Trabajo Fin de Grado
Ingeniería Aeroespacial

Fundamentos de programación de Visual Basic en CATIA V5. Aplicación al diseño de engranajes rectos y helicoidales de ejes paralelos.

Autor: Ángel Caballero Bazán

Tutora: Cristina Torrecillas Lozano

Dep. Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2014

Proyecto Fin de Grado
Ingeniería Aeroespacial

Fundamentos de programación de Visual Basic en CATIA V5. Aplicación al diseño de engranajes rectos y helicoidales de ejes paralelos.

Autor:
Ángel Caballero Bazán

Tutora:
Cristina Torrecillas Lozano
Profesora Contratada Doctora

Dep. de Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2014

Proyecto Fin de Carrera: Fundamentos de programación de Visual Basic en CATIA V5. Aplicación al diseño de engranajes rectos y helicoidales de ejes paralelos.

Autor:

Tutor:

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2014

El Secretario del Tribunal

A mi familia,

mis amigos y sobre todo a ti,

mi Cloti.

Resumen

El presente proyecto estudia el campo de aplicación de Visual Basic for Applications (VBA) al programa de diseño informático CATIA.

En primer lugar se realiza una introducción a la teoría de engranajes donde se estudiarán los tipos de engranajes existentes así como sus principales características y aplicaciones. Se describirá la geometría de la involuta y las principales características geométricas de engranajes rectos y helicoidales de ejes paralelos.

Se define VBA, lenguaje de macros de Microsoft Visual Basic, así como su acceso desde CATIA V5. Se explicará el entorno VBA, el modo en que el usuario podrá crear distintos interfaces para poder comunicarse con el programa y los objetos más importantes del módulo Mechanical Design (sketcher, Part Desing y Wireframe and Surface Design), así como las herramientas que éstos poseen para llevar a cabo distintos tipos de geometrías y modelos 3D.

Una vez familiarizados con el lenguaje de programación, se aplicarán dichos conocimientos para el desarrollo de una aplicación que genera engranajes de distintas geometrías.

El objetivo principal consiste en la realización de un software que permita al usuario obtener diferentes tipos de engranajes a partir de los parámetros más característicos de la geometría de los mismos. Se han realizado engranajes rectos (exteriores e interiores) y helicoidales (tanto simples como dobles) de ejes paralelos, y se introducen algunas posibles aplicaciones a desarrollar en un futuro.

Finalmente se explica el funcionamiento del interfaz que permitirá al usuario introducir los parámetros de entrada para realizar el engranaje que desee.

Objetivo y alcance del proyecto

El objetivo del proyecto es la familiarización con el lenguaje de programación VBA para facilitar la utilización de dicha tecnología para el desarrollo de engranajes. A pesar de que CATIA es la aplicación CAD/CAM más potente en el ámbito de la ingeniería, no posee herramientas específicas para el diseño de elementos mecánicos como engranajes, tuercas, etc. Por ello dicho proyecto abre nuevas vías de desarrollo en el diseño mecánico con CATIA.

En este proyecto en particular se va a desarrollar un software mediante el cual el usuario podrá diseñar de manera automática diferentes tipos de engranajes a partir de los parámetros principales que se requieren para el diseño de los mismos.

Por lo tanto, el alcance del proyecto no es realizar un estudio intensivo de la geometría y cinemática de los engranes, sino adquirir los conocimientos de programación necesarios y permitir que cualquier persona con la formación adecuada sea capaz de hacer uso del software realizado, o crear el suyo propio.

Índice de Figuras

Índice de Códigos

Índice de Tablas

1 Introducción a la teoría de engranajes

1.1 Introducción.

Se denomina engranaje o ruedas dentadas al mecanismo utilizado para transmitir potencia de un componente a otro dentro de una máquina. Los engranajes están formados por dos ruedas dentadas, de las cuales la mayor se denomina 'corona' y la menos 'piñón'. Un engranaje sirve para transmitir movimiento circular mediante contacto de ruedas dentadas. Una de las aplicaciones más importantes de los engranajes es la transmisión del movimiento desde el eje de una fuente de energía, como puede ser un motor de combustión interna o un motor eléctrico, hasta otro eje situado a cierta distancia y que ha de realizar un trabajo. De manera que una de las ruedas está conectada por la fuente de energía y es conocido como engranaje motor, y la otra está conectada al eje que debe recibir el movimiento del eje motor y que se denomina engranaje conducido. *Referencia[1]*

1.2 Historia de los engranajes.

Desde épocas muy remotas se han utilizado cuerdas y elementos fabricados en madera para solucionar los problemas de transporte, impulsión, elevación y movimiento. Nadie sabe a ciencia cierta dónde ni cuándo se inventaron los engranajes. La literatura de la antigua China, Grecia, Turquía y Damasco mencionan engranajes pero no aportan muchos detalles de los mismos.

Figura 1.Mecanismo de Anticicera

El mecanismo de engranajes más antiguo de cuyos restos disponemos es el mecanismo de Anticitera (Figura1). Se trata de una calculadora astronómica datada entre el 150 y el 100 a. C. y compuesta por al menos 30 engranajes de bronce con dientes triangulares. Presenta características tecnológicas avanzadas como por ejemplo trenes de engranajes epicicloidales que, hasta el descubrimiento de este mecanismo, se creían inventados en el siglo XIX. Por citas de Cicerónse sabe que el de Anticitera no fue un ejemplo aislado sino que existieron al menos otros dos mecanismos similares en

esa época, construidos por Arquímedes y por Posidonio. Por otro lado, a Arquímedes se le suele considerar uno de los inventores de los engranajes porque diseñó un tornillo sin fin.

En China también se han conservado ejemplos muy antiguos de máquinas con engranajes. Un ejemplo es el llamado "carro que apunta hacia el Sur" (120-250 d. C.), un ingenioso mecanismo que mantenía el brazo de una figura humana apuntando siempre hacia el Sur gracias al uso de engranajes diferenciales epicicloidales. Algo anteriores, de en torno a 50 d. C., son los engranajes helicoidales tallados en madera y hallados en una tumba real en la ciudad china de Shensi.

No está claro cómo se transmitió la tecnología de los engranajes en los siglos siguientes. Es posible que el conocimiento de la época del mecanismo de Anticitera sobreviviese y contribuyese al florecimiento de la ciencia y la tecnología en el mundo islámico de los siglos IX al XIII. Por ejemplo, un manuscrito andalusí del siglo XI menciona por vez primera el uso en relojes mecánicos tanto de engranajes epicíclicos como de engranajes segmentados. Los trabajos islámicos sobre astronomía y mecánica pueden haber sido la base que permitió que volvieran a fabricarse calculadoras astronómicas en la Edad Moderna. En los inicios del Renacimiento esta tecnología se utilizó en Europa para el desarrollo de sofisticados relojes, en la mayoría de los casos destinados a edificios públicos como catedrales.

Figura 2. Engranaje helicoidal de Leonardo

Leonardo da Vinci, muerto en Francia en 1519, dejó numerosos dibujos y esquemas de algunos de los mecanismos utilizados hoy diariamente, incluido varios tipos de engranajes de tipo helicoidal.

Los primeros datos que existen sobre la transmisión de rotación con velocidad angular uniforme por medio de engranajes, corresponden al año1674, cuando el famoso astrónomo danés Olaf Roemer (1644-1710) propuso la forma o perfil del diente en epicicloide.

Robert Willis (1800-1875), considerado uno de los primeros ingenieros mecánicos, fue el que obtuvo la primera aplicación práctica de la epicicloide al

emplearla en la construcción de una serie de engranajes intercambiables. De la misma manera, de los primeros matemáticos fue la idea del empleo de la evolvente de círculo en el perfil del diente, pero también se deben a Willis las realizaciones prácticas. A Willis se le debe la creación del odontógrafo, aparato que sirve para el trazado simplificado del perfil del diente de evolvente.

Es muy posible que fuera el francés Phillipe de Lahire el primero en concebir el diente de perfil en evolvente en 1695, muy poco tiempo después de que Roemer concibiera el epicicloidal.

La primera aplicación práctica del diente en evolvente fue debida al suizo Leonhard Euler (1707). En 1856, Christian Schiele descubrió el sistema de fresado de engranajes rectos por medio de la fresa madre, pero el procedimiento no se llevaría a la práctica hasta 1887, a base de la patente Grant.

Figura 3. Transmision antigua

En 1874, el norteamericano William Gleason inventó la primera fresadora de engranajes cónicos y gracias a la acción de sus hijos, especialmente su hija Kate Gleason (1865-1933), convirtió a su empresa Gleason Works, radicada en Rochester (Nueva York, EEUU) en una de los fabricantes de máquinas herramientas más importantes del mundo.

En 1897, el inventor alemán Robert Hermann Pfauter (1885-1914), inventó y patentó una máquina universal de dentar engranajes rectos y helicoidales por fresa madre. A raíz de este invento y otros muchos inventos y aplicaciones que realizó sobre el mecanizado de engranajes, fundó la empresa Pfauter Company que, con el paso del tiempo, se ha convertido en una multinacional fabricante de todo tipo de máquinas-herramientas.

En 1906, el ingeniero y empresario alemán Friedrich Wilhelm Lorenz (1842-1924) se especializó en crear maquinaria y equipos de mecanizado de engranajes y en 1906 fabricó una talladora de engranajes capaz de mecanizar los dientes de una rueda de 6 m de diámetro, módulo 100 y una longitud del dentado de 1,5 m.

Figura 4. Antigua grúa accionada con engranajes ubicada en el puerto de Sevilla

A finales del siglo XIX, coincidiendo con la época dorada del desarrollo de los engranajes, el inventor y fundador de la empresa Fellows Gear Shaper Company, Edwin R. Fellows (1846-1945), inventó un método revolucionario para mecanizar tornillos sin fin glóbicos tales como los que se montaban en las cajas de dirección de los vehículos antes de que fuesen hidráulicas.

En 1905, M. Chambon, de Lyon (Francia), fue el creador de la máquina para el dentado de engranajes cónicos por procedimiento de fresa madre. Aproximadamente por esas fechas André Citroën inventó los engranajes helicoidales dobles. *Referencia[1]*

1.3 Clasificación de engranajes.

Existen engranajes de diversos tamaños, con los dientes rectos o curvos y con distintos ángulos de inclinación. Se conectan entre sí de varias maneras para la transmisión de fuerza y movimiento en las máquinas. Sin embargo, sólo existen cuatro tipos básicos de engranajes. Todos actúan de modo que la rueda de un engranaje gira más rápido o más despacio que la otra, o se mueven en distinta dirección. La diferencia de velocidad entre dos engranajes produce un cambio en la fuerza que se transmite.

La principal clasificación de los engranajes se efectúa según la disposición de sus ejes de rotación y según los tipos de dentado. Según estos criterios existen los siguientes tipos:

-Ejes paralelos:

- Cilíndricos de dientes rectos.
- Cilíndricos de dientes helicoidales.
- Doble helicoidales.

-Ejes perpendiculares:

- Helicoidales cruzados
- Cónicos de dientes rectos.
- Cónicos de dientes helicoidales.
- Cónicos hipoides.
- De rueda y tornillo sinfín.

Figura 5. Tipos de engranajes

Se van a describir a continuación las características generales de los engranajes presentados anteriormente, así como sus principales funciones y diferencias.

1.3.1 Engranajes de ejes paralelos.

Se fabrican a partir de un disco cilíndrico cortado de una plancha de un trozo de barra maciza redonda. Este disco se lleva al proceso de fresado, en donde se retira material para formar los dientes. La fabricación de estos engranajes es más simple, por lo tanto reduce sus costos.

- Engranajes cilíndricos de dientes rectos.

Los engranajes cilíndricos rectos son el tipo de engranaje más simple y corriente que existe. Se utilizan generalmente para velocidades pequeñas y medias; a grandes velocidades, si no son rectificados, o ha sido corregido su tallado, producen ruido cuyo nivel depende de la velocidad de giro que tengan. *Referencia[1]*.

Figura 6. Engranaje recto

- Engranajes cilíndricos de dientes helicoidales.

Los engranajes cilíndricos de dentado helicoidal están caracterizados por su dentado oblicuo con relación al eje de rotación. En estos engranajes el movimiento se transmite de modo igual que en los cilíndricos de dentado recto, pero con mayores ventajas. Los ejes de los engranajes helicoidales pueden ser paralelos o cruzarse, generalmente a 90º. Para eliminar el empuje axial el dentado puede hacerse doble helicoidal.

Los engranajes helicoidales tienen la ventaja que transmiten más potencia que los rectos, y también pueden transmitir más velocidad, son más silenciosos y más duraderos; además, pueden transmitir el movimiento de ejes que se corten. De sus inconvenientes se puede decir que se desgastan más que los rectos, son más caros de fabricar y necesitan generalmente más engrase que los rectos.

Lo más característico de un engranaje cilíndrico helicoidal es la hélice que forma, siendo considerada la hélice como el avance de una vuelta completa del diámetro primitivo del engranaje. De esta hélice deriva el ángulo β que forma el dentado con el eje axial. Este ángulo tiene que ser igual para las dos ruedas que engranan pero de orientación contraria, o sea: uno a derechas y el otro a izquierda. Su valor se establece a priori de acuerdo con la velocidad que tenga la transmisión, los datos orientativos de este ángulo son los siguientes:

Velocidad lenta: β = (5º - 10º)

Velocidad normal: β = (15º - 25º)

Velocidad elevada: β = 30º. *Referencia[1].*

Figura 7. Engranaje helicoidal

- Engranajes cilíndricos doble helicoidales.

Este tipo de engranajes fueron inventados por el fabricante de automóviles francés André Citroën, y el objetivo que consiguen es eliminar el empuje axial que

tienen los engranajes helicoidales simples. Los dientes de los dos engranajes forman una especie de V.

Los engranajes dobles son una combinación de hélice derecha e izquierda. El empuje axial que absorben los apoyos o cojinetes de los engranajes helicoidales es una desventaja de ellos y ésta se elimina por la reacción del empuje igual y opuesto de una rama simétrica de un engrane helicoidal doble.

Un engrane de doble hélice sufre únicamente la mitad del error de deslizamiento que el de una sola hélice o del engranaje recto. Toda discusión relacionada con los engranes helicoidales sencillos (de ejes paralelos) es aplicable a los engranajes helicoidales dobles, exceptuando que el ángulo de la hélice es generalmente mayor para los helicoidales dobles, puesto que no hay empuje axial.

Figura 8. Engranaje doble helicoidal

Con el método inicial de fabricación, los engranajes dobles, conocidos como engranajes de espina, tenían un canal central para separar los dientes opuestos, lo que facilitaba su mecanizado. El desarrollo de las máquinas talladoras mortajadoras por generación, tipo Sykes, hace posible tener dientes continuos, sin el hueco central. Como curiosidad, la empresa Citroën ha adaptado en su logotipo la huella que produce la rodadura de los engranajes helicoidales dobles. *Referencia[1]*.

1.3.2 Engranajes de ejes perpendiculares.

Los engranajes cónicos tienen forma de tronco de cono y permiten transmitir movimiento entre ejes que se cortan. Sus datos de cálculo se encuentran en prontuarios específicos de mecanizado.

- Engranajes cónicos rectos.

Efectúan la transmisión de movimiento de ejes que se cortan en un mismo plano, generalmente en ángulo recto aunque no es el único ángulo pues puede variar dicho ángulo como por ejemplo 45, 60, 70, etc., por medio de superficies cónicas dentadas. Los dientes convergen en el punto de intersección de los ejes. Son utilizados para efectuar reducción de velocidad con ejes en 90°. Estos engranajes generan más ruido que los engranajes cónicos helicoidales. En la actualidad se usan muy poco. *Referencia[1]*.

- Engranaje cónicos helicoidales.

Se utilizan para reducir la velocidad en un eje de 90°. La diferencia con el cónico recto es que posee una mayor superficie de contacto. Es de un funcionamiento relativamente silencioso. Además pueden transmitir el movimiento de ejes que se corten. Los datos constructivos de estos engranajes se encuentran en prontuarios técnicos de mecanizado. Se mecanizan en fresadoras especiales, en la actualidad se utilizan en las transmisiones posteriores de camiones y automóviles. *Referencia[1]*.

Figura 9 : Engranaje cónico

- Engranaje cónico hipoide. Un engranaje hipoide es un grupo de engranajes cónicos helicoidales formados por un piñón reductor de pocos dientes y una rueda de muchos dientes, que se instala principalmente en los vehículos industriales que tienen la tracción en los ejes traseros. Tiene la ventaja

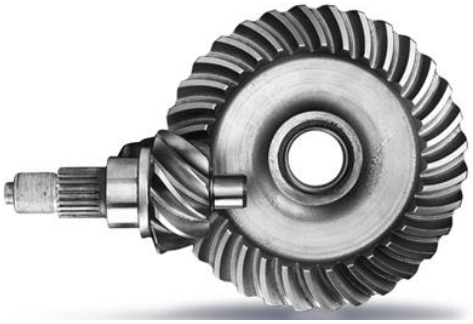

Figura 10. Engranaje cónico hipoide

de ser muy adecuado para las carrocerías de tipo bajo, ganando así mucha estabilidad el vehículo. Por otra parte la disposición helicoidal del dentado permite un mayor contacto de los dientes del piñón con los de la corona, obteniéndose mayor robustez en la transmisión. Su mecanizado es muy complicado y se utilizan para ello máquinas talladoras especiales (Gleason). *Referencia[1]*.

1.3.3 Tornillo sin fin y corona.

Es un mecanismo diseñado para transmitir grandes esfuerzos, que también se utiliza como reductor de velocidad aumentando el torque en la transmisión. Generalmente trabaja en ejes que se cruzan a 90º.

Tiene la desventaja de que su sentido de giro no es reversible, sobre todo en grandes relaciones de transmisión, y de consumir en rozamiento una parte importante de la potencia. En las construcciones de mayor calidad la corona está fabricada de bronce y el tornillo sin fin, de acero templado con el fin de reducir el rozamiento. Si este mecanismo transmite grandes esfuerzos es necesario que esté muy bien lubricado para matizar los desgastes por fricción.

El número de entradas de un tornillo sin fin suele ser de una a ocho. Los datos de cálculo de estos engranajes están en prontuarios de mecanizado.

El tornillo sin fin puede mecanizarse mediante tornos, fresas bicónicas o fresas centrales. La corona, por su parte, requiere fresas normales o fresas madre. *Referencia[1]*.

Figura 11. Tornillo sin fin y corona

- Tornillos sin fin y corona glóbicos.

Normalmente el contacto entre los dientes del tornillo sin fin y los de la corona ocurre en un solo punto, es decir, en una superficie muy reducida de metal. Por tanto, cuando la fuerza a transmitir es elevada se genera una fuerte presión en el punto de contacto. Para reducir la presión se puede aumentar la superficie de contacto entre el tornillo sin fin y la corona, aplicando una de las tres formas siguientes de acoplamiento:

1. corona glóbica y tornillo sin fin convencional
2. tornillo sin fin glóbico y corona convencional
3. tornillo sin fin glóbico y corona también glóbica

Para el mecanizado de tornillos sin fin glóbicos se utiliza el procedimiento de generación que tienen las máquinas Fellows. *Referencia[1]*.

