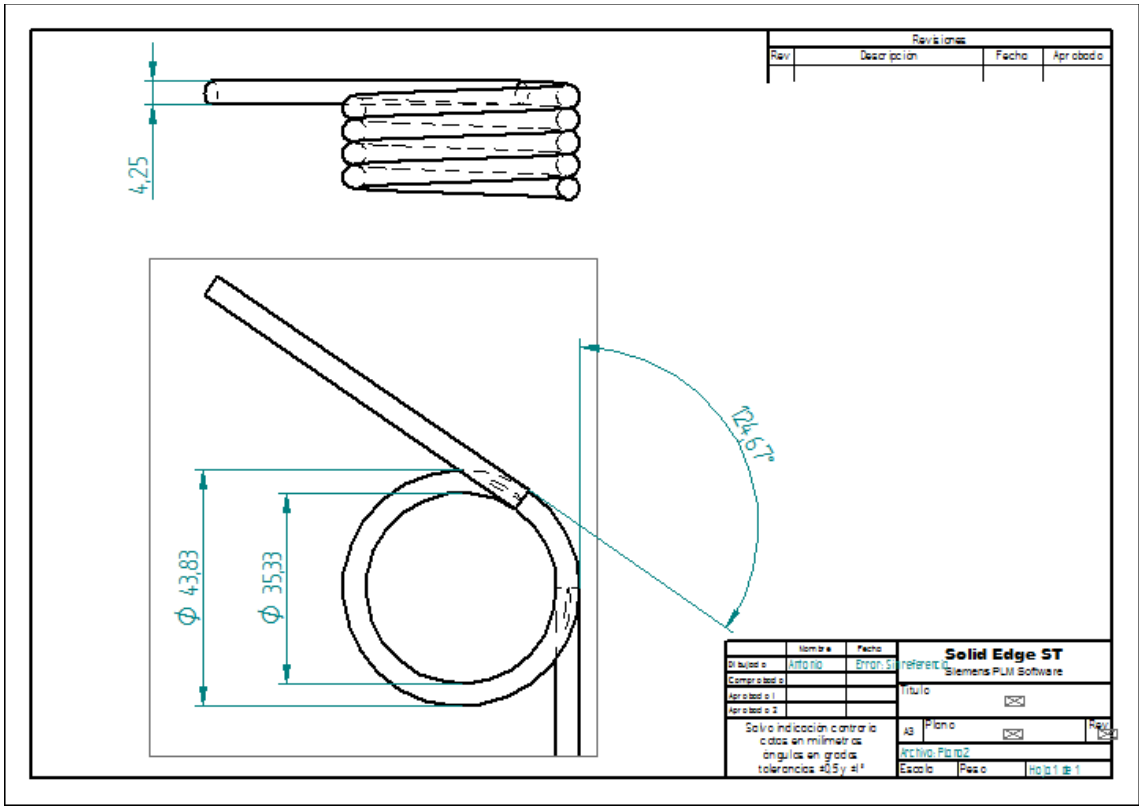

Figura 45: Cotas resorte 2

A modo de comprobación se calcula la constante del resorte a través del sus dimensiones :

$$k = \frac{M}{\theta} = \frac{d^4 E}{3888 \mathrm{DN}} = \frac{(4{,}25 \cdot 10^{-3})^4 \cdot 210 \cdot 10^9}{3880 \cdot 39{,}857 \cdot 10^{-3} \cdot 4{,}12467} = 0{,}10719\ Nm/grad$$

Valor que coincide con el suministrado por el programa que, recordemos de su código, lo calcula a través de los momentos y la variación del ángulo, por tanto este cálculo sirve de doble comprobación.

7 Conclusiones.

Este proyecto se ha centrado en la automatización de un proceso de diseño de un elemento en tres dimensiones que se implementado con una herramienta capaz de realizar cálculos matemáticos introducidos a través de una interfaz gráfica, dando como resultado una herramienta con un claro enfoque ingenieril.

Parte de la justificación del proyecto era complementar la paleta de herramientas incluidas en Solid Edge, en especial Engineering Reference, que sirvió como punto de partida e inspiración del macro creado.