Figura 12. Tornillo sin fin y corona glóbico

1.3.4 Engranajes para aplicaciones especiales.

Se distinguen principalmente los engranajes epicicloidales, , engranajes interiores y de cremallera.

- Engranajes epicicloidales.

También llamados engranajes planetarios, son un tipo de engranajes que pertenecen a los que transmiten el movimiento. Dentro de este grupo los podemos clasificar como ruedas dentadas directas montadas en ejes paralelos.

Los engranajes epicicloidales se pueden definir como aquellos en los se pueden realizar múltiples cambios en la transmisión de movimiento, es decir, podemos cambiar la forma de giro alterando una de las partes del mecanismo.

Se componen de una corona dentada interiormente, que contiene un piñón central llamado planetario y otros tres piñones más pequeños situados entre el planetario y la corona que se denominan satélites. Estos satélites están unidos a su vez al portasatélites. Con esta disposición se pueden realizar múltiples movimientos. Podemos hacer que gire todo el conjunto al mismo tiempo, o también podemos matener fijo el planetario provocando que los satélites giren a su alrededor y que también gire la corona.

Este tipo de engranajes presenta muchas ventajas porque permite realizar fácilmente muchos cambios de movimientos y por eso mismo se suele utilizar en las cajas de cambio de los coches automáticos. *Referencia[1]*.

Figura 13. Engranaje planetario

- Engranajes interiores.

Los engranajes interiores o anulares son variaciones del engranaje recto en los que los dientes están tallados en la parte interior de un anillo o de una rueda con reborde, en vez de en el exterior. Los engranajes interiores suelen ser impulsados por un piñón, un engranaje pequeño con pocos dientes. La cremallera (barra dentada plana que avanza en línea recta) funciona como una rueda dentada de radio infinito y puede emplearse para transformar el giro de un piñón en movimiento alternativo, o viceversa. Dichos engranajes mantienen la velocidad angular constante. *Referencia[1].*

Figura 14. Engranajes interiores

- Engranajes de cremallera.

El mecanismo de cremallera aplicado a los engranajes lo constituyen una barra con dientes la cual es considerada como un engranaje de diámetro infinito y un engranaje de diente recto de menor diámetro, y sirve para transformar un movimiento de rotación del piñón en un movimiento lineal de la cremallera. Quizás la cremallera más conocida sea la que equipan los tornos para el desplazamiento del carro longitudinal. Referencia[1].

Figura 15. Engranajes de cremalleras

1.3.5 Otros tipos de engranajes

- Mecanismo piñón cadena.

El mecanismo piñón cadena es un método de transmisión muy utilizado para transmitir un movimiento giratorio entre dos ejes paralelos que estén bastante separados. Es el mecanismo de transmisión que utilizan las bicicletas, motos y muchas máquinas e instalaciones industriales. También se emplea en sustitución de los reductores de velocidad por poleas cuando es importante evitar el deslizamiento entre la rueda conductora y el mecanismo de transmisión (en este caso una cadena).

Figura 16. Mecanismo piñon y cremallera

Este mecanismo se compone de tres elementos: dos piñones, uno en cada uno de los ejes, y una cadena cerrada. Los dientes de los piñones engranan de manera muy precisa en los eslabones de la cadena, transmitiéndose así el movimiento.

Comparado con el sistema correa-polea, el mecanismo piñón-cadena presenta la ventaja de poder transmitir grandes potencias con un buen rendimiento energético si bien es más ruidoso y necesita lubricantes. *Referencia[1].*

- Poleas dentadas.

Para la transmisión entre dos ejes que estén separados a una distancia donde no sea económico o técnicamente imposible montar una transmisión por engranajes se recurre a un montaje con poleas dentadas que mantienen las mismas propiedades que los engranajes es decir, que evitan el patinamiento y mantienen exactitud en la relación de transmisión.

Los datos más importantes de las poleas dentadas son: número de dientes, paso, y ancho de la polea.

El paso es la distancia entre los centros de las ranuras y se mide en el círculo de paso de la polea. El círculo de paso de la polea dentada coincide con la línea de paso de la banda correspondiente.

Las poleas dentadas se fabrican en diversos materiales tales como aluminio, acero y fundición.*Referencia[1].*

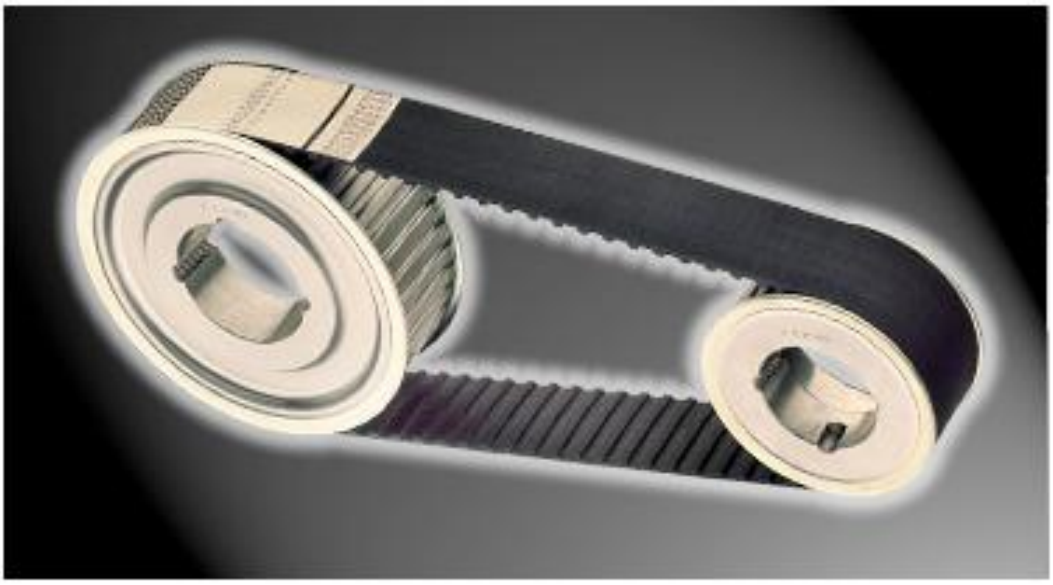

Figura 17. Polea dentada

1.4 Aplicaciones de los engranajes.

Existe una gran variedad de formas y tamaños de engranajes, desde los más pequeños usados en relojería e instrumentos científicos (se alcanza el módulo 0,05) a los de grandes dimensiones, empleados, por ejemplo, en las reducciones de velocidad de las turbinas de vapor de los buques, en el accionamiento de los hornos y molinos de las fábricas de cemento, etc.

El campo de aplicación de los engranajes es prácticamente ilimitado. Los encontramos en las centrales de producción de energía eléctrica, hidroeléctrica y en los elementos de transporte terrestre: locomotoras, automotores, camiones, automóviles, transporte marítimo en buques de todas clases, aviones, en la industria siderúrgica: laminadores, transportadores, minas y astilleros, fábricas de cemento, grúas, montacargas, máquinas-herramientas, maquinaria textil, de alimentación, de vestir y calzar, industria química y farmacéutica, etc., hasta los más simples movimientos de accionamiento manual.

Toda esta gran variedad de aplicaciones del engranaje puede decirse que tiene por única finalidad la transmisión de la rotación o giro de un eje a otro distinto, reduciendo o aumentando la velocidad del primero.

Incluso, algunos engranes coloridos y hechos de plástico son usados en algunos juguetes educativos. *Referencia[1]*.

1.4.1 Bombas hidráulicas.

Una bomba hidráulica es un dispositivo tal que recibiendo energía mecánica de una fuente exterior la transforma en una energía de presión transmisible de un lugar a otro de un sistema hidráulico a través de un líquido cuyas moléculas estén sometidas precisamente a esa presión. Las bombas hidráulicas son los elementos encargados de impulsar el aceite o líquido hidráulico, transformando la energía mecánica rotatoria en energía hidráulica.

Hay un tipo de bomba hidráulica que lleva en su interior un par de engranajes de igual número de dientes que al girar provocan que se produzca el trasiego de aceites u otros líquidos. Una bomba hidráulica la equipan todas las máquinas que tengan circuitos hidráulicos y todos los motores térmicos para lubricar sus piezas móviles. *Referencia[1]*

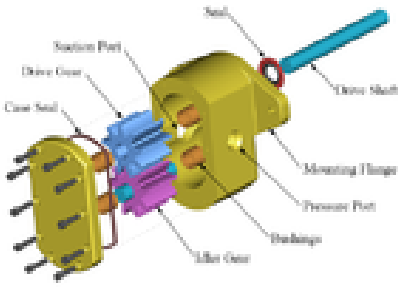

Figura 18. Bomba hidráulica

1.4.2 Mecanismo diferencial.

El mecanismo diferencial tiene por objeto permitir que cuando el vehículo dé una curva sus ruedas propulsoras puedan describir sus respectivas trayectorias sin patinar sobre el suelo. La necesidad de este dispositivo se explica por el hecho de que al dar una curva el coche, las ruedas interiores a la misma recorren un espacio menor que las situadas en el lado exterior, puesto que las primeras describen una circunferencia de menor radio que las segundas.

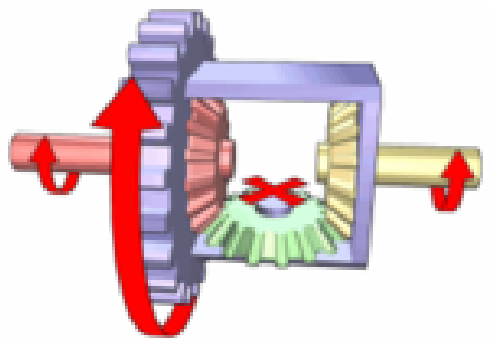

Figura 19. Mecanismo diferencial

El mecanismo diferencial está constituido por una serie de engranajes dispuestos de tal forma que permite a las dos ruedas motrices de los vehículos girar a velocidad distinta cuando circulan por una curva. Así si el vehículo toma una curva a la derecha, las ruedas interiores giran más despacio que las exteriores, y los satélites encuentran mayor dificultad en mover los planetarios de los semiejes de la derecha porque empiezan a rotar alrededor de su eje haciendo girar los planetarios de la izquierda a una velocidad ligeramente superior. De esta forma provocan una rotación más rápida del semieje y de la rueda motriz izquierda.

El mecanismo diferencial está constituido por dos piñones cónicos llamados planetarios, unidos a extremos de los palieres de las ruedas y otros dos piñones cónicos llamados satélites montados en los extremos de sus ejes porta satélites y que se engranan con los planetarios.

Una variante del diferencial convencional está constituida por el diferencial autoblocante que se instala opcionalmente en los vehículos todo-terreno para viajar sobre hielo o nieve o para tomar las curvas a gran velocidad en caso de los automóviles de competición. *Referencia[1]*.

1.4.3 Cajas de velocidades.

En los vehículos, la caja de cambios o caja de velocidades es el elemento encargado de acoplar el motor y el sistema de transmisión con diferentes relaciones de engranes o engranajes, de tal forma que la misma velocidad de giro del cigüeñal puede convertirse en distintas velocidades de giro en las ruedas.

Figura 20. Caja de velocidad

El resultado en la ruedas de tracción generalmente es la reducción de velocidad de giro e incremento del torque.

Los dientes de los engranajes de las cajas de cambio son helicoidales y sus bordes están redondeados para no producir ruido o rechazo cuando se cambia de velocidad. La fabricación de los dientes de los engranajes es muy cuidada para que sean de gran duración. Los ejes del cambio están soportados por rodamientos de bolas y todo el mecanismo está sumergido en aceite denso para mantenerse continuamente lubricado.

Referencia[1].

1.4.4 Reductores de velocidad.

Los reductores de velocidad son mecanismos que transmiten movimiento entre un eje que rota a alta velocidad, generalmente un motor, y otro que rota a menor velocidad, por ejemplo una herramienta. Se componen de juegos de engranajes de diámetros diferentes o bien de un tornillo sin fin y corona.

Figura 21. Reductor de velocidad

El reductor básico está formado por mecanismo de tornillo sin fin y corona. En este tipo de mecanismo el efecto del rozamiento en los flancos del diente hace que estos engranajes tengan los rendimientos más bajos de todas las transmisiones; dicho rendimiento se sitúa entre un 40 y un 90% aproximadamente, dependiendo de las características del reductor y del trabajo al que está sometido. Factores que elevan el rendimiento:

- Ángulos de avance elevados en el tornillo.
- Rozamiento bajo (buena lubricación) del equipo.
- Potencia transmitida elevada.
- Relación de transmisión baja (factor más determinante).

Existen otras disposiciones para los engranajes en los reductores de velocidad, estas se denominan conforme a la disposición del eje de salida (eje lento) en comparación con el eje de entrada (eje rápido). Así pues serían los llamados reductores de velocidad de engranajes coaxiales, paralelos, ortogonales y mixtos (paralelos + sin fin corona). En los trenes coaxiales, paralelos y ortogonales se considera un rendimiento

aproximado del 97-98%, en los mixtos se estima entre un 70% y un 90% de rendimiento.

Además, existen los llamados reductores de velocidad de disposición epicicloidal, técnicamente son de ejes coaxiales y se distinguen por su formato compacto, alta capacidad de trasmisión de par y su extrema sensibilidad a la temperatura.

Las cajas reductoras suelen fabricarse en fundición gris dotándola de retenes para que no salga el aceite del interior de la caja. *Referencia[1]*

1.5 Descripción de la geometría de los engranajes que se van a realizar.

1.5.1 Introducción a los engranajes rectos de involuta.

Al considerar dos superficies curvas en contacto directo, se ha demostrado que la relación de las velocidades angulares es inversamente proporcional a los segmentos en que es cortada la línea de centros por la línea de acción o normal común a las dos superficies en contacto. Si la línea de acción siempre interseca la línea de centros en un punto fijo, entonces la relación de las velocidades angulares permanece constante. Esta es la condición que se desea cuando se acoplan dos dientes de engranajes: la relación de velocidades debe ser constante. Es posible suponer la forma del diente en uno de los engranajes y, aplicando el principio anterior (la normal común interseca la línea de centros en un punto fijo, determinar el perfil del diente que se acopla. Dichos dientes se consideran conjugados y las posibilidades solamente están limitadas por la habilidad personal para formar los dientes. De las muchas formas posibles solamente se han estandarizado la cicloide y la involuta. La cicloide se empleó inicialmente, aunque ahora se ha reemplazado por la involuta en todas las aplicaciones salvo en los relojes de pulso y pared. La involuta tiene varias ventajas, siendo las más importantes su facilidad de fabricación y el hecho de que la distancia entre los centros de dos engranajes de involuta puede variar sin cambiar la relación de velocidades.

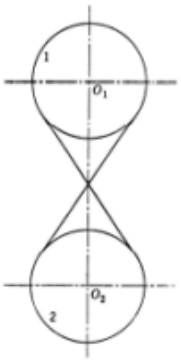

Figura 22. Poleas de cable cruzado

Como sucede en el caso de las poleas de cable cruzado (figura 22), la relación de las velocidades angulares es inversamente proporcional a los diámetros de las ruedas. Si se cambia la distancia entre centros, la involuta 1 seguirá moviendo a la involuta 2, aunque ahora estarán en contacto diferentes porciones de las dos involutas. En tanto no se cambien los diámetros de las ruedas, la relación de velocidades seguirá siendo la misma.

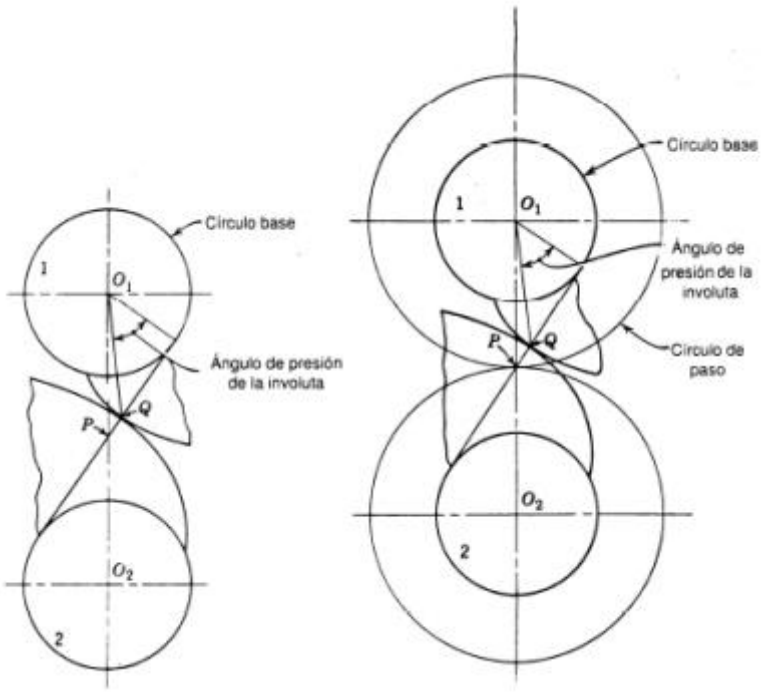

Figura 23. Contacto entre involutas.

Los círculos empleados como base para generar las involutas se conocen como círculos base, y son el corazón del sistema de engranajes de involuta. En la figura anterior, el ángulo comprendido entre una línea perpendicular a la línea de acción que pasa por el centro del circulo base y una línea desde O1 a Q (o desde O2 a Q), se conoce como ángulo de presión de la involuta y es una dimensión del punto en la involuta en donde está ocurriendo el contacto. Si se marca el punto P como punto de intersección de la línea de acción y la línea de centros, la relación de las velocidades angulares será inversamente proporcional a los segmentos en que este segmento divide la línea de centros.

Cabe la posibilidad de dibujar círculos por el punto P usando primero O1como centro y luego O2, como se muestra en la figura 2 anterior. El punto P se conoce como punto de paso y los círculos que pasan por este punto se conocen como círculos de paso. Se puede demostrar que cuando la involuta1 mueve a la involuta 2, los dos círculos de paso de mueven juntos con la acción de rodamiento puro. Debido a que los segmentos en que el punto p divide a la línea de centros son ahora los radios de paso, la relación de las velocidades angulares es inversamente proporcional a los radios de los

dos círculos de paso. Si el diámetro del círculo de paso 1 es D1 y el del círculo 2 es D2, entonces:

$$\frac{w1}{w2} = \frac{D2}{D1} \quad (1.1)$$

En una sección posterior se demostrará que el número de dientes en un engrane es directamente proporcional al diámetro de paso, por lo que:

$$\frac{w1}{w2} = \frac{D2}{D1} = \frac{N2}{N1} \quad (1.2)$$

1.5.2 Involumetría.

Al considerar la involuta para la forma de un diente, es necesario poder calcular determinadas propiedades de la involuta.