La intención en todo momento ha sido crear un programa que, aprovechando la funcionalidad de Solid Edge, permita al usuario crear el elemento mecánico objeto del proyecto, un resorte helicoidal de torsión, de la manera más sencilla posible, garantizando en la medida de lo posible el éxito a la hora de generar un elemento mecánico capaz de satisfacer las necesidades requeridas sin sobrepasar los límites de resistencia estática y dinámica del material elegido.

A la hora de realizar la interfaz del programa y seleccionar las variables ofrecidas al usuario para diseñar el resorte se ha priorizado la facilidad de cálculo del resorte, desde el punto de vista del usuario, y maximizar las posibilidades de éxito. Ofrecer más variables no complicaría la redacción del código, quizá lo haría más largo pero no más complejo, sin embargo a medida que aumentan las variables configuradas por el usuario también aumenta la posibilidad de que surjan conflictos con los resultados obtenidos, disminuyendo la posibilidad de obtener un resorte el que no se sobrepasen los límites del material que lo conforma.

Otro aspecto a comentar como conclusión del proyecto es la comparativa entre el método de trabajo “tradicional” a través de la interfaz del programa y la creación de macros y establecer en qué circunstancias tiene sentido desarrollar un macro.

La utilización de macros supondrá una gran ventaja en términos de productividad a la hora de desarrollar tareas repetitivas que han de realizarse

frecuentemente, donde quizá no se necesite un alto grado de flexibilidad pero si reducir en la manera de lo posible el tiempo invertido. También será posible agregar nuevas funciones al software sobre el que se ejecuta el macro, ésta es sin lugar a dudas la mejor ventaja que pude ofrecer el uso de macros, agregar nuevas funciones sirviéndose del entorno de trabajo del software anfitrión, lo cual permitirá satisfacer necesidades no cubiertas por las herramientas incluidas por defecto en el software.

Las macroinstrucciones o macros tendrán a su disposición todas y cada una de las herramientas propias del programa sobre el que se ejecutan, lo cual darán al programador la posibilidad de replicar y automatizar la creación de cualquier objeto u operación realizable a través de la interfaz del programa.

Por otra parte el uso de macros plantea una desventaja manifiesta, sobretodo desde el punto de vista del programador. El desarrollo de un macro puede ser una tarea ardua y larga, el tiempo necesario para desarrollar un macro puede equivaler al tiempo empleado en la generación de muchas piezas u operaciones similares a las que se desee automatizar. A esto hay que sumarle el largo proceso de aprendizaje que puede ser necesario para dominar con soltura la programación de macros.

Por este motivo la creación de macros sólo tiene sentido para automatizar tareas que se realicen muy frecuentemente, implementar la funcionalidad del software y cubrir necesidades no satisfechas o por motivos comerciales. En este caso el macro desarrollado para Solid Edge viene a cubrir una necesidad no satisfecha por las herramientas incluidas por defecto, una herramienta capaz de calcular y modelar resortes helicoidales de torsión.

8. Futuros estudios.

- Añadir al macro la posibilidad de elegir entre distintos tipos de terminaciones que se adapten a las necesidades del usuario.

- Otro elemento carente en el Engineer Reference son las barras de torsión, así que otra posible opción de cara estudios y trabajos futuros sería añadir la posibilidad de calcular barras y diseñar barras de torsión, dado que también es un elemento mecánico relativamente común, especialmente en la industria del automóvil. Por tanto añadir dentro del macro la posibilidad de calcular resortes o barras de torsión sería otro posible trabajo.

- Añadir la posibilidad en la interfaz de introducir todos los parámetro de un resorte de torsión (material, cargas, desplazamientos, etc) y que sea capaz de estimar la vida útil del resorte y predecir su momento de fallo.

- El macro diseña todo los resortes con un perfil circular, pero en muchas aplicaciones son utilizados alambres con otro tipo de perfiles, especialmente perfiles cuadrado. Versiones futuras del macro deberían incluir al menos la posibilidad de seleccionar entre perfil circular y cuadrado.

- El macro genera la pieza y el usuario debe acotarla manualmente para comprobar que efectivamente las dimensiones se corresponden con las calculadas, agregar la generación automática de cotas, de forma opcional es otra implementación necesaria.

- Agregar en la interfaz la opción de elegir en qué entorno se quiere visualizar la pieza, o bien en el entorno pieza, o bien en el entorno plano.

9 Bibliografía.

[1] “Apuntes de la asignatura Diseño de Máquinas” de Ingeniería Técnica Industrial Mecánica.

[2] “Solid Edge ST7 SDK”:
(http://support.industrysoftware.automation.siemens.com/training/se/107/api/webframe.html)

[3] “Visual Basic 2012 (VB.NET) Los fundamentos del lenguaje”. Thierry Groussard

[4] “Diseño de elementos de máquinas I”. Jorge F. Ma San Zapata.

[5] “.NET Programmer´s Guide. Solid Edge with Synchronous Technology API.”

[6] “Solid Edsge ST. Tradicional y síncrono”. Rafael Gutiérrez Olivar, Lidia Esteban Viñado, Esther Pascual Albarracín.

[7] Apuntes de la asignatura “Diseño mecánico” de 4º de Ingeniería Industrial a través del open course:
http://ocw.uc3m.es/ingenieria-mecanica/diseno-mecanico-1

[8]A través de internet:
http://blog.utp.edu.co/lvanegas/dis1/
“Diseño de resortes capítulo 5” :
http://www.frbb.utn.edu.ar/frbb/images/carreras/elementosdemaquinas/cap05-01.pdf

[9] Microsoft Windows Dev Center a través de:
https://msdn.microsoft.com/en-us/library/windows/desktop/ee663262(v=vs.85).aspx

10. Anexo. Código completo del programa.

```
Imports System.Runtime.InteropServices
Imports SolidEdgeConstants

Public Class Form1
    Dim k_rad As Double
    Dim k_grad As Double
    Dim M As Double
    Dim Se As Double
    Dim Sec As Double
    Dim Sf As Double
    Dim Su As Double
    Dim xa As Double
    Dim Ka As Double
    Dim Kb As Double
    Dim d(47) As Double
    Dim df As Double
    Dim dfa As Double
    Dim da As Double
    Dim Dm As Double
    Dim De As Double
    Dim Di As Double
    Dim Na As Double
    Dim N As Double
    Dim ap As Double
    Dim am As Double
    Dim ai As Double
    Dim Mf As Double
    Dim ND As Double
    Dim C As Double
    Dim LongS As Double
    Dim LongI As Double
    Dim L As Double
    Dim Tmax As Double
    Dim DiametroNormalizado(47) As Double
    Dim c1 As Integer
    Dim c2 As Integer
    Dim c3 As Integer
    Dim c4 As Integer
    Dim c5 As Integer
    Dim Nc As Double
    Dim Dmc As Double
    Dim Dic As Double
    Dim Dec As Double

    'Variables para Solid Edge

    Dim objApplication As SolidEdgeFramework.Application = Nothing
    Dim objDocuments As SolidEdgeFramework.Documents = Nothing
```

```
Dim objPart As SolidEdgePart.PartDocument = Nothing
Dim objProfileSets As SolidEdgePart.ProfileSets = Nothing
Dim objProfileSet As SolidEdgePart.ProfileSet = Nothing
Dim objProfiles As SolidEdgePart.Profiles = Nothing
Dim objProfile As SolidEdgePart.Profile = Nothing
Dim objPlanoRef As SolidEdgePart.RefPlanes = Nothing
Dim objModels As SolidEdgePart.Models = Nothing
Dim objModel As SolidEdgePart.Model = Nothing
Dim aProfiles As Array
Dim objLines2d As SolidEdgeFrameworkSupport.Lines2d = Nothing
Dim objLine2d As SolidEdgeFrameworkSupport.Line2d = Nothing
Dim objEje As SolidEdgePart.RefAxis = Nothing
Dim sketches As SolidEdgePart.Sketchs = Nothing
Dim sketch As SolidEdgePart.Sketch = Nothing
Dim objcircles2d As SolidEdgeFrameworkSupport.Circles2d = Nothing
Dim objcircle2d As SolidEdgeFrameworkSupport.Circle2d = Nothing
Dim objRefplanes As SolidEdgePart.RefPlanes = Nothing
Dim objRPlanes As SolidEdgePart.RefPlanes
Dim objPPlane(1) As SolidEdgePart.RefPlane
Dim objRPParallel(1) As SolidEdgePart.RefPlane
Dim objLineas As SolidEdgeFrameworkSupport.Lines2d = Nothing
Dim objLinea As SolidEdgeFrameworkSupport.Lines2d = Nothing
Dim objProfil As SolidEdgePart.Profile
Dim r As Single
Dim Radio As Single
Dim x As Single
Dim y As Single


'Código para interfaz y cáculos
Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
    'Rellenar matriz con diámetro de hilo normalizados
    DiametroNormalizado(0) = (1 / 1000)