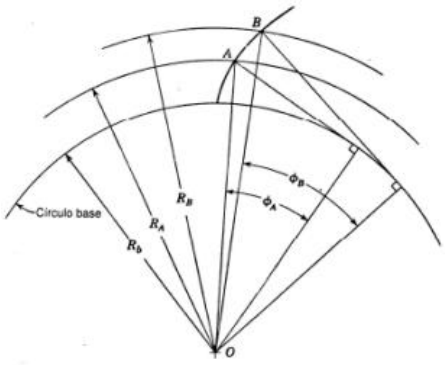

Figura 24. Radios característicos

La figura 24 muestra una involuta que se generó a partir de un círculo base de radio R_b.

La involuta contiene dos puntos, A y B, con radios correspondientes R_a, R_b y ángulos depresión φ_a, φ_b. Se puede por lo tanto establecer una relación para los factores anteriores debido a que el radio del circulo base permanece constante sin importar el punto que se esté considerando.

Por lo tanto,

$$R_b = R_A \cos \varphi_A \quad (2.1)$$

O bien

$$R_b = R_B \cos \varphi_B \quad (2.2)$$

De manera que igualando ambas ecuaciones:

$$\cos \varphi_B = \frac{R_A}{R_B} \cos \varphi_A \, . \, (2.3)$$

A partir de esta última ecuación es posible determinar el ángulo de presión de la involuta en cualquier punto de radio conocido en la involuta.

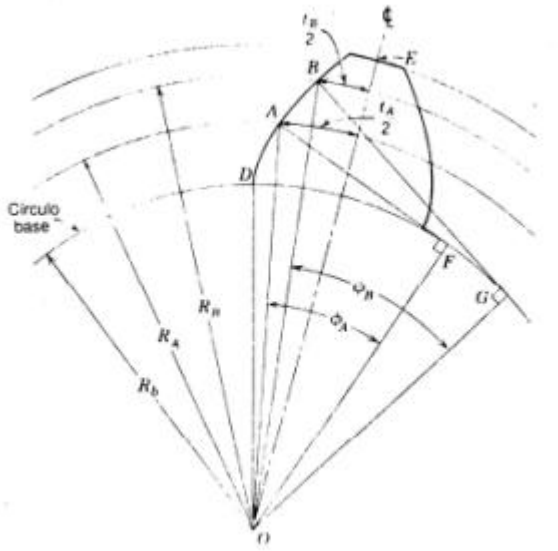

Figura 25. Geometría del diente

A partir de la ilustración es posible desarrollar una ecuación para encontrar el espesor del diente en cualquier punto B, dado el espesor en el punto A.

Por el principio de la generación de una involuta, el arco DG es igual a la longitud BG, por lo que:

$$\llcorner DOG = \frac{arcDG}{OG} = \frac{BG}{OG} = tan\varphi_B (3.1)$$

De manera que,

$$\llcorner DOB = \llcorner DOG - \varphi_B = tan\varphi_B - \varphi_B \; (3.2)$$

Y pudiéndose demostrar que

$$\llcorner DOA = tan\varphi_A - \varphi_A \; (3.3)$$

La expresión $tan\varphi - \varphi$ se denomina función de involuta y a veces se escribe como inv φ. Es fácil calcular la función involuta cuando el ángulo es conocido, donde φ se expresa en radianes. Sin embargo es difícil convertir de inv φ a φ.

Observando de nuevo la figura 25, podemos concluir también que:

$$\angle DOE = \angle DOB + \frac{\frac{1}{2}tB}{RB} = inv\ \varphi_B + \frac{tB}{2RB} (3.4)$$

$$\angle DOE = \angle DOA + \frac{\frac{1}{2}tA}{RA} = inv\ \varphi_A + \frac{tA}{2RA}\ (3.5)$$

Y de las relaciones anteriores,

$$t_B = 2R\left[\frac{\frac{1}{2}tA}{RA} + inv\varphi_A - inv\varphi_B\right]\ (3.6),$$

se obtiene el espesor del diente en cualquier punto de la involuta, si se conoce el espesor en cualquier otro punto.

Sin embargo, en este proyecto se va a usar principalmente la ecuación paramétrica de la involuta en un sistema de referencia cartesiano:

$$x = rb(cost + tsent)\ (4.1)$$

$$y = rb(sent - tcost)\ (4.2)$$

Donde:

x=coordenada en abcisa del punto de la involuta en el que nos encontramos.
y= coordenada en ordenada del punto de la involuta en el que nos encontremos.
rb=radio de la circunferencia base.
t=parámetro, en este caso es el ángulo polar en radianes.

Referencia[2].

1.5.3 Características geométricas de los engranajes rectos.

A continuación se van a estudiar los elementos básicos de un engrane como el que se muestra en la figura 26.

El círculo de paso es un círculo teórico en el que por lo general se basan todos los cálculos; su diámetro es el diámetro de paso. Los círculos de paso de un par de engranajes acoplados son tangentes entre sí. Un piñón es el menor de dos engranajes acoplados, el mayor a menudo se llama rueda.

El paso circular p es la distancia, medida sobre el círculo de paso, desde un punto en un diente a un punto correspondiente en un diente adyacente. De esta manera, el paso circular es igual a la suma del espesor del diente y del ancho del espacio.

El módulo m representa la relación del diámetro de paso con el número de dientes. La unidad de longitud que suele emplearse es el milímetro. El módulo señala el índice del tamaño de los dientes en unidades SI.

El paso diametral P está dado por la relación del número de dientes en el engrane al diámetro de paso. En consecuencia, es el recíproco del módulo. Ya que el paso diametral se da sólo en unidades del sistema inglés, se expresa en dientes por pulgada.

La cabeza a se determina por la distancia radial entre la cresta y el círculo de paso. La raíz b equivale a la distancia radial entre la cresta y el círculo de paso. La altura total h, es la suma de la cabeza y la raíz.

El círculo de tolerancia es un círculo tangente al círculo de la raíz del engrane acoplado. La tolerancia c está dada por la cantidad por la que la raíz en un engrane dado excede la cabeza de su engrane acoplado. La holgura se determina mediante la cantidad por la cual el ancho del espacio de un diente excede el espesor del diente de acoplamiento medido en los círculos de paso.

De manera, que se pueden demostrar las siguientes expresiones:

$$P = \frac{N}{d},$$

donde P= paso diametral [dientes/pulgada], N= número de dientes y d=diámetro de paso[in].

También

$$m = \frac{d}{N},$$

con m=módulo [mm] y d=diámetro de paso[mm].

Y finalmente se obtiene

$$p = \pi m = \frac{\pi d}{N}, con\ p = paso\ circular\ y\ obteniendo\ pP = \pi.$$

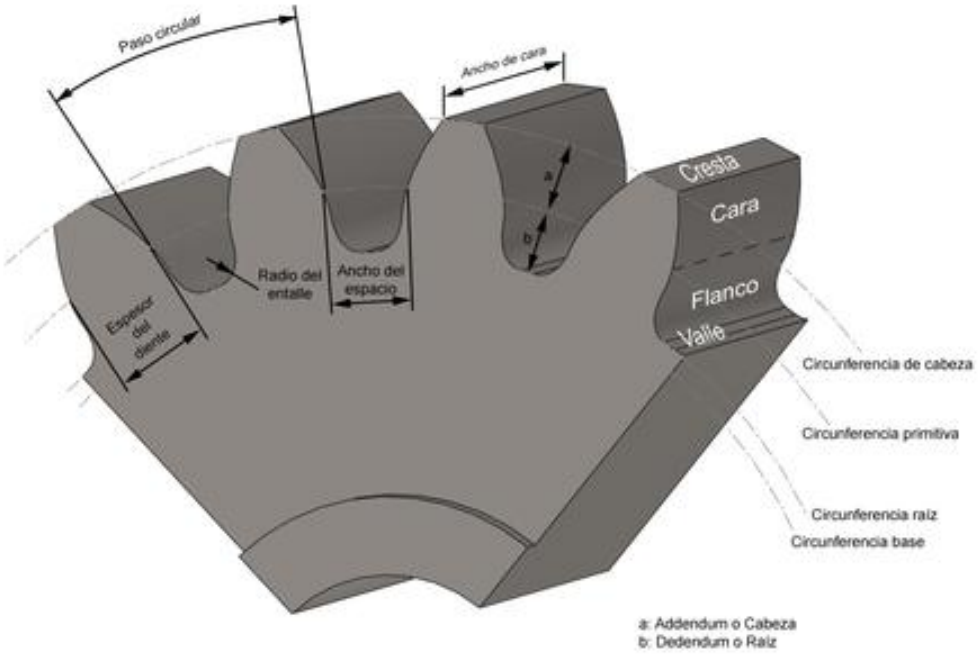

Figura 26. Geometría del engranaje

1.5.4 Sistemas de dientes.

Un sistema de dientes es una norma que especifica las relaciones que implican la cabeza, raíz, profundidad de trabajo, espesor del diente y ángulo de presión. Al principio, las normas se plantearon para posibilitar el intercambio de engranes con cualquier número de dientes, pero con el mismo ángulo de presión y paso.

La tabla 13-1 contiene las normas más empleadas para engranes rectos. En algún tiempo se usó un ángulo de presión de $14\frac{1}{2}^{\circ}$ para estos engranes, pero en la actualidad es obsoleto, ya que los engranes resultantes tenían que ser comparativamente mayores para evitar problemas de interferencia.

La tabla 13-2 es muy útil para la selección del paso o del módulo de un engrane, ya que por lo general se dispone de cortadoras para los mismos tamaños proporcionados en la tabla.

Tabla 1. Sistema de dientes (engranaje recto)

Sistema de dientes	Ángulo de presión[°]	Cabeza a	Raíz b
Profundidad	20	$1/P_d$ o	$1.25/P_d$ o

total		bien 1m	bien 1.25m. $1.35/P_d$ o bien 1.35m
	$20\frac{1}{2}$	$1/P_d$ o bien 1m	$1.25/P_d$ o bien 1.25m. $1.35/P_d$ o bien 1.35m
	25	$1/P_d$ o bien 1m	$1.25/P_d$ o bien 1.25m. $1.35/P_d$ o bien 1.35m
Profundidad parcial(dientes cortos)	20	$0.8/P_d$ o bien 0.8m	$1/P_d$ o bien 1m

Tabla 2. Números preferidos

Paso diametral	
Basto	*2,2$\frac{1}{4}$,2$\frac{1}{2}$,3,4,6,8,10,12,16*
Fino	20,24,32,40,48,64,72,80,96,120,150,200.
Módulo	
Preferidos	*1,1.25,1.5,1.75,2,2.25,2.5,2.75,3,4,5,6,8,10,12,16,20,25,32,40,50*
Siguiente elección	1,1.125,1.375,1.75,2.25,2.75,3.5,4.5,5.5,7,9,11,14,18,22,28,36,45

Referencia[3]

1.5.5 Introducción a la teoría de los engranajes helicoidales.

Si se hace girar un plano en un cilindro base, una línea en el plano paralelo al eje del cilindro generará la superficie del diente de un engranaje recto de involuta (Figura 27.a). Sin embargo, si la línea generatriz se inclina hacia el eje, entonces se generará la superficie del diente de un engranaje helicoidal (Figura 27.b).

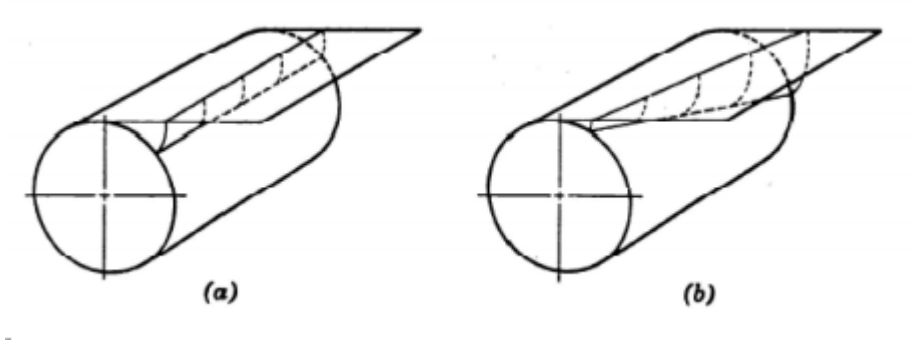

Figura 27. Generatrices en engranajes rectos y helicoidales.

Para determinar las proporciones de los dientes de un engrane helicoidal es necesario considerar la manera en que los dientes se van a cortar. Si el engrane se va a fresar, todas las dimensiones de calculan en un plano normal al elemento de paso del diente, y el paso diametral y el ángulo de presión son valores estándares en dicho plano. Debido a que la acción de corte de un fresa ocurre en un plano normal, es posible utilizar la misma fresa para cortar tanto engranajes helicoidales como engranajes rectos de un paso dado; en un engranaje recto el plano normal y el plano de rotación son idénticos.

En este proyecto se va a considerar que los engranajes se cortan mediante el método Fellows para formado de engranajes, por lo que se podrán usar las ecuaciones estudiadas para engranajes rectos. Para este método no se puede emplear el mismo cortador para cortar tanto engranes helicoidales como rectos.

En el presente proyecto se van a desarrollar los engranajes helicoidales de manera que la única diferencia con los engranajes rectos sea el giro de la sección del diente a lo largo del espesor del engrane. Para ello, dado el ángulo de la hélice se tiene que calcular el giro que presenta el perfil del diente a lo largo del espesor, en un plano normal al eje de rotación del engranaje.

A partir de la figura 28 se puede hallar fácilmente el ángulo theta correspondiente:

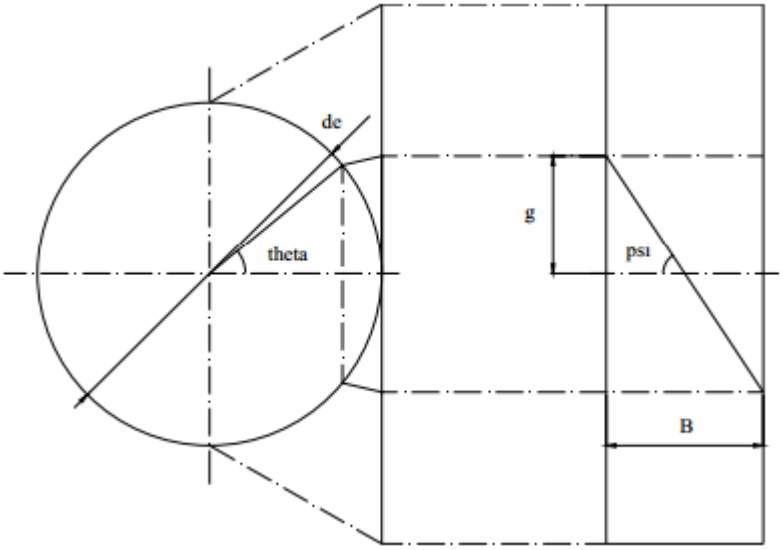

Figura 28. Desarrollo del círculo. Ángulo theta

$$\tan(\psi) = \frac{2g}{B} \quad (4.1)$$

Por lo que despejando g obtenemos:

$$g = \frac{\mathrm{B}}{2}\tan(\psi) \quad (4.2)$$

Por otro lado, considerando que la cuerda del diámetro exterior del engranaje es aproximadamente igual al arco de circunferencia correspondiente:

$$\theta = \frac{2g}{de} \quad (4.3)$$

De manera que obtenemos:

$$\theta = \frac{B}{de}\tan(\psi)\,. \quad (4.4)$$

Referencia[4].

2 El entorno de programación VBA en CATIA V5

En este capítulo se va a presentar el programa CATIA V5 y se va a introducir y explicar a grandes rasgos el lenguaje de programación "Visual Basic for applications"(VBA). Una vez introducido el entorno en el que se va a trabajar, se procederá a explicar el entorno general de trabajo de VB6, así como nociones generales de programación de dicho lenguaje.

2.1 CATIA V5

CATIA (Computer-Aided Three dimensional Interactive Application) es un programa informático de diseño, fabricación e ingeniería asistida por computadora comercial realizado por Dassault Systèmes. El programa está desarrollado para proporcionar apoyo desde la concepción del diseño hasta la producción y el análisis de productos.

Provee una arquitectura abierta para el desarrollo de aplicaciones o para personalizar el programa. Las interfaces de programación de aplicaciones, CAA2 (o CAAV5), se pueden programar en Visual Basic y C++.

Fue inicialmente desarrollado para servir en la industria aeronáutica. Se ha hecho un gran hincapié en el manejo de superficies complejas. CATIA también es ampliamente usado en la industria del automóvil para el diseño y desarrollo de componentes de carrocería. Concretamente empresas como el Grupo VW (Volkswagen, Audi, SEAT y Škoda), BMW, Renault, Peugeot, Daimler AG, Chrysler, Smart y Porsche hacen un amplio uso del programa. La industria de la construcción también ha incorporado el uso del software para desarrollar edificios de gran complejidad formal; el Museo Guggenheim Bilbao, en España, es un hito arquitectónico que ejemplifica el uso de esta tecnología. *Referencia[1].*

2.2 Visual Basic por aplication.

Microsoft VBA (Visual Basic for Applications) es el lenguaje de macros de Microsoft Visual Basic que se utiliza para programar aplicaciones Windows y se incluye en varias aplicaciones Microsoft. VBA permite a usuarios y programadores ampliar la funcionalidad de programas de la suite Microsoft Office. Visual Basic para Aplicaciones es un subconjunto casi completo de Visual Basic 5.0 y 6.0.

Microsoft VBA viene integrado en aplicaciones de Microsoft Office, como Word, Excel, Access y Powerpoint. Prácticamente cualquier cosa que se pueda programar en Visual Basic 5.0 o 6.0 se puede hacer también dentro de un documento de Office, con la sola limitación que el producto final no se puede compilar separadamente del documento, hoja o base de datos en que fue creado; es decir, se convierte en una macro (o más bien súper macro). Esta macro puede instalarse o distribuirse con sólo copiar el documento, presentación o base de datos.

Su utilidad principal es automatizar tareas cotidianas, así como crear aplicaciones y servicios de bases de datos para el escritorio. Permite acceder a las funcionalidades de un lenguaje orientado a eventos con acceso a la API de Windows.

Al provenir de un lenguaje basado en Basic tiene similitudes con lenguajes incluidos en otros productos de ofimática como *StarBasic y Openoffice*.

La primera versión de Visual Basic fue presentada en 1991, siendo la última la versión 6, liberada en 1998. *Visual Basic for Applications* (VBA) es el lenguaje de macros de *Visual Basic v6*, incorporado en muchas aplicaciones de Microsoft y posteriormente en otras aplicaciones para ampliar la funcionalidad de las mismas. *VBA* incorpora las librerías y herramientas de *Visual Basic* a las que añade librerías de objetos propias de cada software donde está incluido. La debilidad de este lenguaje radica en que la compilación de la macro no puede realizarse si no se dispone del entorno en el que se ha desarrollado. Otra debilidad es su falta de versatilidad para trabajar en otros sistemas operativos.