    DiametroNormalizado(1) = (1.05 / 1000)
    DiametroNormalizado(2) = (1.1 / 1000)
    DiametroNormalizado(3) = (1.2 / 1000)
    DiametroNormalizado(4) = (1.25 / 1000)
    DiametroNormalizado(5) = (1.3 / 1000)
    DiametroNormalizado(6) = (1.4 / 1000)
    DiametroNormalizado(7) = (1.5 / 1000)
    DiametroNormalizado(8) = (1.6 / 1000)
    DiametroNormalizado(9) = (1.7 / 1000)
    DiametroNormalizado(10) = (1.8 / 1000)
    DiametroNormalizado(11) = (1.9 / 1000)
    DiametroNormalizado(12) = (2 / 1000)
    DiametroNormalizado(13) = (2.1 / 1000)
    DiametroNormalizado(14) = (2.25 / 1000)
    DiametroNormalizado(15) = (2.4 / 1000)
    DiametroNormalizado(16) = (2.5 / 1000)
    DiametroNormalizado(17) = (2.6 / 1000)
    DiametroNormalizado(18) = (2.7 / 1000)
```

```
DiametroNormalizado(19) = (2.8 / 1000)
DiametroNormalizado(20) = (2.9 / 1000)
DiametroNormalizado(21) = (3 / 1000)
DiametroNormalizado(22) = (3.1 / 1000)
DiametroNormalizado(23) = (3.2 / 1000)
DiametroNormalizado(24) = (3.3 / 1000)
DiametroNormalizado(25) = (3.4 / 1000)
DiametroNormalizado(26) = (3.5 / 1000)
DiametroNormalizado(27) = (3.6 / 1000)
DiametroNormalizado(28) = (3.7 / 1000)
DiametroNormalizado(29) = (3.8 / 1000)
DiametroNormalizado(30) = (4 / 1000)
DiametroNormalizado(31) = (4.25 / 1000)
DiametroNormalizado(32) = (4.5 / 1000)
DiametroNormalizado(33) = (4.75 / 1000)
DiametroNormalizado(34) = (5 / 1000)
DiametroNormalizado(35) = (5.3 / 1000)
DiametroNormalizado(36) = (5.6 / 1000)
DiametroNormalizado(37) = (6 / 1000)
DiametroNormalizado(38) = (6.3 / 1000)
DiametroNormalizado(39) = (6.5 / 1000)
DiametroNormalizado(40) = (7 / 1000)
DiametroNormalizado(41) = (7.5 / 1000)
DiametroNormalizado(42) = (8 / 1000)
DiametroNormalizado(43) = (8.5 / 1000)
DiametroNormalizado(44) = (9 / 1000)
DiametroNormalizado(45) = (9.5 / 1000)
DiametroNormalizado(46) = (10 / 1000)
Kb = 1.12
c4 = 0
c5 = 2
' Primer paso, cálculo de la constante del muelle
k_grad = (Val(TextBox4.Text) - Val(TextBox2.Text)) / (Val(TextBox3.Text) - Val(TextBox1.Text))
k_rad = k_grad * ((Math.PI) / 180)
TextBox12.Text = Val(k_rad)
TextBox13.Text = Val(k_grad)
' Segundo paso, cálculo del diámetro del hilo y tensión maxima Sf
Ka = 0.827374669
Su = 2020000000
Se = 700000000 * Ka
xa = Val(TextBox2.Text) / Val(TextBox4.Text)
M = Val(TextBox4.Text)
While c4 = 0
   If df > 2.8 Then
      Sec = Se * Kb
   Else
      Kb = 1.12
      Sec = Se * Kb
   End If
   Sf = Sec / (1 + ((1 + xa) * (Sec / (2 * Su))))
   da = ((32 * M) / ((Math.PI) * Sf)) ^ (1 / 3)
```

```
c1 = 0
c2 = 0
c3 = 0
While c3 = 0
  If da < DiametroNormalizado(46) Then
    While c2 = 0
      If (da > DiametroNormalizado(c1)) And (da < DiametroNormalizado(c1 + 1)) Then
        c2 = 1
        c3 = 1
        d(c5) = DiametroNormalizado(c1 + 1)
        TextBox14.BackColor = Color.White
        TextBox4.BackColor = Color.White
      Else
        c1 = c1 + 1
      End If
    End While
    Kb = ((d(c5) * 1000) / 7.62) ^ (-0.1133)
  Else
    MessageBox.Show("Momento demasiado grande, reintroduzca los datos")
    c3 = 1
    TextBox14.BackColor = Color.OrangeRed
    TextBox4.BackColor = Color.OrangeRed
  End If
End While
If d(c5) = d(c5 - 1)  Or d(c5) = d(c5 –2) Then
  c4 = 1
Else
  c5 = c5 + 1
End If
End While
df = d(c5)
TextBox10.Text = Val(df * 1000)
' Determinar Diámetro medio multiplicado por Número de vueltas, determinar ángulo inicial respecto sistema de
referencia
Mf = Val(TextBox4.Text)
ai = Val(TextBox3.Text) - (Mf / k_grad)
ND = ((df ^ 4) * 210000000000) / (3888 * k_grad)
' Determinar Diámetro medio y Número de espiras para un C acotado entre 10 y 4
If Val(TextBox22.Text) = 0 Then

  Na = 5
  N = Na + (ai / 360)
  Dm = (ND / N)
  C = (Dm / df)
  If C > 5 Then
    While C > 5
      Na = Na + 1
      N = Na + (ai / 360)
      Dm = (ND / N)
      C = (Dm / df)
    End While
```

```
        End If
        If C < 3 Then
            While C < 3
                Na = Na - 1
                N = Na + (ai / 360)
                Dm = (ND / N)
                C = (Dm / df)
            End While

        End If
    Else
        'Determinar D y N para un diámetro interior mínimo
        Na = 100
        While (Dm - df) < (Val(TextBox22.Text) / 1000)
            Na = Na - 1
            N = Na + (ai / 360)
            Dm = (ND / N)
            C = (Dm / df)

        End While
    End If
    'Mostrar resultado para precarga
    TextBox11.Text = Val(N)
    TextBox9.Text = Val(Dm) * 1000
    TextBox17.Text = Val(C)
    TextBox7.Text = Val(Dm + df) * 1000
    TextBox8.Text = Val(Dm - df) * 1000
    'Determinar longitud total de los extremos
    LongS = Val(TextBox5.Text) / 1000
    LongI = Val(TextBox6.Text) / 1000
    L = LongS + LongI
    'Determinar masa
    M = 7800000 * (((Math.PI) * (df ^ 2)) * 0.25) * ((Math.PI) * Dm * N + L)
    TextBox15.Text = Val(M)
    'Mostrar Altura y tensión máxima
    TextBox14.Text = Val((Sf * ((da / df) ^ 3)) / 1000000)
    TextBox16.Text = (N + 1) * df * 1000
    'Asignar dimensiones en precarga
    Nc = N + ((Val(TextBox2.Text) / k_grad) / 360)
    Dmc = Dm * (N / Nc)
    Dic = Dmc - df
    Dec = Dmc + df
    TextBox18.Text = Val(Dec) * 1000
    TextBox19.Text = Val(Dic) * 1000
    TextBox20.Text = Val(Dmc) * 1000
    TextBox21.Text = Val(Nc)

End Sub

' Acción para el botón de reiniciar
```

```
Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
    c1 = 0
    c2 = 0
    c3 = 0
    c4 = 0
    c5 = 2
    TextBox1.Text = 0
    TextBox2.Text = 0
    TextBox3.Text = 0
    TextBox4.Text = 0
    TextBox5.Text = 0
    TextBox6.Text = 0
    TextBox7.Text = 0
    TextBox8.Text = 0
    TextBox9.Text = 0
    TextBox10.Text = 0
    TextBox11.Text = 0
    TextBox12.Text = 0
    TextBox13.Text = 0
    TextBox14.Text = 0
    TextBox15.Text = 0
    TextBox16.Text = 0
    TextBox17.Text = 0
    TextBox18.Text = 0
    TextBox19.Text = 0
    TextBox20.Text = 0
    TextBox21.Text = 0
    TextBox22.Text = 0
    Kb = 1.3
    da = 0
    N = 0
    Dm = 0
    Na = 0
    LongS = 0
    LongI = 0

End Sub
'Acción del botón Cerrar
Private Sub Button4_Click(sender As System.Object, e As System.EventArgs) Handles Button4.Click
    End
End Sub
'Acción para el botón crear
Private Sub Button3_Click(sender As System.Object, e As System.EventArgs) Handles Button3.Click
    Muelle()
    End
End Sub

'Instrucciones para Solid Edge
Public Sub Muelle()

    Radio = 0.5 * Dmc
```

```
x = Radio
y = 0
r = df / 2

'EXTRUSION SUPERIOR
If LongS > 0 Then
   Try

      objApplication = Marshal.GetActiveObject("SolidEdge.Application")
      Radio = 0.5 * Dmc
      x = Radio
      r = (df / 2)

      ' Creacion del perfil
      objDocuments = objApplication.Documents
      objPart = objApplication.ActiveDocument
      objProfileSets = objPart.ProfileSets
      objProfileSet = objProfileSets.Add()
      objProfiles = objProfileSet.Profiles
      objPlanoRef = objPart.RefPlanes
      ' Generación del boceto con la circunferencia y el eje de revolución
      sketches = objPart.Sketches
      sketch = sketches.Add()
      objProfile = objProfiles.Add(objPlanoRef.Item(3))
      objDocuments = objApplication.Documents