CATIA en 1998 con la versión V5 incorporó VBA a su entorno, pudiendo realizar macros en VB y en lenguaje C++, siendo aún los lenguajes de macros que se ha dispuesto para su versión V6. *Referencia[1]*

2.2.1 Entorno

En primer lugar se va a explicar que es una Macro. Una macro consiste en una serie de funciones escritas en un lenguaje de programación que agrupa una serie de comandos los cuales permiten realizar las operaciones requeridas automáticamente, de manera que son una forma de ahorrar tiempo y reducir la posibilidad de errores humanos a la hora de realizar operaciones que se lleve a cabo de forma repetitiva.

El uso de Macros para la automatización en el proceso de diseño es prácticamente ilimitado, siendo algunos ejemplos la Importación de puntos desde Excel a un modelo CAD 3D, generación de geometría de manera automática, creación de planos de módelos 3D, etc.

Como es sabido, CATIA posee la posibilidad de trabajar en el entorno VBA. Ello se puede realizar desde cualquier módulo abriendo la pestaña Tools, donde se observa la opción Macros dentro de la cual se podrá o bien comenzar a grabar una macro, acceder a las macros ya realizadas y librerías o al editor de *visual basic* . *Referencia [5].*

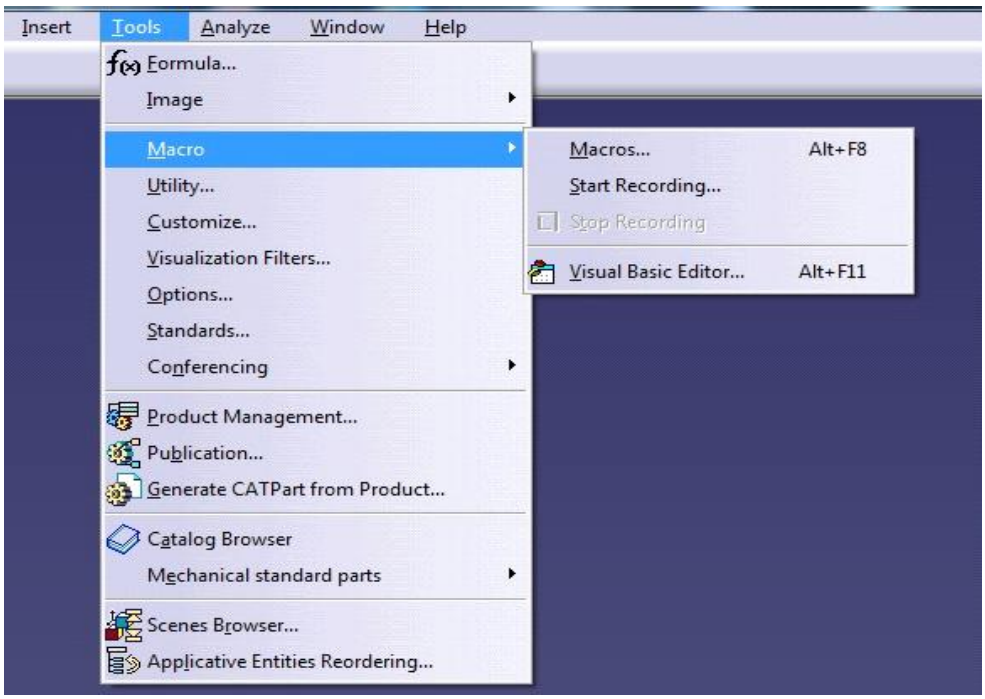

Figura 29. Acceso a macros

2.2.2 Librerías.

Las macros de CATIA son almacenadas en las librerías de macros de tres formas posibles: "*Folders*" (vbscript y CATScript), "*project files*" (catvba), o CATParts/CATProducts. Solo una de esas tres librerías de macros pueden ser usadas al mismo tiempo. Para crear una nueva librería el procedimiento a usar es el explicado anteriormente:

1. Ir a *Tools-->Macro-->Macros*

2. Abrir "Macro libraries"

3. Hay que asegurarse de que el tipo de librerías es cargada en "*Directories*", entonces clicar "*Add existing library*".

4. Seleccionar la carpeta donde se van a guardar los CATScripts a lo largo del proyecto.

5. Cerrar la librería de macros. En dicha librería creada deberían aparecer la lista de CATScripts que se realicen. *Referencia[5]*

2.2.3 Macro recording.

Un método para crear macros es grabando las acciones que se realicen con el ratón. Para macros grabadas en un fichero o en un CATpart o CATproduct, los estamentos declarados se grabaran para CATScript pero no para MSVBscript. Para macros grabadas en una librería .catvba, "MS VBA" es la única opción.

A la hora de grabar macros mediante este procedimiento hay que tener algunas cosas en cuenta:

- No seleccionar *Workbenches* (entornos de trabajo) mientras se está grabando una macro.

- No grabar más de lo que sea absolutamente necesario.

-No usar la opción "deshacer" mientras se esté grabando.

- Ser consciente y darse cuenta de la configuración de CATIA cuando se está grabando.

-Salir de los *sketches* (bocetos) antes de parar de grabar.

-Verificar cada macro una vez se haya grabado.

Una vez se haya finalizado la grabación, se deshará todo lo realizado y se "correrá" la macro, de manera que se podrá comprobar si la macro es correcta y si reproduce la operación que se quería realizar.

También hay que tener en cuenta que mediante dicho procedimiento aparecerán numerosas líneas de código que no son realmente necesarias, por lo que pueden eliminarse.

Por otro lado, tampoco aparecerán comentarios acerca de lo que se está realizando o explicando los parámetros de entrada, por lo que se deberán añadir manualmente. *Referencia[5].*

A lo largo de la realización del proyecto, se ha hecho uso de la grabación de macros solamente para aprender a usar operaciones de gran complejidad, ya que todos los códigos que se han desarrollado en el presente proyecto se han realizado mediante la otra opción de desarrollar macros que presenta CATIA, el editor de *visual basic*.

2.3 Visual Basic editor.

Dicho entorno es el que se va a usar durante la realización de nuestro proyecto. Para acceder al mismo, como se dijo previamente, entramos en *Tools-->Macro-->Visual Basic Editor*, obteniendo la imagen que se muestra a continuación:

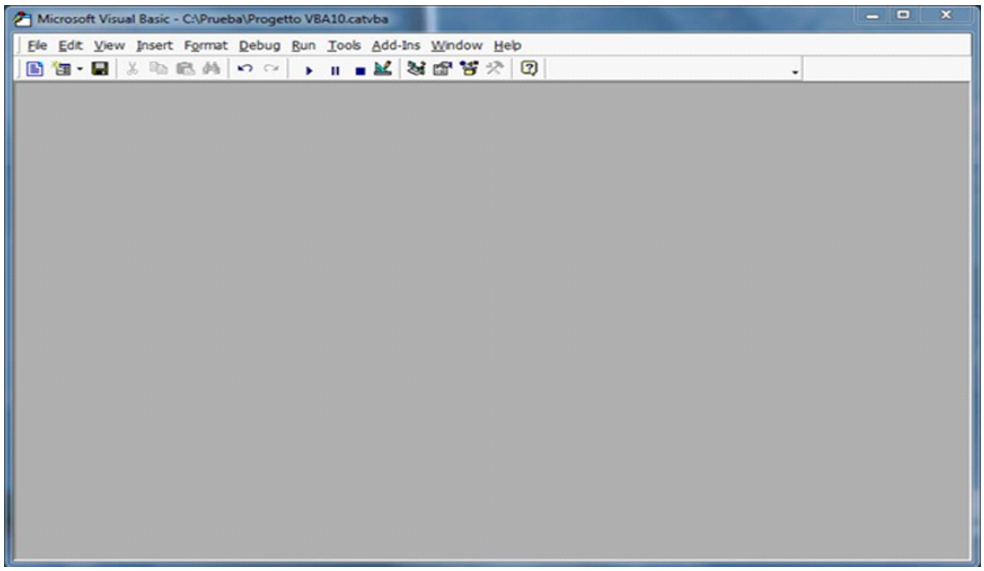

Figura 30. Entorno VBA

Todo lo que se programe dentro de esta ventana puede interactuar directamente con CATIA si empleamos los objetos del programa. A continuación se van a explicar las distintas partes que constituyen el entorno de dicho editor para poder comprender mejor su funcionamiento.

Conviene tener visibles y a disposición del programador el menú denominado *Project Explorer*, así como el *Properties Windows*, ya que son los dos menús fundamentales en los que se trabaja y los cuales facilitan mucho el trabajo. Para acceder a dichos menús, que se muestran en la figura a continuación, es necesario ir a la pestaña View y seleccionar ambas aplicaciones para que sean visibles en la ventana principal. *Referencia[4]*.

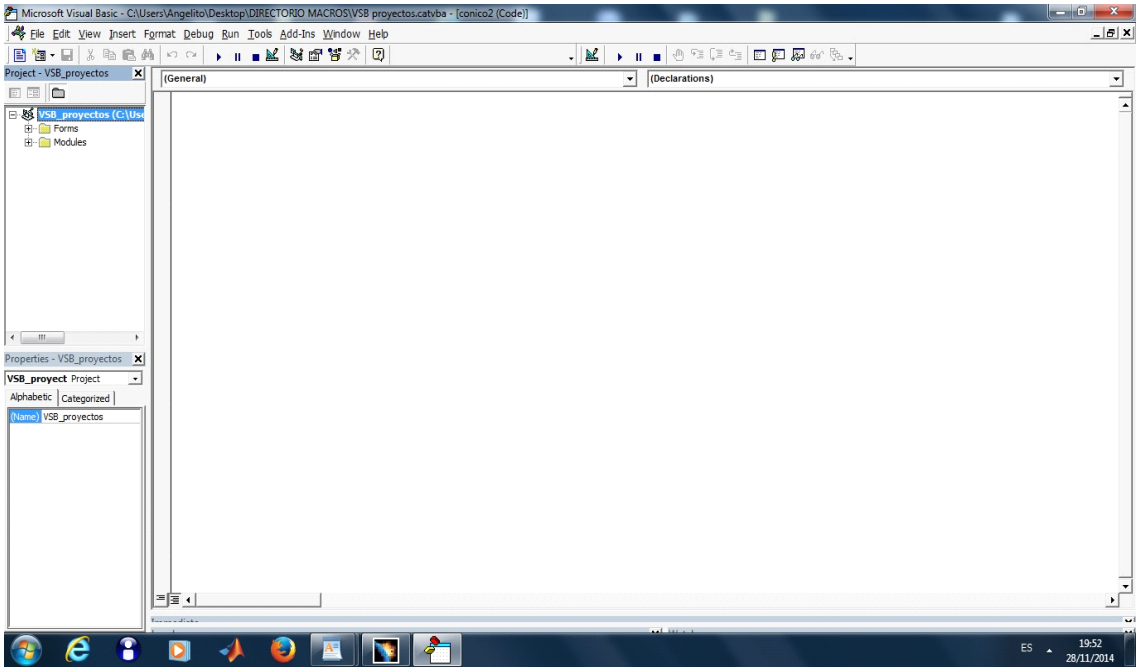

Figura 31. Project explorer y properties windows

La ventana *Project explorer* recoge cada uno de los datos y archivos que conforman el *Project* (proyecto de programación) en el que se va a trabajar. Puede albergar formularios (*Useforms*), módulos (*Modele*) y clases (*class Module*), los cuales se explican a continuación:

- Módulos: Son rutinas independientes. Dado que el código contenido en un módulo estándar de Visual Basic es accesible desde distintos formularios del programa, será ventajoso colocar en este módulo todo lo que queramos disponer como "código compartido", es decir, que pueda ser utilizado en cualquier formulario del programa. Se guardan para su exportación en ficheros con extensión *.bas

- Formularios: Son rutinas asociados a ventanas gráficas donde se incorporan objetos y eventos. Los formularios son el elementos básico que permite la interacción del programa con el usuario, demandando variables, opciones, etc. Se almacenan con extensión *.frm .

- Clases: Son definiciones de nuevos objetos de tipo plantilla donde se definen las propiedades y eventos del mismo, son almacenados como *.cls.

En lo que respecta a la ventana *Properties Window*, refleja todas las propiedades de cada formulario, así como de los controles que conforman los mismos. Para acceder a dicha información solo tenemos que pinchar sobre el formulario o control deseado.

Finalmente vamos a explicar también otra ventana que ha sido de gran utilidad a lo largo de la realización del proyecto, la ventana *object browser*, la cual se acciona de la misma forma que las anteriores.

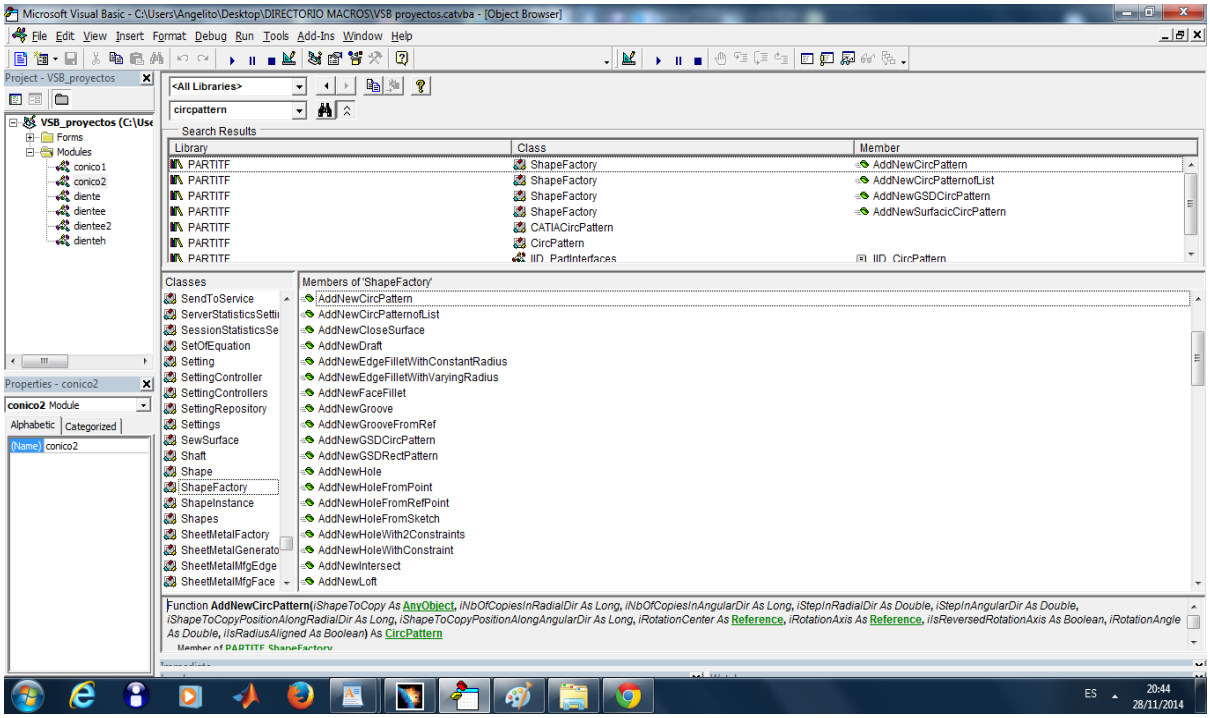

Figura 32. Object browser

Dicha herramienta es la mejor que se puede usar cuando se está atascado en la programación, o no se sabe bien que hacer o como funciona cualquier tipo de objeto. Si se quiere aprender a usar cualquier tipo de objeto, basta con poner el susodicho nombre en la barra de búsqueda, y observar el contenido de las tres columnas que aparecen. La tercera de ellas, la columna *members*, muestra distintas formas para realizar la misma operación, eligiéndose la más conveniente. La segunda columna, muestra a qué tipo de objeto pertenece la operación que se quiere realizar, por lo que ya se sabrá que objeto hay que definir para poder acceder a la herramienta deseada. *Referencia[5]*

Como se puede observar en la parte inferior de la imagen, también se muestra una ayuda, que desglosa uno a uno todos los argumentos de entrada que son necesarios para realizar la operación.

2.4 Iniciación a la programación.

Como es sabido *visual basic* es un lenguaje de programación por eventos, es decir, la ejecución del programa se produce en diferentes secciones debido a respuestas

a eventos. Dichos eventos se producen por la interacción con el usuario a través de una interfaz gráfica, la cual demanda al usuario los parámetros necesarios para el correcto funcionamiento del programa, o la ejecución de cualquier acción.

Por lo tanto, a la hora de programar en dicho lenguaje habrá que poner especial hincapié en realizar una programación precisa y ordenada para que el código responda de manera adecuada a los eventos para los que se ha diseñado la aplicación.

A continuación se adjuntan una serie de conceptos cuyo entendimiento es de vital importancia, ya que en ellos se basa la programación controlada por eventos.

Tabla 3. Conceptos básicos

Concepto	Definición
Tiempo de diseño	Instante en que se crea la aplicación
Tiempo de ejecución	Instante en el cual se ejecuta e si interactúa con la aplicación.
Formulario	Ventana sobre la que es posible personalizar la interfaz de la aplicación o cuadro de diálogo para obtener información del usuario.
Objetos	Formularios y controles.
Controles	Representación gráfica de objetos, con los que el usuario interactúa e aporta la información que se le pide.
Propiedades	Los valores de un objeto (*properties window* antes mencionada).Son características de un objeto, y definen el estado del mismo en un momento específico.
Métodos	Las acciones que un objeto puede realizar sobre sí mismo. Se suelen usar verbos para dar nombre a los métodos.
Eventos	Acciones que son reconocidas por un formulario o control. Los eventos ocurren a medida que el usuario interactúa con los objetos de la aplicación.
Colección	Grupo o lista de objetos similares que se ponen juntos por una razón específica. Las colecciones son objetos que agregan un conjunto de otros objetos.

Clases	Definen un tipo de objeto. Se suele usar la herencia para crear jerarquía entre clases y subclases.
Herencia	Todos aquellas clases que sean herencia de la misma clase tienen todas las propiedades y métodos en común de la herencia de la que provienen, pero también tienen sus propios métodos y propiedades que las diferencian entre ellas.

Referencias [4] y [5].

Otros conceptos importantes a la hora de programas son los que siguen:

2.4.1 Declaración de estamentos.

En *visual basic*, antes de poder trabajar con una determinada variable, constante o herramienta es necesario nombrarla, obteniendo de esta forma una primera información acerca del estamento que posteriormente estableceremos.