      objPart = objApplication.ActiveDocument
      objProfileSets = objPart.ProfileSets
      objProfileSet = objProfileSets.Add()
      objProfiles = objProfileSet.Profiles
      objRefplanes = objPart.RefPlanes
      objLines2d = objProfile.Lines2d
      objEje = objProfile.SetAxisOfRevolution(objLines2d.AddBy2Points(0, 0, 0, 0.1))
      'Crear plano auxiliar
      objRPlanes = objPart.RefPlanes
      objPPlane(1) = objRefplanes.Item(3)
      objRPParallel(1) = objRPlanes.AddAngularByAngle(ParentPlane:=objPPlane(1), _
                                                      Angle:=((2) * (Nc) * (Math.PI)),_
NormalSide:=SolidEdgePart.ReferenceElementConstants.igPlaneRotateLeft, Pivot:=objEje, _
                                                      PivotOrigin:=SolidEdgePart.ReferenceElementConstants.igPlaneAlignY,
Local:=False)

      objProfil = objProfiles.Add(objRPParallel(1))
      sketches = objPart.Sketches
      sketch = sketches.Add()
      objcircles2d = objProfil.Circles2d

      If Nc < 4 Then
         dfa = df + 0.00001
      Else
         If Nc > 14 Then
            If Nc > 17 Then
```

```
                dfa = df + 0.0000001
            Else
                dfa = df + 0.000002
            End If
        Else
            dfa = df + 0.000003
        End If
    End If

    objcircle2d = objcircles2d.AddByCenterRadius(x, ((Nc * (dfa))), r)
    objProfil.End( _
    SolidEdgePart.ProfileValidationType.igProfileClosed)
    objProfil.Visible = False
    aProfiles = Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
    aProfiles.SetValue(objProfil, 0)
    'Generar extrusión
    objModels = objPart.Models
    objModel = objModels.AddFiniteExtrudedProtrusion( _
    aProfiles.Length, _
    aProfiles, _
    SolidEdgePart.FeaturePropertyConstants.igLeft, _
    LongS)
    objRPParallel(1).Visible = False

Catch ex As Exception
    Console.WriteLine(ex.Message)

    If Not (objModel Is Nothing) Then
        Marshal.ReleaseComObject(objModel)
        objModel = Nothing
    End If
    If Not (objModels Is Nothing) Then
        Marshal.ReleaseComObject(objModels)
        objModels = Nothing
    End If
    If Not (objLine2d Is Nothing) Then
        Marshal.ReleaseComObject(objLine2d)
        objLine2d = Nothing
    End If
    If Not (objLines2d Is Nothing) Then
        Marshal.ReleaseComObject(objLines2d)
        objLines2d = Nothing
    End If
    If Not (objPlanoRef Is Nothing) Then
        Marshal.ReleaseComObject(objPlanoRef)
        objPlanoRef = Nothing
    End If
    If Not (objProfile Is Nothing) Then
        Marshal.ReleaseComObject(objProfile)
        objProfile = Nothing
    End If
```

```
        If Not (objcircle2d Is Nothing) Then
            Marshal.ReleaseComObject(objcircle2d)
            objcircle2d = Nothing
        End If
        If Not (objcircles2d Is Nothing) Then
            Marshal.ReleaseComObject(objcircles2d)
            objcircles2d = Nothing
        End If
        If Not (objProfiles Is Nothing) Then
            Marshal.ReleaseComObject(objProfiles)
            objProfiles = Nothing
        End If
        If Not (objProfileSet Is Nothing) Then
            Marshal.ReleaseComObject(objProfileSet)
            objProfileSet = Nothing
        End If
        If Not (objProfileSets Is Nothing) Then
            Marshal.ReleaseComObject(objProfileSets)
            objProfileSets = Nothing
        End If
        If Not (objPart Is Nothing) Then
            Marshal.ReleaseComObject(objPart)
            objPart = Nothing
        End If