Si no definimos el tipo de variable, VBA declarará la variable de tipo *variant*, la cual puede aceptar cualquier tipo de variable. Cabe destacar que en muy raras ocasiones se tiene una buena razón para usar una variable tipo *variant*. En la mayoría de ocasiones tendremos que usar un determinado tipo de variable, haciendo de esta manera que el código se ejecute más rápido y reduciendo errores, lo cual se debe a que:

1. CATIA ejecutará el tipo de variable que se especifique, y no la que crea conveniente (que sería lo que ocurriría en el caso de variable tipo *variant*.

2. A la hora de revisar el código siempre se sabrá de que tipo es cada variable, y la intención por la cual se creó.

El comando que se usará para definir el tipo de variable con el que trabajaremos es *Dim*, como se muestra en el siguiente ejemplo:

Dim documents1 as documents. *Referencia[5]*

2.4.2 Estamentos.

Instrucción completa que puede contener operadores (+,-,*,/,%,...), variables, constantes, expresiones, y *keywords*(*And, if, for, while, sub, function*...). Un ejemplo de estamento sería:

Set documents1=CATIA.documents

El comando *Set* se usa por lo tanto para asociar la variable definida con un objeto. *Referencia[4]*

2.4.3 Estamentos ejecutables

Son acciones iniciales como:

Select1.Search("name='Optimization.MinimumValue',all"). Referencia[4]

2.4.4 Funciones y subfunciones.

Secuencia de estados que conforman la operación que se desea realizar. Dicha operación viene especificada en una función:

Sub mySubwithParameter (myParameter)

MsgBox myParameter

End Sub

La principal diferencia entre una función y una subfunción, es que la primera devuelve un valor y la segunda no. *Referencias [4] y[5].*

2.4.5 Estructuras condicionales e iterativas.

1. Condicionales: En muchas ocasiones será necesario utilizar estructuras condicionales durante la programación para poder restringir una acción al cumplimiento de una condición de partida, comprobar que un valor se encuentra en un intervalo aceptable, etc. Para ello se utilizará le estructura *if... then...*, como se muestra a continuación:

If [condición] Then

[Estamento]

ElseIf [Condición] Then

[Estamento de ElseIf]

Else

[Estamento de Else]

End If.

2. Iterativas: Las iteraciones son frecuentemente usadas para llevar a cabos operaciones repetitivas, situación la cual se da con bastante frecuencia. Se puede hacer de distintas formas:

- *For… Next:* Cuando queremos iterar un número dado de cosas.

  Contador

  For[contador] = [inicio] To [end] {paso a paso}

  [Estamentos]

  Next

- While… Wend : Cuando iteramos hasta que el contador cumple cierta condición.

  Contador

  While [{Contador} Condición]

  [Estamentos]

  Wend

- *Do…Loop :* Cuando iteramos siempre que se cumpla cierta condición.

  Do [{While|Until} condition]

  [Estamentos]

  [Exit Do]

  [Estamentos]

  Loop

Do [Estamentos]

[Exit Do]

[Estamentos]

Loop [{While|Unitl} condition]

- For *each... Next*: Cuando se quiere iterar sobre los objetos en una determinada colección.

Referencia[4].

2.4.6 Objetos orientados a la programación.

Los objetos son una parte de la memoria del programa en los que está contenida cierta información y metodología que permite operar con dicha información almacenada. Requiere de un código preestablecido para que funcione correctamente, ya que realiza operaciones tales como:

- Pasar información a través de parámetros.
- Operaciones de cálculo que pueden:

1. Cambiar cierta parte de los datos iniciales.
2. Diseñar ciertas operaciones que no están por defecto
3. Devolver valores de los datos iniciales.
4. Devolver resultados de los cálculos usando tanto la información introducida externamente como de la contenida en el objeto. *Referencia [5]*

2.4.7 Como definir un objeto.

Para cada objeto en particular se define una clase (*classe*) que sirve como plantilla en la que se recoge cómo operan los objetos y los datos que contienen. Es conveniente resaltar que una clase puede ser utilizada para hacer funcionar uno o más objetos.

Tabla 4. Diferencias entre clase y objeto

Clase	Objeto
Describe la estructura del objeto	Es el resultado de la clase

Es una plantilla	Tiene una copia única de toda variable no-estática pero no de las variables tipo clase (*static*).
Especifica la representación de la información, el comportamiento, la interrelación (via variables, métodos y *parents*-estructura lógica)	

Referencia[4] .

2.5 Interfaz con el usuario.

A continuación se va a explicar cómo el usuario puede crear una interfaz para comunicarse e interactuar con el programa, llegando así a la programación por eventos explicada anteriormente.

Para comenzar es necesario insertar un formulario en el que se añadirán los distintos controles necesarios que servirán de ayuda para introducir los valores necesarios o demandar al usuario determinadas operaciones. Para insertar un formulario, abrimos la pestaña *insert-->Useform.*

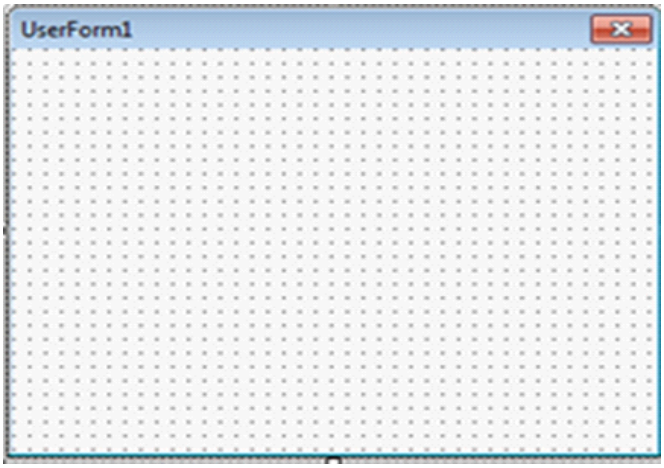

Figura 33.Useform

A continuación es necesario activar la ventana "*Toolbox*", lo cual haremos desde la pestaña *view--> Toolbox.*

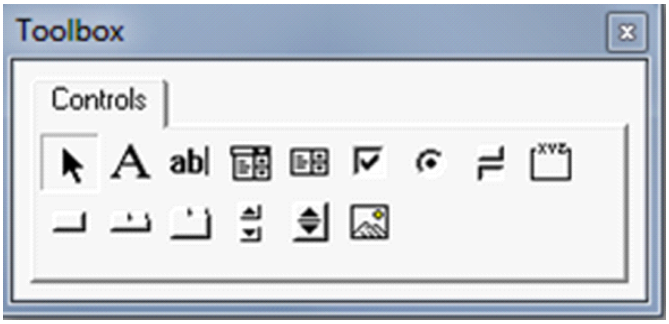

Figura 34.Herramientas de diseño

Se activará una ventana con los distintos controles que posee VBA para que usuario y programa interactúen. A continuación se van a explicar los controles de los que se disponen para diseñar el formulario.

Tabla 5. Opciones de la barra de herramientas

Símbolo	Etiqueta	Descripción
	Label	Permite escribir títulos o comentarios
	Textbox	Permite al usuario introducir texto
	ListBox	Control en el que se muestran varios registros, pudiendo seleccionar uno o más de uno
	ComboBox	Control parecido al ListBox con una propiedad llamada*Style*, que permite 3 formas distintas de presentar una lista
	CheckBox	Permite seleccionar una acción al usuario
	OptionButton	Permite seleccionar una opción al usuario.
	ToggleButton	Botón para selección de opciones
	Frame	Agrupar diferentes objetos referidos a un mismo tema o
	CommandButton	Permite ejecutar un evento
	TabStrip	Separadores o etiquetas
	MultiPage	Contenedor para una colección de

		objetos
	ScrollBar	Permite tener una barra para desplazamientos
	SpinButton	Permite aumentar o disminuir la cifra conforme se presionan las flechas del control
	Image	Insertar una imagen en el formulario

Referencia[4].

Con dichos controles podemos diseñar la interfaz con la que se quiere trabajar. Sin embargo es necesario introducir un código y programar cada uno de ellos ya que la interfaz de por si no realiza ninguna operación.

Para añadir código a cualquier control simplemente hay que hacer doble clic sobre el mismo en el formulario y se abrirá una ventana como la del ejemplo que se muestra a continuación, el cual se corresponde en este caso con un *command botton*:

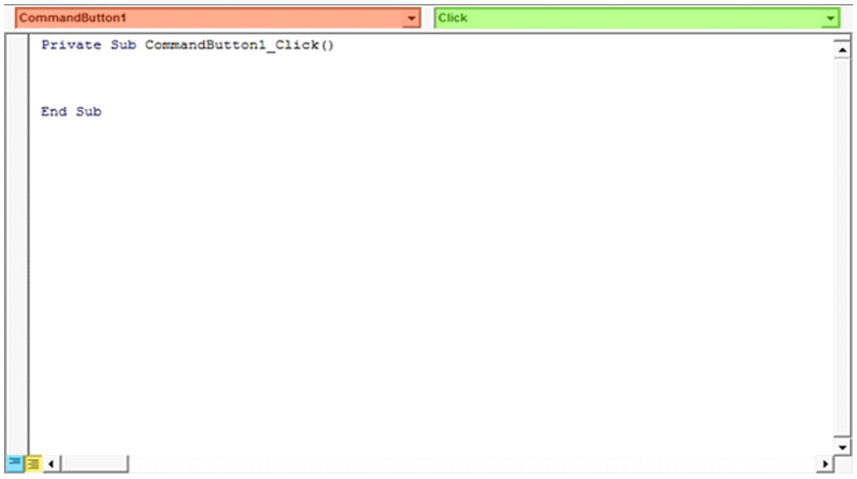

Figura 35. Editor

Como se observa en la ventana adjunta se distinguen dos pestañas, una roja que indica la lista de objetos disponibles para programar, y una verde, que indica para el objeto seleccionado, los eventos disponibles.

En los siguientes capítulos se explicará cómo funcionan algunos de los controles que se han mostrado en esta sección.

3 Herramientas de programación para engranajes

3.1 Introducción.

A continuación se van a ilustrar todos los procedimientos que se han llevado a cabo para realizar algunos de los engranajes que se describieron en el primer apartado de dicho trabajo.

Se trabajará con tres módulos principalmente, el módulo "*sketcher*", para realizar los diferentes *sketches* necesarios, y los módulos "*Part Design*" y "*Wireframe and surface Design*" ,comprobándose que a pesar de que en el caso de los engranajes se pueden realizar perfectamente con cualquiera de los dos, la versatilidad del módulo de superficies es mucho mayor. En el presente proyecto se realizarán los engranajes usando el módulo *Part Design* principalmente.

3.2 Arranque.

En primer lugar se va a explicar cómo se define y establece el entorno de trabajo sin necesidad de que el *part* se encuentre abierto, lo cual se hará activando algunos parámetros que permitan al programa responder a las acciones que se le indican.

```
Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part

Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part
```

Código 1. Arranque de documentos

El *PartDocument* activa el *Part Design* , donde se definen el resto de elementos que componen el proyecto en el que se está trabajando. También se define el archivo *part* donde se va a trabajar, el cual se establece en el *PartDocument* definido anteriormente.

Una vez definido el *part*, se creará el árbol de trabajo, que como es sabido es dónde se reflejarán todas las operaciones que realicemos a lo largo de la realización de nuestro proyecto. Para ello, establecemos el *body* de trabajo, el cual, como se hizo anteriormente, se definen primero *bodies1*, donde se establecerá el *body*.

Como se puede comprobar, la estructura en la que se trabaja está bien jerarquizada, donde cada archivo está activado dentro de otro de manera que en el caso de que falle alguno de ellos el programa no podrá realizar las operaciones que se demandan puesto que se produce un fallo global del mismo.

```
Dim bodies1 As Bodies
Dim body1 As Body
Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
```

Código 2. Bodies

Al igual que cuando diseñamos cualquier tipo de pieza manualmente, lo primero que se va a realizar es generar un sketch donde poder dibujar. Dicho sketch se genera dentro del *body* definido anteriormente. También es necesario definir el sistema de referencia 3D con el que vamos a trabajar y al igual que se hace cuando se trabaja de manera manual, tenemos que definir en qué plano vamos a trabajar. Finalmente, colocamos el sketch en el que dibujaremos dentro del plano de referencia definido.

```
Dim sketches1 As Sketches
Set sketches1 = body1.Sketches
Dim originElements1 As OriginElements
Set originElements1 = part1.OriginElements
Dim reference1 As Reference
Set reference1 = originElements1.PlaneYZ
Dim sketch1 As Sketch
Set sketch1 = sketches1.Add(reference1)
```

Código 3. Sketches

Una vez creados el plano y el sketch de trabajo, tenemos que definir las direcciones dentro del mismo de forma que se sepa siempre en qué zona del plano nos encontramos.

```
Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0# 'Vector unidad x en el plano X
arrayOfVariantOfDouble1(1) = 0# 'Vector unidad y en el plano X
arrayOfVariantOfDouble1(2) = 0# 'Vector unidad z en el plano X
arrayOfVariantOfDouble1(3) = 0# 'Vector unidad x en el plano Y
arrayOfVariantOfDouble1(4) = 1# 'Vector unidad y en el plano Y
arrayOfVariantOfDouble1(5) = 0# 'Vector unidad z en el plano Y
```

```
arrayOfVariantOfDouble1(6) = 0# 'Vector unidad x en el plano Z
arrayOfVariantOfDouble1(7) = 0# 'Vector unidad y en el plano Z
arrayOfVariantOfDouble1(8) = 1# 'Vector unidad z en el plano Z
Set sketch1Variant = sketch1
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
'Establece el sistema de ejes absoluto del sketch en 3D
```

Código 4. Vector de coordenadas

```
part1.InWorkObject = sketch1
```

Finalmente se informa al programa que dicho *sketch* es el lugar de trabajo y que todas las operaciones que se realicen a partir de este punto serán sobre el mismo.

A continuación se van a definir y establecer en primer lugar el conjunto de herramientas para realizar el dibujo 2D, y a continuación, todos los elementos geométricos necesarios.

```
'Establecimiento del conjunto de herramientas
'2D y asignación al sketch de trabajo.
Dim factory2D1 As Factory2D
Set factory2D1 = sketch1.OpenEdition()

'Se establecen los elementos geométricos
Dim geometricElements1 As GeometricElements
Set geometricElements1 = sketch1.GeometricElements

'Se define el sistema de ejes dentro del sketch
Dim axis2D1 As Axis2D
Set axis2D1 = geometricElements1.Item("AbsoluteAxis")

'Establecimiento de las direcciones horizontal y vertical
Dim line2D1 As Line2D
Set line2D1 = axis2D1.GetItem("HDirection")
line2D1.ReportName = 1
Dim line2D2 As Line2D
Set line2D2 = axis2D1.GetItem("VDirection")
line2D2.ReportName = 2

el sketch y se establece como objeto de trabajo
Sketch1.closeEdition
Part1.InWorkObject=skethc1
Part1.Update
```

Código 5. Objeto factory 2D

Las últimas líneas de código cierran el *sketch* en cuestión, lo establece como objeto de trabajo (es decir, sobre el que se realizarán las operaciones para el modelado de un sólido o superficie), y lo carga en el *part*.

Como se puede observar, existe una jerarquía en la definición de los distintos archivos, de manera que para acceder a un determinado nivel, es preciso que se hayan definido y establecidos los niveles anteriores. *Referencia[4]*.

3.3 SKETCHER

Dentro de dicho módulo encontramos una serie de objetos y colecciones que serán de gran utilidad.

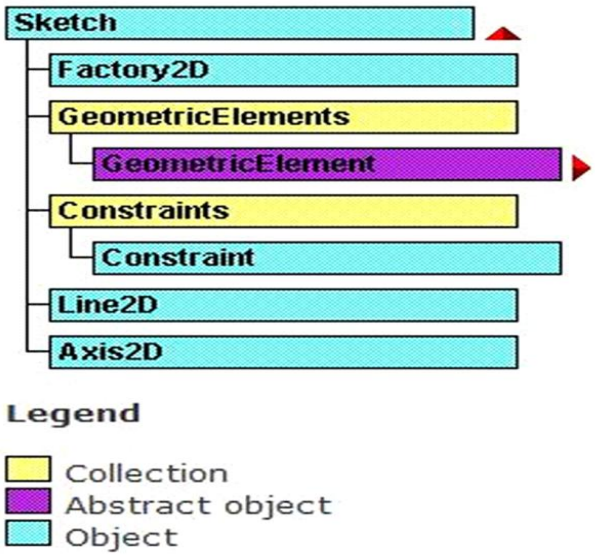

Figura 36. Estructura del módulo Sketcher

Como se puede observar, los objetos que se encuentran en dicho módulo son *"Factory2D", "Line2D", y "Axis2D*", dentro de los cuales se encuentran los siguientes métodos:

Tabla 6. Objetos del módulo sketcher

Objeto	Comentario	Contenido
Factory2D	Incluye todos los métodos necesarios para poder trabajar en el módulo Sketcher	*CreateCircle*
		CreateClosedCircle
		CreateClosedEllipse
		CreateControlPoint
		CreateEllipse
		CreateHyperbola
		CreateIntersection

		CreateIntersections
		CreateLine
		CreateLineFromVector
		CreateParabola
		CreatePoint
		CreateProjection
		CreateProjections
		CreateSpline
Line2D	Engloba tres métodos que dan el uni-vector de la dirección de la recta, un punto sobre la línea y un último que permite modificar las características de la línea infinita.	*GetDirection*
		GetOrigine
		SetData
Axis2D	Vienen detalladas las propiedades del sistema de coordenadas	*HorizontalReference*
		Origin
		VerticalReference

Una vez vista la estructura del módulo *sketcher*, se va a describir cómo usar los diferentes métodos que componen los principales objetos y colecciones de dicho módulo para diseñar la geometría de los diferentes perfiles en cuestión. *Referencia [4]*

3.3.1 Crear un punto.

Como se ha observado en la tabla 6, el objeto factory2D del módulo *sketcher* posee herramientas para crear una amplia gama de geometrías. Se va a describir a continuación como crear un punto. Para ello, se usará la siguiente línea de código:

```
Dim point2D1 As Point2D
Set point2D1 = factory2D1.CreatePoint(coord_x, coord_y)
point2D1.ReportName = 3
```

Código 6. Creación de un punto

Solo son necesarios como argumentos de entrada las coordenadas del punto en cuestión. Se puede añadir un nombre al punto para que sea más fácil localizarlo en el sketch, así como imponer que dicho punto no sea de construcción, que es como CATIA lo crea por defecto.

```
point2D1.ReportName = 3
Point2D1.construction=false
```

Código 7. Geometría en construcción

3.3.2 Crear una recta.

Otra herramienta de vital importancia que se encuentra en el objeto *factory2d* del módulo *sketcher* es la que permite crear una recta a partir de dos puntos, el punto de origen y el punto final. El código necesario para crear una recta es:

```
Dim line2D1 as line2D
Set line2D1=factory2D1.createline(x1,y1,x2,y2)
Line2d1.startpoint=Point2D1
Line2d1.endpoint=point2D2
```

Código 8. Creación de una recta

Los puntos de origen y final (*Point2D1, Point2D2)* se crean como se ha ilustrado en al apartado anterior, y para que la recta queda perfectamente definida, es necesario establecer que dichos puntos son los puntos de origen y fin, de forma que sepa que el programa que se trata de hacer una sola recta y no continúe con otra después.