        If Not (objDocuments Is Nothing) Then
            Marshal.ReleaseComObject(objDocuments)
            objDocuments = Nothing
        End If
        If Not (objApplication Is Nothing) Then
            Marshal.ReleaseComObject(objApplication)
            objApplication = Nothing
        End If
    End Try
End If

' EXTRUSION INFERIOR
If LongI > 0 Then

    Try
        objApplication = Marshal.GetActiveObject("SolidEdge.Application")
        objDocuments = objApplication.Documents
        objPart = objApplication.ActiveDocument
        objProfileSets = objPart.ProfileSets
        objProfileSet = objProfileSets.Add()
        objProfiles = objProfileSet.Profiles
        objRefplanes = objPart.RefPlanes
        'Generar perfil
        sketches = objPart.Sketches
        sketch = sketches.Add()
        objProfile = objProfiles.Add(objRefplanes.Item(3))
        objcircles2d = objProfile.Circles2d
```

```
        objcircle2d = objcircles2d.AddByCenterRadius(x, 0, r)
        objProfile.End( _
        SolidEdgePart.ProfileValidationType.igProfileClosed)
        objProfile.Visible = False
        aProfiles = Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
        aProfiles.SetValue(objProfile, 0)
        'Generar extrusión
        objModels = objPart.Models
        objModel = objModels.AddFiniteExtrudedProtrusion( _
        aProfiles.Length, _
        aProfiles, _
        SolidEdgePart.FeaturePropertyConstants.igRight, _
        LongI)
    Catch ex As Exception
        Console.WriteLine(ex.Message)
    Finally
        If Not (objModel Is Nothing) Then
            Marshal.ReleaseComObject(objModel)
            objModel = Nothing
        End If
        If Not (objModels Is Nothing) Then
            Marshal.ReleaseComObject(objModels)
            objModels = Nothing
        End If
        If Not (objcircle2d Is Nothing) Then
            Marshal.ReleaseComObject(objcircle2d)
            objcircle2d = Nothing
        End If
        If Not (objcircles2d Is Nothing) Then
            Marshal.ReleaseComObject(objcircles2d)
            objcircles2d = Nothing
        End If
        If Not (objLine2d Is Nothing) Then
            Marshal.ReleaseComObject(objLine2d)
            objLine2d = Nothing
        End If
        If Not (objLines2d Is Nothing) Then
            Marshal.ReleaseComObject(objLines2d)
            objLines2d = Nothing
        End If
        If Not (objRefplanes Is Nothing) Then
            Marshal.ReleaseComObject(objRefplanes)
            objRefplanes = Nothing
        End If
        If Not (objProfile Is Nothing) Then
            Marshal.ReleaseComObject(objProfile)
            objProfile = Nothing
        End If
        If Not (objProfiles Is Nothing) Then
            Marshal.ReleaseComObject(objProfiles)
            objProfiles = Nothing
```

```
        End If
        If Not (objProfileSet Is Nothing) Then
            Marshal.ReleaseComObject(objProfileSet)
            objProfileSet = Nothing
        End If
        If Not (objProfileSets Is Nothing) Then
            Marshal.ReleaseComObject(objProfileSets)
            objProfileSets = Nothing
        End If
        If Not (objPart Is Nothing) Then
            Marshal.ReleaseComObject(objPart)
            objPart = Nothing
        End If
        If Not (objDocuments Is Nothing) Then
            Marshal.ReleaseComObject(objDocuments)
            objDocuments = Nothing
        End If
        If Not (objApplication Is Nothing) Then
            Marshal.ReleaseComObject(objApplication)
            objApplication = Nothing
        End If