3.3.3 Crear un círculo o arco de circunferencia.

Es sumamente importante saber generar círculos o arcos de circunferencia a la hora de formar partes curvas en las piezas.

Para definir un círculo cerrado necesitamos tres variables, que son las coordenadas x e y del punto que será el centro de la circunferencia, y el radio de la misma.

Para definir el círculo es necesario en primera instancia crear el punto de origen, *point2D1*, con la orden *factory2D1.CreatePoint()* en las coordenadas que corresponda. Una vez establecido el punto, con *factory2D1.CreateClosedCircle()* se genera el círculo. Como puede apreciarse, el definir el punto sirve para que a través de *circle2D1.CenterPoint* el programa centre la circunferencia en dichas coordenadas. A continuación se muestran las líneas de código con las que se crea el círculo cerrado.

```
Dim circle2D1 As Circle2D
Set circle2D1 = factory2D1.CreateClosedCircle(x1, y1, r)
circle2D1.CenterPoint = point2D1
circle2D1.ReportName = 4
```

Código 9. Creación de un círculo

También es de gran interés conocer un método para crear arcos de circunferencia puesto que es uno de los elementos que más se usan después a la hora de realizar piezas con zonas curvas.

Para ello usaremos el comando *factory2D1.CreateCircle(x,y,r,start,finish*), donde como se puede observar es necesario indicar cinco variables:

x: coordenada horizontal del centro del arco de circunferencia.
y: coordenada vertical del centro del arco de circunferencia.
start: ángulo en radianes del punto donde comienza el arco.

finish: ángulo en radianes del punto final donde finaliza el arco.

Las líneas de comando para realizar el arco de circunferencia siguen la misma estructura que para el círculo cerrado como viene reflejado a continuación:

```
Dim circle2D1 As Circle2D
Set circle2D1 = factory2D1.CreateCircle(x, y, r, ang_inicio [radianes], ang_fin[radianes])
```

Código 10. Creación de un arco de circunferencia

3.3.4 Crear un spline.

Otra operación de real interés para la realización del presente proyecto es conocer algún método para generar curvas a partir de puntos de la misma, es decir, se quiere crear una curva mediante una aproximación polinómica a partir de distintos puntos que la conforman.

El objeto *factory2D* presenta una herramienta para crear dichas curvas a partir de diferentes puntos de control, llamados "*control Point*".

Para originar dichos puntos de control se usa el comando *createcontrolpoint(x,y)*, y es necesario crear un vector que incluya todos los *controlpoint* que conforman la curva que queremos desarrollar. Para programar un *spline* se usarán las siguientes líneas de código:

```
Dim controlPoint2D1 As ControlPoint2D
Set controlPoint2D1 = factory2D1.CreateControlPoint(x1, y1)
controlPoint2D1.ReportName = 3

Dim controlPoint2D2 As ControlPoint2D
Set controlPoint2D2 = factory2D1.CreateControlPoint(x2, y2)
controlPoint2D2.ReportName = 4

Dim controlPoint2D3 As ControlPoint2D
Set controlPoint2D3 = factory2D1.CreateControlPoint(x3, y3)
controlPoint2D3.ReportName = 5

Dim controlPoint2D4 As ControlPoint2D
Set controlPoint2D4 = factory2D1.CreateControlPoint(x4, y4)
controlPoint2D4.ReportName = 6

Dim arrayOfObject1(3)
Set arrayOfObject1(0) = controlPoint2D1
Set arrayOfObject1(1) = controlPoint2D2
Set arrayOfObject1(2) = controlPoint2D3
Set arrayOfObject1(3) = controlPoint2D4


Dim spline2D1 As Spline2D
Set factory2D1temp = factory2D1
Set spline2D1 = factory2D1temp.CreateSpline(arrayOfObject1)
```

```
spline2D1.ReportName = 7
```

Código 11. Creación de un Spline

Como se puede observar el único argumento de entrada que se necesita para el correcto funcionamiento de dicha herramienta es el vector de objetos (que en este caso son *controlpoints*) . En el caso de este proyecto serán puntos de la involuta que define el perfil de los dientes de un engranaje, y los cuales se calcularán con la ecuación paramétrica de la misma.

3.4 Restricciones.

A continuación se va explicar cómo establecer los *contraints*, es decir, las restricciones que permiten mantener en una posición fija toda la geometría que se crea en el *sketch.*

Los *constraints* trabajan con referencias, es decir, hay que establecer la referencia de cada uno de los objetos del sketch. Una vez definidas éstas, es necesario una instrucción que permita relacionarlas entre sí, para lo cual se usa la siguiente línea de código: *constraintsX.AddBiEltCst(catCstTypeDistance, reference1, reference2).* En dicho comando se observan dos propiedades de gran interés.

La instrucción *AddBiEltCs* nos indica que la restricción va a usar dos referencias, es decir, se van a relacionar dos objetos. Se pueden relacionar uno, dos o tres objetos. Para ello solo se ha de cambiar el prefijo "Bi-" por el correspondiente. Es decir , una restricción tal que *AddMonoEltCs* solo necesitará una referencia, y otra cómo *AddTriBiEltCs* tres. En este proyecto se van a trabajar con restricciones de uno, dos y tres objetos.

CatCsTypeDistance refleja el tipo de restricción que se quiere imponer. Existen numerosos tipos de restricciones cómo se muestra en la tabla 7:

Tabla 7. Restricciones

Número de referencias	Tipo
BiEtlcs	*CatCsTypeAnnulContact*
	CatCsTypeParallelims
	CatCsTypePerpendicularity
	CatCsTypeChamfer
	CatCsTypeConcentry
	CatCsTypeDistance
	CatCsTypeHorizontaly
	CatCsTypeLength

	CatCsTypeLineContact
	CatCsTypeMajor/MinorRadius
	CatCsTypeMidpoint
	CatCsTypeOn
	CatCsTypePlanarangle
	CatCsTypeDistance
	CatCsTypeTangency
MonoEtlC	*CatCsTypeRadius*
TriEtlcs	*CatCsTypeSimetry*

Con las órdenes indicadas en la tabla el programa sabe que las referencias en cuestión se encuentran relacionadas según el tipo de restricción escogida. Sin embargo no conoce la cantidad exacta del parámetro (el cual dependerá del tipo de restricción en cuestión) que las relaciona (distancia, ángulo…), por lo que hay que señalar que *contraint* tiene dos modos de trabajo:

1. Modo *constraint*: El valor asignado restringe la geometría en dicha posición
   >> *constraint1.Mode = catCstMode**Driving**Dimension*
2. Modo *Measurement*: El valor solo refleja aquello que puede ser observado desde dicha posición.
   >> *constraint1.Mode = catCstMode**Driven**Dimension*

Se van a mostrar a continuación algunas de las restricciones que se van a usar en el desarrollo del proyecto.

3.4.1 CatCsTypeDistance.

Se utiliza para fijar la posición respecto al origen de coordenadas o la distancia entre dos objetos. También puede usarse para determinar la longitud de algunos objetos.

Como se puede observar además de los comandos introducidos al inicio de la subsección, es necesario declarar una longitud la cual es definida como la dimensión de la restricción. Así mismo se debe dar el valor que tomará dicha longitud.

```
'.....................Establecimiento de los constraints.....................
Dim constraints1 As Constraints
Set constraints1 = sketch1.Constraints

'..Constraint de distancia horizonal de un punto  respecto al origen de coordenadas.
Dim reference2 As Reference
Set reference2 = part1.CreateReferenceFromObject(point2D1)
Dim reference3 As Reference
Set reference3 = part1.CreateReferenceFromObject(line2D2)
```

```
Dim constraint1 As Constraint
Set constraint1 = constraints1.AddBiEltCst(catCstTypeDistance, reference2, reference3)
constraint1.Mode = catCstModeDrivingDimension
Dim length1 As Length
Set length1 = constraint1.Dimension
length1.Value = x1
```

Código 12. catCstTypeDistance

3.4.2 CatcsTypeRadius.

Con esta restricción se puede fijar el radio de circunferencias y arcos de circunferencia. De nuevo es necesario definir la longitud y darle el valor del radio que se desea.

```
'.....Establecimiento de los constraints.....
Dim reference6 As Reference
Set reference6 = part1.CreateReferenceFromObject(circle2D1)
Dim constraint3 As Constraint
Set constraint3 = constraints1.AddMonoEltCst(catCstTypeRadius, reference6)
constraint3.Mode = catCstModeDrivingDimension
Dim length3 As Length
Set length3 = constraint3.Dimension
length3.Value = r1
```

Código 13. CatcsTypeRadius

3.4.3 CatcsTypeOn.

La restricción *CatcsTypeOn* es una de las más interesantes de las restricciones existentes.

Dicha restricción permite imponer la coincidencia de dos puntos o curvas cualesquiera sean, por lo que permite introducir imposiciones geométricas muy importantes en el sketch en el que se esté trabajando.

Por otro lado, se puede usar para hallar la intersección entre dos curvas planas que ya estén perfectamente restringidas. Para ello basta con crear un punto de coordenadas arbitrarias, al cual no restringimos su posición, y realizar dos restricciones *TypeOn*, es decir, imponer la coincidencia del punto con cada curva, obteniendo así la intersección de ambas.

```
'ejemplo uso de restricción TypeOn para hallarla interseccion de 'un punto con dos 'curvas.
Dim reference20 As Reference
Set reference20 = part1.CreateReferenceFromObject(spline2D1)
Dim reference30 As Reference
Set reference30 = part1.CreateReferenceFromObject(pointict)
Dim constraint200 As Constraint
Set constraint200 = constraints1.AddBiEltCst(catCstTypeOn, reference20, reference30)

Dim reference70 As Reference
Set reference70 = part1.CreateReferenceFromObject(circle1)
Dim reference50 As Reference
```

```
Set reference50 = part1.CreateReferenceFromObject(pointict)
Dim constraint300 As Constraint
Set constraint300 = constraints1.AddBiEltCst(catCstTypeOn, reference70, reference50)
```

Código 14. CatcsTypeOn

3.4.4 CatcsTypeVertically.

En ocasiones es necesario crear una recta vertical de las cual no conocemos ni coordenadas de los puntos de inicio o fin ni la longitud.

En estos casos es más cómodo crear una recta arbitraria e imponer que sea vertical a crear dos puntos arbitrarios como puntos de inicio y fin de la recta y restringir sus coordenadas (se necesitan más restricciones).

```
Dim reference6 As Reference
Set reference6 = part1.CreateReferenceFromObject(line2D4)
Dim reference7 As Reference
Set reference7 = part1.CreateReferenceFromObject(line2D2)
Dim constraint3 As Constraint
Set constraint3 = constraints1.AddBiEltCst(catCstTypeVerticality, reference6, reference7)
constraint3.Mode = catCstModeDrivingDimension
```

Código 15. CatcsTypeVertically

3.4.5 CatcsTypeHorizontally.

Lo mismo sucede en el caso de una recta horizontal:

```
Dim reference4 As Reference
Set reference4 = part1.CreateReferenceFromObject(line2D5)
Dim reference5 As Reference
Set reference5 = part1.CreateReferenceFromObject(line2D1)
Dim constraint2 As Constraint
Set constraint2 = constraints1.AddBiEltCst(catCstTypeHorizontality, reference4, reference5)
constraint2.Mode = catCstModeDrivingDimension
```

código 16. CatcsTypeHorizontally

3.4.6 CatcsTypeSymetry.

Está es una de las herramientas más potentes y complejas dentro de las restricciones. Para su uso es necesario definir tres referencias, puesto que trabaja con tres objetos: objeto al que se le aplica la simetría, objeto respecto al que se aplica la misma y el eje de simetría.

Es importante recalcar que para aplicar dicha restricción se necesitan tres objetos, por lo que aparte del objeto de referencia y el eje de simetría, es necesario crear un objeto arbitrario (aunque del mismo tipo del primero) al que aplicar la simetría.

```
        Dim contraintmirror1 As Constraint
        Dim refsim1 As Reference
        Dim refsim2 As Reference
        Dim refsim3 As Reference
        Set refsim1 = part1.CreateReferenceFromObject(point1)
        Set refsim2 = part1.CreateReferenceFromObject(point11)
        Set refsim3 = part1.CreateReferenceFromObject(linemirror)
        Set contrainstmirror1 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim1,
refsim2, refsim3)
```

Código 17. CatcsTypeHorizontally

4 CATPART.

La sección de CATIA denominada CATPART englobe tres módulos:

- *PartDesign*
- *Wireframe and surfaceDesign*

GenerativeShapeDesign.

De dichos módulos se usarán el módulo PartDesign, para generar módelos sólidos, y el Wireframe And SurfaceDesign, para generar superficies. A continuación se estudian en profundidad dichos módulos.

4.1 PART DESIGN.

Como se ha dicho anteriormente, el módulo *PartDesign* se usa para generar modelos sólidos 3D con la ayuda de una gran variedad de objetos (abstractos y particulares) y colecciones que se muestran a continuación en el esquema de la estructura del módulo:

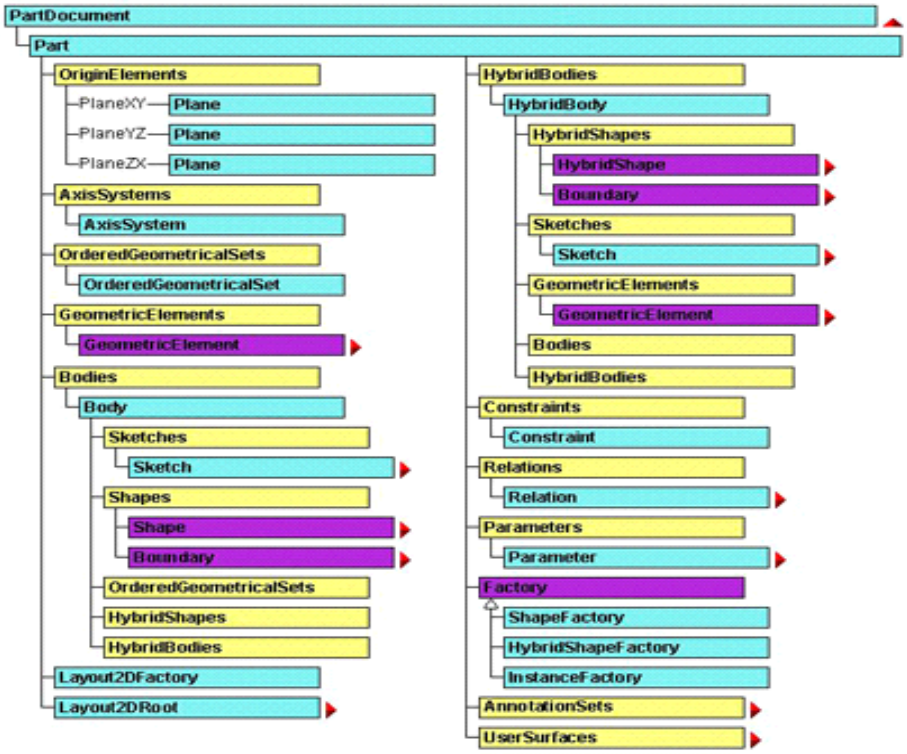

Figura 37. Estructura del módulo PartDesign

Como se muestra en la figura 37, se especifica el origen del sistema de referencia, los ejes, paquetes geométricos, *bodies*, *sketches*, relaciones, parámetros, etc.

Como ya se ha explicado anteriormente, la estructura de CATIA es muy jerárquica, de manera que para acceder a los planos de referencia por ejemplo, habrá que definir en primer lugar los objetos y propiedades referentes a *Plane* que contienen dichos planos (*PartDocument-->Part-->Originelements*).

Figura 38. Estructura general

A continuación se muestran las propiedades de los objetos que conforman el módulo *Partdesign*, las cuales se usarán comúnmente a lo largo de la programación:

Tabla 8. Objetos PartDesign

Propiedad	Comando	Comentario
OriginElements	Dim origin1 As OriginElements Set origin1 = part1.OriginElements	Constituye el sistema de referencia absoluto del documento de planos XY, XZ e YZ.
AxisSystems	Dim refsist1 As AxisSystems Set refsist1 = part1. AxisSystems	Es la colección de sistemas de referencia que pueden existir en el documento *part*.
Geometric Elements	Dim geometric1 As GeometricElements Set geometric1 = Part1.GeometricElements	Colección de elementos geométricos 3D del *partdocument* generadas directamente en 3D, es decir, sin la mediación del módulo *Sketcher*
Bodies	Dim Bodies1 As Bodies Set Bodies1 = Part1.Bodies	Colección de cuerpos sólidos dentro del

		partdocument
HybridBodies	Dim HybBod1 As HybridBodies Set HybBod1 = Part1.HybridBodies	Colección de *OpenBodies* (elementos de referencia)
Constraints	Dim Constraints1 As Constraints Set Constraints1 = Part1.Constraints	Colección de restricciones geométricas y dimensionales del *partdocument*.
Relations	DimRelations1 AsRelations SetRelations 1 = Part1.Relations	Colección de relaciones del *partdocument*.
Parameters	DimParameters1 AsParameters SetParameters1 = Part1.Parameters	Colección completa de todos los parámetros del *partdocument.*

Reference[4].

A continuación, se va a explicar el objeto *"shapefactory*", que contiene todas las operaciones que permiten pasar de un dibujo 2D a un sólido 3D, o que una vez obtenido este, realizan transformaciones sobre el mismo. Debido a la amplia gama de operaciones que presenta dicho objeto, vamos a incluir solamente las más importantes y todas las que usaremos a lo largo del proyecto.

Tabla 9. Herramientas del Shapefactory

Objeto	Comentario	Contenido
Shapefactory	Incluye todos los métodos necesarios para poder trabajar en el módulo *Partdesign.*	*AddnewChamfer*
		AddnewCircPattern
		AddnewCloseSurface
		AddnewDraft
		AddnewEdgeFilletWith ConstantRadius
		AddnewEdgeFilletWith VaryingRadius
		AddnewFaceFillet
		AddnewHole
		AddnewLoft

		AddnewMirror
		AddnewPad
		AddnewPocket
		AddnewRecPattern
		AddnewRib
		AddnewShaft
		AddnewSlot
		AddnewStiffner
		Addnewthickness
		Addnewthicksurface

A continuación se van a explicar las herramientas que conforman el objeto *shapefactory* que se han usado para realizar los engranajes. Lo primero que se debe hacer es definir y cargar el objeto en cuestión. Para ello, se escribirá siempre las siguientes líneas de código:

```
Dim shapefactory1  As shapefactory
Set shapefactory1=part1.shapefactory
```

Código 18. Definición de shapefactory

Una vez cargado dicho objeto, se podrá hacer uso de todas las herramientas que se incluyen en el mismo.

4.1.1 Pad.

El *pad* es una herramienta que se utiliza para realizar una extrusión. Es muy potente ya que con dicha herramienta, y con alguna más que explicaremos posteriormente, como *hole y pocket*, se pueden realizar una infinidad de sólidos. Para realizar un *pad* se usarán las siguientes líneas de código:

```
Dim pad1 As Pad
Set pad1 = shapeFactory1.AddNewPad(sketch1, espesor)
pad1.Name = "Nombrepad"' Darle un nombre al pad
Dim limit1 As Limit
Set limit1 = pad1.FirstLimit
```

```
Dim length1 As Length
Set length1 = limit1.Dimension
length1.Value = espesor
```

Código 19. Pad

La herramienta pide dos argumentos solamente, el *sketch* que se va a extruir, y el espesor que se quiere.

Sin embargo, es necesario restringir la altura del *pad* para que el programa no de errores. Para ello se define un límite, y se establece como el primer límite del *pad* realizado. Finalmente para darle un valor numérico a dicho límite, se define una longitud, y se establece como la dimensión del límite.

4.1.2 Pocket.

La estructura del *pocket* es prácticamente igual a la del *pad* salvo que, en este caso, lo que se quiere hacer es un hueco sobre un sólido hecho previamente, de manera que, tendremos que fijar una referencia sobre la superficie del mismo en la que se quiere generar el hueco.

Ello se puede hacerlo de dos formas:

1. Creando una referencia directamente sobre el sólido usando las líneas de código mostradas en el código 20. Se observa que se especifica que el *pocket* se realizará sobre una superficie *FSur:Face* ,especificando que se realiza sobre el *pad* que se ha creado previamente.

```
Dim reference10 As Reference
Setreference10=
part1.CreateReferenceFromName("Selection_RSur:(Face:(Brp:(Pad.1;2);None:();~~
Cf11:());Pad.1_ResultOUT;Z0;G3055)")
```

Código 20. Referencia del pad

2. O bien creando un plano con un offset determinado, tal que dicho offset coincida con la altura del sólido en cuestión, quedando el plano sobre la superficie en la que queremos generar el pocket. Para ello se crea un plano, herramienta que se encuentra dentro del objeto hybridshapefactory, en el cual se encuentran las principales herramientas del módulo WireFrameandSurfaceDesign cómo se explicará en la siguiente sección.

```
Dim hybridShapePlaneExplicit1 as HybridShapePlaneExplicit
Set hybridShapePlaneExplicit1 = originElements1.PlaneXY
Dim reference10 As Reference
Set reference10 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit1)
```

```
        Dim hybridShapePlaneOffset1 As HybridShapePlaneOffset
        Set hybridShapePlaneOffset1 = hshapefactory1.AddNewPlaneOffset(reference10,
altura, False)
```

Código 21. Continuación referencia del pad

Como se puede observar, en primer lugar se crea un plano XY, que servirá de referencia a nuestro plano, indicando que son paralelos.

Una vez explicado cómo crear la referencia para el *pocket*, el resto del proceso se realiza de manera análoga al caso del *pad*, quedando, usando el primero de los casos indicados anteriormente:

```
        Dim reference10 As Reference
        Set reference10 =
part1.CreateReferenceFromName("Selection_RSur:(Face:(Brp:(Pad.1;2);None:();~
        ~ Cf11:());Pad.1_ResultOUT;Z0;G3055)")

        Dim sketch2 As Sketch
        Set sketch2 = sketches1.Add(reference10)

        Dim pocket1 As Pocket
        Set pocket1 = shapeFactory1.AddNewPocket(sketch2, espesor)
        pocket1.Name = "hueco interno"' Dar nombre al agujero

        Dim limit2 As Limit
        Set limit2 = pocket1.FirstLimit
        limit2.LimitMode = catUpToSurfaceLimit ' Se define su profundidad a la
        ' superficie más próxima
```

Código 22. Pocket

4.1.3 Hole.

Al igual que en el caso del *pocket*, queremos realizar un orificio sobre un sólido hecho previamente, y de nuevo se tiene que crear una referencia sobre el mismo, para lo que usaremos las siguientes líneas de código:

```
        Dim reference3 As Reference
        Set reference3 =part1.CreateReferenceFromBRepName("FSur:~
~Face:(Brp:(cilindro;2);None:();Cf11:());WithTemporaryBody;~
        ~WithoutBuildError;WithInitialFeatureSupport;MonoFond;~
        ~MFBRepVersion_CXR15)", pad1)
```

Código 23. Referencia del hole

Sin embargo en este caso no será necesario realizar ningún *sketch*, ya que la propia herramienta incluye esta acción, solo teniendo que tener en cuenta que ésta trabaja con diámetros.

Cabe destacar también que hay varias herramientas para programar un *hole*, cada una de las cuales nos pedirán distintos argumentos, y se usarán unas u otras según el caso. Algunos de las distintas formas de programarlo se muestran a continuación:

Orden	Parámetros
AddNewHole	X, Y,Z (coordenadas absolutas), profundidad
AddNewHoleFromPoint	X,Y,Z (coordenadas absolutas), plano de referencia, profundidad
AddNewHoleFromRefPoint	Punto origen, plano de referencia, profundidad
AddNewHoleFromSketch	Sketch, profundidad
AddNewHoleWithConstraints	X,Y,Z (coordenadas absolutas), contorno, plano de referencia, profundidad. *(si el contorno es circular, el hole será concéntrico a dicho contorno)*

A continuación se explica cómo programar un *hole* a partir de un punto, donde como se aprecia, se nos piden como argumentos de entrada las coordenadas absolutas del punto, un plano de referencia y la profundidad del mismo.

```
Dim hole1 As Hole
Set hole1 = shapeFactory1.AddNewHoleFromPoint(X,Y,Z,~~reference3, longitud)
hole1.Type = catSimpleHole
hole1.AnchorMode = catExtremPointHoleAnchor
hole1.BottomType = catFlatHoleBottom
hole1.Name = "hueco interno"

Dim limit2 As Limit
Set limit2 = hole1.BottomLimit
limit2.LimitMode = catOffsetLimit
Dim length3 As Length
Set length3 = hole1.Diameter
length3.Value = 2 * radio_int 'Hole trabaja con diámetro no con radio
```

Código 24. Hole

Además, es necesario aportar información extra para que el programa funcione correctamente. Para ello se definen los siguientes elementos:

- El ***Type***: indica el tipo de *hole* que se quiere realizar, en este caso un *catSimpleHole.*

- El ***AnchorMode*:** indica donde está anclado el *hole*. En el caso del ejemplo se indica *catExtremPointHoleAnchor*, indicando que se trata de un *hole* que parte de un punto específico en el extremo del sólido.

- El ***BottomType***:se precisa dónde finaliza el agujero que se desea realizar, que en *catFlatHoleBottom* se está especificando que es hasta el fondo del sólido.

4.1.4 Shaft.

La herramienta *shaft* es una herramienta muy poderosa dentro del ámbito de la ingeniería, ya que se utiliza para generar sólidos de revolución, los cuales son muy comunes tanto en dicho campo como en los objetos que nos rodean.

Para realizar un *shaf*t es necesario en primer lugar crear el *sketch* que se va a revolucionar en torno a un eje para conformar el sólido de revolución deseado.

Los dos parámetros en los que se basa la herramienta son:

- **El ángulo de revolución:** es muy importante destacar que se debe programar en grados sexagesimales y no en radianes como ocurría en el círculo (más información en el capítulo 3).

- **El eje de revolución:** es necesario indicar la dirección alrededor de la cual se genera la pieza.

En las siguientes líneas de código se ilustra como programar un *shaft*:

```
Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory
Dim shaft1 As Shaft
Set shaft1 = shapeFactory1.AddNewShaft(sketch1)

' Se define el ángulo de revolución

Dim angle1 As Angle
Set angle1 = shaft1.FirstAngle
angle1.Value = TextBox1' Valor del ángulo introducido en el menú

Dim parameters1 As Parameters
Set parameters1 = part1.Parameters
Dim length1 As Length
Set length1 = parameters1.Item("Part1\PartBody\Shaft.1\ThickThin1")
length1.Value = 0
shaft1.Name = "Revolución"

' Se establece el eje de revolución

Dim reference12 As Reference
Set reference12 = part1.CreateReferenceFromObject(line2D2)
```

```
shaft1.RevoluteAxis = reference12
```

Código 25. Shaft

Como se observa, es necesario definir un ángulo, y establecer que éste sea el ángulo de revolución del sólido, así como crear una referencia para definirla como el eje de rotación del *sketch.*

4.1.5 Rib.

La herramienta *rib* es una herramienta muy sencilla de programar, para la cual se necesitan generar dos *sketch*:

- El perfil: Da forma al sólido que queremos generar y podrá ser un *sketch* abierto en el caso en el que el *rib* se realice tangente a una superficie, y queramos un acabado suave sobre la misma.

- Guía: Indica el camino que queremos que siga el perfil. Se podría decir que es la dirección que queremos que siga el perfil, aunque éste puede ser cualquier curva.

Es muy importante a la hora de programar un *rib* que tanto perfil como guía tengan un punto en común, es decir, que estén en contacto en algún punto de sus geometría, ya que de otra forma. Las líneas de código para programar un *rib*, como se dijo al principio, son muy sencillas, y se muestran a continuación:

```
Dim rib1 As Rib
Set Rib = shapeFactory1.AddNewRibFromRef(refrib1, refrib2)
part1.Update
```

Código 26. Rib

4.1.6 Remove multisection solid

De las herramientas que se van a usar en el presente proyecto, *Remove Multisection Solid (loft)* es la más compleja debido a que necesita varias argumentos de entrada para su correcto funcionamiento y que la geometría debe estar perfectamente cerrada y restringida.

Además dicha herramienta requiere que el perfil que se va a usar como sección no sea muy complejo, ya que se producen errores debido a que la geometría se deforma y varios puntos de la misma pueden llegar a pertenecer al mismo punto tras la operación. Por ello es conveniente definir varios perfiles como secciones y más de una guía para que el programa sepa perfectamente cómo debe eliminarse el material.

Las líneas de código qué se usarán para realizar dicha operación se muestran a continuación:

```
‘Remove multisection solid
Dim  loft1 As Loft
Set loft1 = shapeFactory1.AddNewRemovedLoft()
Dim hybridShapeLoft1 As HybridShapeLoft
Set hybridShapeLoft1 = loft1.HybridShape

hybridShapeLoft1.SectionCoupling = 3
hybridShapeLoft1.Relimitation = 1
hybridShapeLoft1.CanonicalDetection = 2

' Dim reference1000 As Reference
' Set reference1000 = part1.CreateReferenceFromObject(curvaguía)
'hybridShapeLoft1.AddGuide reference1000


Dim reference3000 As Reference
Set reference3000 = part1.CreateReferenceFromObject(seccion1)

Dim reference4000 As Reference
'Setreference4=
part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Br
p:(GSMRotate.1;(Brp:(Sketch.1;6)));None:();Cf11:());Face:(Brp:(GSMRotate.1;(Brp:(
Sketch.1;1)));None:();Cf11:()));Cf11:());WithPermanentBody;WithoutBuildError;With
InitialFeatureSupport;MFBRepVersion_CXR15)", hybridShapeRotate1)
hybridShapeLoft1.AddSectionToLoft reference3000, 1, Nothing

’Closing points
Dim reference6000 As Reference
'Setreference6=
part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Br
p:(Sketch.3;6);None:();Cf11:());Face:(Brp:(Sketch.3;1);None:();Cf11:()));Cf11:());Wit
hPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVersion_CX
R15)", sketch1)

hybridShapeLoft1.AddSectionToLoft reference5500, 1, Nothing
‘Closing points

' Dim reference6000 As Reference
' Set t reference6=
part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Br
p:(Sketch.3;6);None:();Cf11:());Face:(Brp:(Sketch.3;1);None:();Cf11:()));Cf11:());Wit
hPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVersion_CX
R15)", sketch1)
part1.InWorkObject = hybridShapeLoft1

part1.Update
```

Código 27. Remove Multisection-solid.

Como se puede observar se definen tres objetos principalmente: Secciones, guías y *closing points.* La elección de los *closingpoints* es compleja de programar y dado que CATIA los elige por defecto, nunca se programarán salvo que la operación que se desee realizar lo requiera.

Los parámetros propios de la operación como *SectionCoupling, Relimitation y CanonicalDetection* son los que se obtienen por defecto de grabar un macro realizando dicha operación.

4.1.7 CircularPattern.

Al igual que en el caso anterior, la herramienta *CircularPattern* es una de las más difíciles de programas debido a la cantidad de argumentos de entrada que pide dicha función.

Dicha herramienta es de gran utilidad, ya que permite hacer copias de un sólido en las direcciones radial y circunferencial sin necesidad de crear un origen de coordenadas cilíndrico.

La principal diferencia que tiene dicha herramienta con todas las anteriores que se han usado para generar sólidos (excluimos con esta mención a la herramienta hole, que se usa para eliminar materia), es que no pide un sketch como entrada de la función, sino que se necesita como partida un sólido, el cual se puede realizar con cualquier operación de las descritas.

Se va a ilustrar en primer lugar el código, y posteriormente se analizará todos los argumentos que necesitamos como entrada, así como diferentes parámetros que se deben definir para el correcto funcionamiento de dicha herramienta.

```
'Circular pattern
Dim ref5 As Reference
Set ref5 = part1.CreateReferenceFromName("")
Dim ref6 As Reference
Set ref6 = part1.CreateReferenceFromName("")

Dim circPattern1 As CircPattern
Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, 720 / Z, 1, 1,
ref5, ref6, True, 0#, True)

'parametro 5=angulo del diente
```

```
'parametro4=numero de dientes

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing
circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

Dim angularRepartition1 As AngularRepartition
Set angularRepartition1 = circPattern1.AngularRepartition

Dim intParam1 As IntParam
Set intParam1 = angularRepartition1.InstancesCount
intParam1.Value = Z

Dim intParam2 As IntParam
Set intParam2 = angularRepartition1.AngularSpacing

'intParam2.Value = 360 / N
```

Código 28. CircularPattern

De las distintas opciones que permite escoger dicha herramienta, se ha explicado la que se ha usado en la realización del proyecto, la cual crea las copias que se desean basándose en los siguientes parámetros:

- Número de copias que queremos realizar.
- Espacio entre instancias(en ángulos entre mismas).

4.2 WIREFRAME AND SURFACEDESIGN.

Como se explicó anteriormente, el módulo "*Wireframe and surfaceDesign*" pertenece a la sección "*CATIApart*" de CATIA, se la cual explicamos anteriormente la estructura general.

A continuación, se va a dar una visión global del módulo *Wireframe and Surface Design*, que cómo es sabido, representa una de las herramientas más potentes de CATIA, y que hace que dicho programa tenga la relevancia e importancia a nivel mundial que tiene.

En dicho apartado se van a presentar las herramientas más importantes de dicho módulo, que posteriormente usaremos para demostrar que aunque en el caso de los engranajes dicho módulo no aporta ninguna ventaja en particular, se pueden realizar engranajes a partir de superficies, y darles espesor posteriormente para obtener un engranaje idéntico al obtenido con el módulo "part Design". Es decir, con dicho módulo no se realizará ningún cuerpo macizo, sino que siempre se obtendrán superficies huecas.

Para trabajar con superficies, es necesario activar en primer lugar los *"hybridbodies*" que se observan en el esquema general de la sección CATPART. Una vez activado, se definirá y cargará el objeto " *hybridshapefactory*", objeto que contiene todas las herramientas necesarias para trabajar con superficies.

Las principales herramientas de dicho módulo se resumen a continuación:

Tabla 10. Herramientas del hybridshapefactory

Nombre	Icono	Función
Extrude		Herramienta capaz de crear una superficie a través de la extrusión de un sketch en diferentes direcciones y dimensiones.
Fill		Crea superficies a partir de contornos creados previamente, útiles para rellenar superficies de manera suave.
Split		Capaz de eliminar partes de superficies intersecada con otras.
Join		Une diferentes superficies dando como resultado una única superficie que engloba a las demás
Close Surface		Cierra superficies y las vuelve maciza, es una forma de obtener un elemento relleno y cerrado.
Thick Surface		Otorga espesor a una superficie, obteniendo un sólido macizo.

La principal diferencia entre el módulo *Partdesign* y *Wireframe and Surface Design* es que a diferencia del primero, en el que se establece el árbol de trabajo y se crea un body, en el módulo de superficies, se define y establece un *hybridbody*, elemento que es representado en el árbol de trabajo como *Geometrical Set*.

La manera de definir y cargar tanto los *hybridbodies*, como el objeto *hybridshapefactory* es la que sigue:

```
Dim hybridBodies1 As HybridBodies
Set hybridBodies1 = part1.HybridBodies

Dim hybridBody1 As HybridBody
Set hybridBody1 = hybridBodies1.Add()

Dim hshapefactory As HybridShapeFactory
Set hshapefactory = part1.HybridShapeFactory
```

Código 29. Establecimiento deHybridBody y HybridShapeFactory

En lo que sigue, se van a explicar las herramientas para generar superficies que se han usado para modelar engranajes.

4.2.1 Extrude.

Con dicha herramienta se puede extender una curva realizada en un *sketch* en una determinada dirección, que se dará al programa con un vector o una recta, y dos límites, superior e inferior despectivamente.

```
Dim ref As Reference
Set ref = part1.CreateReferenceFromObject(sketch1)
Dim direction As HybridShapeDirection
Set direction = hshapefactory.AddNewDirectionByCoord(0, 0, 1)
Dim Extrude As HybridShapeExtrude
Set Extrude = hshapefactory.AddNewExtrude(ref, limit1, limit2, direction)

hybridBody1.AppendHybridShape Extrude

part1.InWorkObject = Extrude
```

Código 30. Extrude

Como vemos, la herramienta *AddNewExtrude* pide una referencia, motivo por el cual creamos la referencia previamente.

4.2.2 Fill.

Con dicha herramienta, podemos crear superficies suaves (planas) a partir de contornos realizados en un *sketch*, y el cual limita a la superficie en cuestión.

```
Dim ref1 As Reference
Set  ref1 = part1.CreateReferenceFromObject(sketch1)

Dim Fill As HybridShapeFill
```

```
Set Fill = hshapefactory.AddNewFill
Fill.AddBound ref1

hybridBody1.AppendHybridShape Fill
part1.InWorkObject = Fill
```

Código 31.Fill

Se puede observar, que para usar dicha herramienta, se debe definir el contorno (que en este caso viene dado por un *sketch*).