    End Try
End If

'EXTRUSION HELICOIDAL

Try
    objApplication = Marshal.GetActiveObject("SolidEdge.Application")
    Radio = 0.5 * Dmc
    x = Radio
    r = (df / 2)
    objDocuments = objApplication.Documents
    objPart = objApplication.ActiveDocument
    objProfileSets = objPart.ProfileSets
    objProfileSet = objProfileSets.Add()
    objProfiles = objProfileSet.Profiles
    objPlanoRef = objPart.RefPlanes
    ' Generación del boceto con la circunferencia y el eje de revolución
    sketches = objPart.Sketches
    sketch = sketches.Add()
    objProfile = objProfiles.Add(objPlanoRef.Item(3))
    objcircles2d = objProfile.Circles2d
    objcircle2d = objcircles2d.AddByCenterRadius(x, 0, r)
    objLines2d = objProfile.Lines2d

    If LongS = 0 Then
        objLines2d = objProfile.Lines2d
        objEje = objProfile.SetAxisOfRevolution(objLines2d.AddBy2Points(0, 0, 0, 0.1))

    End If
    objProfile.End(SolidEdgePart.ProfileValidationType.igProfileClosed)
```

```
    objProfile.Visible = False
    aProfiles = Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
    aProfiles.SetValue(objProfile, 0)
    'Generar extrusión helicoidal
    objModels = objPart.Models
    objModel = objModels.AddFiniteBaseHelix(HelixAxis:=objEje, _
                                AxisStart:=SolidEdgePart.FeaturePropertyConstants.igStart, _
                                NumCrossSections:=1, CrossSectionArray:=aProfiles, _
                                ProfileSide:=SolidEdgePart.FeaturePropertyConstants.igRight, _
                                Height:=0, Pitch:=df + 0.000001, _
                                NumberOfTurns:=(Nc), HelixDir:=SolidEdgePart.FeaturePropertyConstants.igRight)



Catch ex As Exception
    Console.WriteLine(ex.Message)
End Try
If Not (objModel Is Nothing) Then
    Marshal.ReleaseComObject(objModel)
    objModel = Nothing
End If
If Not (objModels Is Nothing) Then
    Marshal.ReleaseComObject(objModels)
    objModels = Nothing
End If
If Not (objLine2d Is Nothing) Then
    Marshal.ReleaseComObject(objLine2d)
    objLine2d = Nothing
End If
If Not (objLines2d Is Nothing) Then
    Marshal.ReleaseComObject(objLines2d)
    objLines2d = Nothing
End If
If Not (objcircle2d Is Nothing) Then
    Marshal.ReleaseComObject(objcircle2d)
    objcircle2d = Nothing
End If
If Not (objcircles2d Is Nothing) Then
    Marshal.ReleaseComObject(objcircles2d)
    objcircles2d = Nothing
End If
If Not (objPlanoRef Is Nothing) Then
    Marshal.ReleaseComObject(objPlanoRef)
    objPlanoRef = Nothing
End If
If Not (objProfile Is Nothing) Then
    Marshal.ReleaseComObject(objProfile)
    objProfile = Nothing
End If
If Not (objProfiles Is Nothing) Then
    Marshal.ReleaseComObject(objProfiles)
```

```
            objProfiles = Nothing
        End If
        If Not (objProfileSet Is Nothing) Then
            Marshal.ReleaseComObject(objProfileSet)
            objProfileSet = Nothing
        End If
        If Not (objProfileSets Is Nothing) Then
            Marshal.ReleaseComObject(objProfileSets)
            objProfileSets = Nothing
        End If
        If Not (objPart Is Nothing) Then
            Marshal.ReleaseComObject(objPart)
            objPart = Nothing
        End If
        If Not (objDocuments Is Nothing) Then
            Marshal.ReleaseComObject(objDocuments)
            objDocuments = Nothing
        End If
        If Not (objApplication Is Nothing) Then
            Marshal.ReleaseComObject(objApplication)
            objApplication = Nothing
        End If
    End Sub

End Class
```