4.2.3 Join.

Dicho comando es uno de los comandos más interesantes del módulo en el que se está trabajando, puesto que permite transformar más de una superficie en una sola, lo cual es de gran utilidad para transformas la superficie resultante en un sólido macizo (usando *thick surface/close surface*), o para añadir material a toda una superficie.

```
'Unimos dos superficies cualquiera, en este caso un extrude y un fill

Dim refj1, refj2 As Reference
Set refj1 = part1.CreateReferenceFromObject(Fill)
Set refj2 = part1.CreateReferenceFromObject(Extrude)
Dim Join As HybridShapeAssemble
Set Join = hshapefactory.AddNewJoin(refj1, refj2)

hybridBody1.AppendHybridShape Join
part1.InWorkObject = Join
part1.Update
```

Código 32.Join

Se puede observar en el código anterior, que se necesitan como argumentos de la herramienta *join* dos referencias, que se corresponden con las superficies que se quieren unir.

4.2.4 Close surface.

Esta herramienta complementa al *join* explicado con anterioridad. Es usada para realizar cuerpos con acabados suaves, cerrando superficies y dándoles volumen. De manera que dicha herramienta es de gran utilidad, ya que permite pasar del módulo *Wireframe and surface Design* al *Part Design*, transformando superficies sin espesor en sólidos macizos. Cabe destacar que dicha herramienta no pertenece a *WireframeAndSurfaceDesign*, aunque es una herramienta muy potente a la hora de transformas superficies en sólidos, por lo que se ha incluido en dicho módulo.

```
'Cerrar lasuperficie
```

```
Dim refc As Reference
Set refc = part1.CreateReferenceFromObject(Join)
Dim myClose As CloseSurface
Set myClose = hpshapefactory.AddNewCloseSurface(refc)
part1.Update
```

Código 33.Close surface

4.2.5 Thick surface.

Análogamente a la herramienta anterior, "*Thick Surface*" no pertenece al módulo *Wireframe And Surface Design*, sin embargo, es incluida aquí debido a la gran utilidad de dicha herramienta.

Thick surface permite permite añadir espesor a una superficie creada previamente, lo que permite de nuevo obtener un sólido con cierto espesor a partir de una superficie.

```
espesor = TextBox1
Dim reft As Reference
Set reft = part1.CreateReferenceFromObject(superf)
Dim Thick As ThickSurface
Set Thick = hpshapefactory.AddNewThickSurface(reft, 1, espesor / 2, espesor / 2)
part1.Update
```

Código 34. Thick Surface

La herramienta "*AddNewThickSurface*" pide como argumentos la superficie a la cual se le quiere aportar espesor, la dirección en la que se quiere aplicar el mismo (la cual puede ir hacia dentro de la superficie o hacia fuera, lo cual se representa con un 1 o un -1, de manera análoga a la opción "reverse direction" que se puede observar en el interfaz de dicha herramienta), y finalmente el espesor a aumentar para cada dirección normal de la superficie.

5 Aplicación: entorno programación.

En esta sección se van a desarrollar los procedimientos que se han llevado a cabo para realizar el diseño de engranajes.

En primer lugar se va a crear un formulario, en el cual se podrá elegir entre dos tipos de engranajes, los cuales serán rectos o cónicos. También se incluye una tercera aplicación para el desarrollo de ejes nervados. Según el engranaje en cuestión, se requerirá que se introduzcan los datos de partida necesarios.

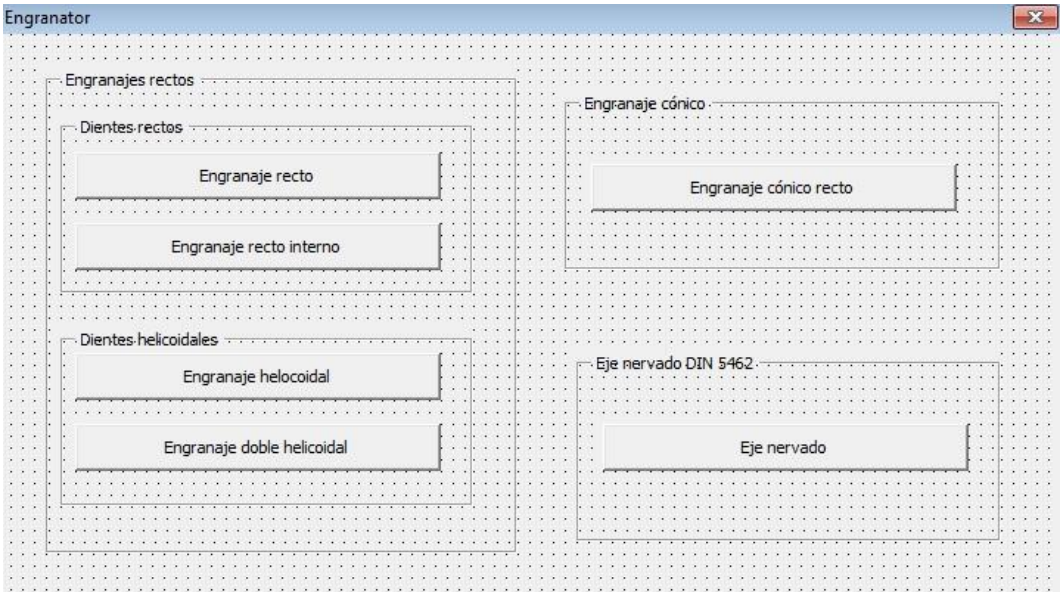

Figura 39. Interfaz del programa

En el formulario se distinguen dos controles principalmente: *Frames* para distinguir entre los distintos tipos de engranajes, y *CommandBotton* para acceder a cada uno de los engranajes.

Cada uno de los *CommandBotton* de este formulario está programado para que cuando se haga clic sobre él, el programa abra automáticamente el formulario correspondiente con el tipo de engranaje en cuestión.

Para ello, se han usado las siguientes líneas de código:

```
Private Sub CommandButton1_Click()
Load NombreUseform
NombreUseform.Show

End Sub
```

Código 35. Nombramiento de Useform con un Command Botton

Habrá un *CommandBotton* y por lo tanto un formulario asociado a cada tipo de engranaje que se va a realizar:

- Engrane recto
- Engranaje recto interno
- Engranaje recto helicoidal
- Engranaje recto doble helicoidal
- Cónico recto
- Eje nervado Din 5462

Una vez definido el interfaz con el que se va a trabajar a lo largo del proyecto, se van a explicar los procedimientos que se han realizado para que dicho programa funcione correctamente.

Lo primero que se va a ilustrar es como se ha diseñado la geometría de los dientes involutos, puesto que será la misma para todos los engranajes que posean dicha geometría de diente salvo por pequeñas modificaciones.

5.1 Geometría de la involuta.

Como se introdujo en el capítulo uno, la geometría más eficiente a la hora de diseñar un engranaje es los dientes de perfiles involutos. A continuación, se va a describir paso a paso el procedimiento que se ha llevado a cabo para programar el módulo que reproduce el *sketch* con geometría de diente involuto.

5.1.1 Geometría de los dientes.

En primer lugar vamos a explicar cómo se han declarado las variables que servirán de argumentos de entrada al módulo, así como los argumentos requeridos.

A la hora de programar un módulo, los argumentos de entrada se pueden declarar como variables globales, o como referencia.

Si las declaramos como referencia, el módulo tomará aquellas variables del programa general que tengan el mismo nombre que las que se pidan como argumentos de entrada, es decir, el módulo solo funcionara para las variables definidas con un nombre en específico.

Sin embargo, si se declaran como variables globales, se dará un nombre a cada variable que el módulo toma como argumento de entrada, definiéndola de una forma diferente a lo hecho normalmente: *Public variable1 as Tipovariable*. Por lo tanto, el módulo trabajará con toda variable que se introduzca en el argumento de entrada asociada a variable1, lo cual nos permite usar el módulo tantas veces como se desee sin tener que cambiar los nombres con los que se definen las variables en su interior. Además las variables definidas como públicas dentro de un módulo se pueden usar en cualquier formulario.

Como se puede observar dicho procedimiento es más eficiente que la definición de los argumentos de entrada como referencia; sin embargo, hay que prestar especial atención en definir correctamente las variables globales, así como que cada argumento de entrada sea del mismo tipo a la variable definida como variable pública, puesto que en caso contrario, se induciría a error.

Una vez explicado esto, vamos a ilustrar como se ha realizado el módulo para el diseño del *sketch* del perfil de los dientes involutos.

Una vez definidas las variables que se van a utilizar en el módulo, la línea de código con la que se empieza a programar el módulo en cuestión es:

```
Public axis2D3 As Axis2D
Public c geometricElements3 As GeometricElements
Public sketch3 As Sketch
Public factory2D3 As Factory2D
Public d1, p, rf, db, f As String """""""""" quitar comentario para engranaje conico
Public Sub dienteh(d1, p, rf, pi, Z, db, h, geometricElements1, axis2D1,
partDocument1, part1, sketch1, originElements1, factory2D1)
```

Código 36.Creación del módulo

Las variables de entrada son parámetros geométricos, así como objetos y herramientas características propias del módulo *sketcher*.

Los parámetros necesarios para definir el engranaje ya se explicaron en el capítulo 1 (la teoría general de engranajes) , y son los que siguen:

```
pi = 4 * Math.Atn(1)
'betha = (pi / 2) / Z " Z=número de dientes
p = pi * d1 / Z "paso circunferencial
m = p / pi
e = pi * m / 2 "" Espesor de un diente curvilineo
rb = d1 / 2
rc = rb * Math.Cos(betha)
A = m 'addendum
B = 1.25 * m 'dedenddum
C = 0.25 * m 'db
```

```
ri = 0.166 * m 'radio de pie/union
h = A + B
‘radios y diametros
db = d1 * Math.Cos(alpha * pi / 180)
df = d1 - 2 * B
rb = db / 2
rf = df / 2
```

Código 37. Parámetros geométricos de un engranaje

Lo primero que se va a hacer es crear un *spline*, cuyos puntos serán calculados con la ecuación paramétrica de la involuta:

$$x = r(cos\theta + \theta sen\theta)$$

$$y = r(sen\theta - \theta cos\theta)$$

Daremos valores al parámetro theta, teniendo cuidado de no dar valores demasiado elevados, puesto que la espiral podría desarrollarse demasiado resultando imposible trabajar con ella.

```
''‘Definimos la primera involuta

theta1 = 0
X1 = rb * (Math.Cos(theta1) + theta1 * Math.Sin(theta1))
Y1 = rb * (Math.Sin(theta1) - theta1 * Math.Cos(theta1))

theta2 = 7 * pi / 180
X2 = rb * (Math.Cos(theta2) + theta2 * Math.Sin(theta2))
Y2 = rb * (Math.Sin(theta2) - theta2 * Math.Cos(theta2))

theta3 = 14 * pi / 180
X3 = rb * (Math.Cos(theta3) + theta3 * Math.Sin(theta3))
Y3 = rb * (Math.Sin(theta3) - theta3 * Math.Cos(theta3))

theta4 = 21 * pi / 180
X4 = rb * (Math.Cos(theta4) + theta4 * Math.Sin(theta4))
Y4 = rb * (Math.Sin(theta4) - theta4 * Math.Cos(theta4))

theta5 = 28 * pi / 180
X5 = rb * (Math.Cos(theta5) + theta5 * Math.Sin(theta5))
Y5 = rb * (Math.Sin(theta5) - theta5 * Math.Cos(theta5))

theta6 = 35 * pi / 180
X6 = rb * (Math.Cos(theta6) + theta6 * Math.Sin(theta6))
Y6 = rb * (Math.Sin(theta6) - theta6 * Math.Cos(theta6))

theta7 = 42 * pi / 180
x7 = rb * (Math.Cos(theta7) + theta7 * Math.Sin(theta7))
y7 = rb * (Math.Sin(theta7) - theta7 * Math.Cos(theta7))

theta8 = 65 * pi / 180  "49  54
x8 = rb * (Math.Cos(theta8) + theta8 * Math.Sin(theta8))
```

```
y8 = rb * (Math.Sin(theta8) - theta8 * Math.Cos(theta8))

X0 = rf + ri
Y0
```

Código 38. Puntos de la envolvente

Como vemos todos los puntos han sido calculados con la ecuación de la involuta menos el primero de ellos, que se ha diseñado estratégicamente entre los radios de fondo y base, puesto que será el punto a partir del cual se hará el redondeo inferior del perfil.

Una vez definidos los puntos, se crea el spline, herramienta propia del objeto factory2D, y que representa el flanco derecho del primer diente que se va a crear.

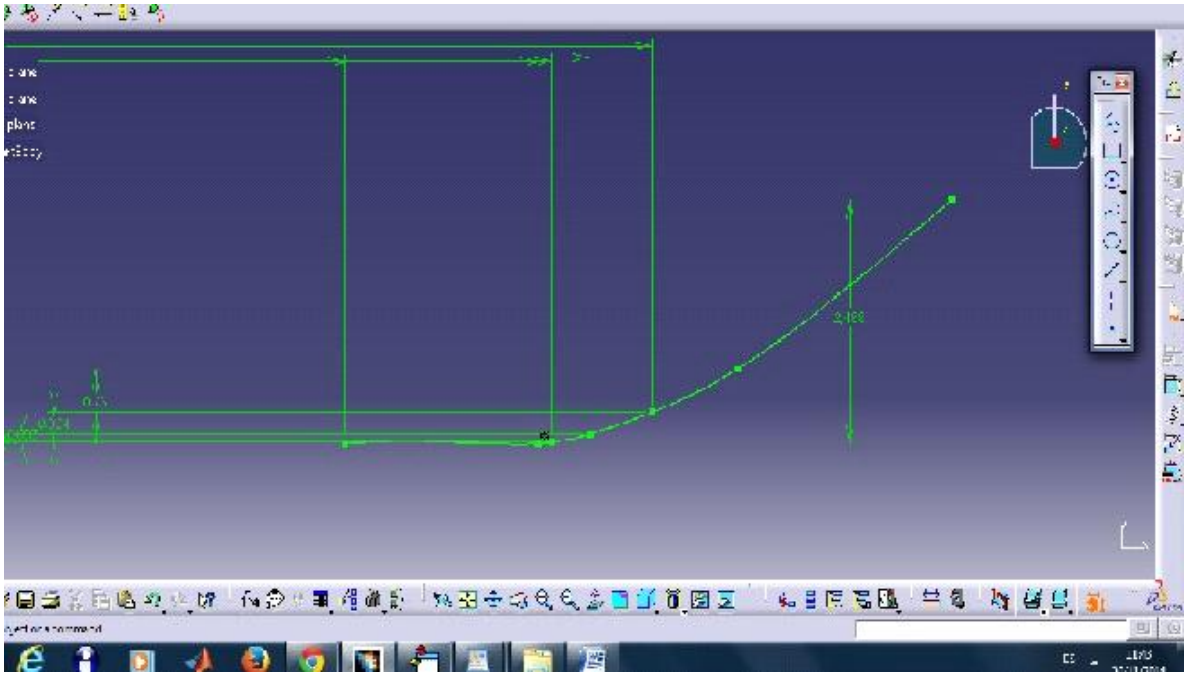

Figura 40. Envolvente 1

Cabe destacar que para crear los engranajes se va " eliminar material" en lugar de "añadir material", por lo que lo que se va a crear no es el perfil del diente en cuestión, sino el hueco que se crea entre dos dientes consecutivos, el cual servirá para eliminarlo posteriormente del bloque de material inicial con las operaciones pertinentes.

Como se vio en el capítulo 1, todas los parámetros y variables de interés están referidas al círculo primitivo del engranaje,"d1", por lo que todas las distancias se

referirán a dicha circunferencia, la cual introducimos en nuestro *sketch* y impondremos que será geometría en construcción.

Se va a crear un arco de circunferencia restringido en los cuadrantes tercero y cuarto ,lo cual es debido a que posteriormente habrá que realizar la intersección entre varias curvas, siendo está una de las implicadas y quedando los puntos resultantes uno en el primer cuadrante y otro en el cuarto, siendo este el punto de interés. Por lo que restringiendo la circunferencia en estos cuadrantes no habrá ambigüedad alguna, y el programa escogerá la solución adecuada.

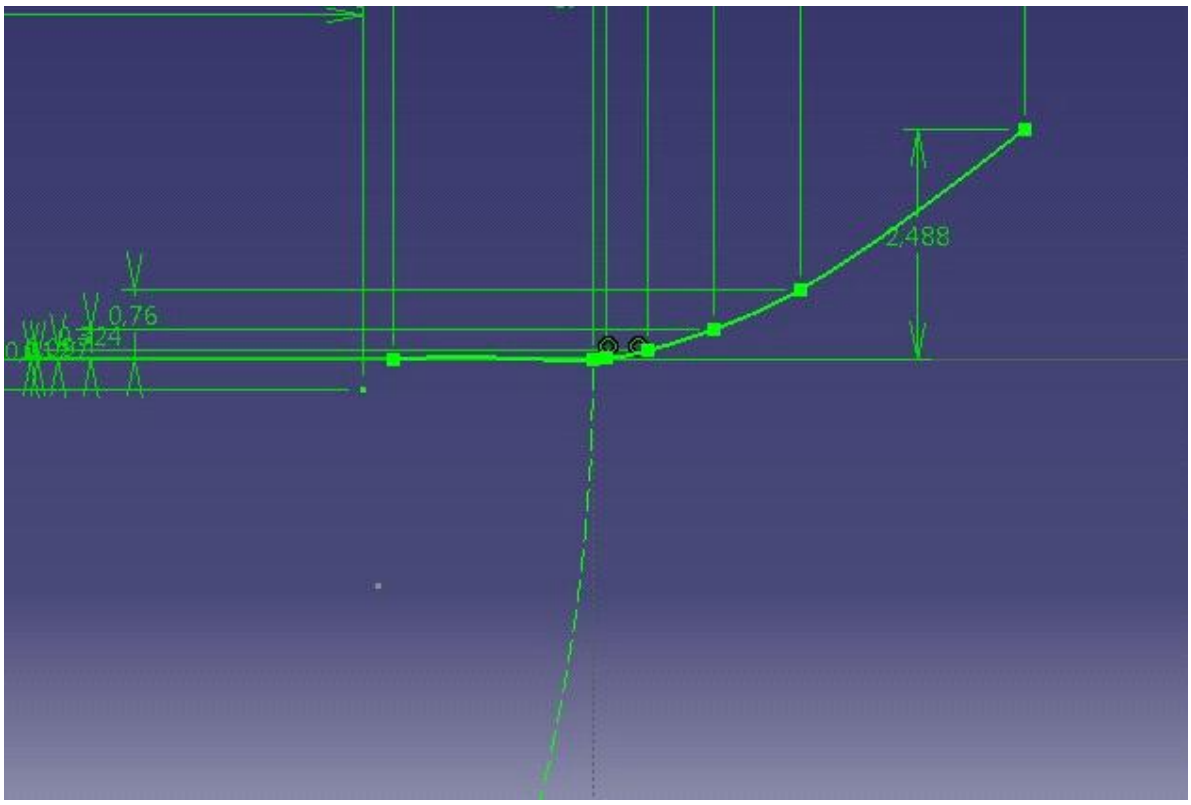

Figura 41. Diámetro primitivo

En la figura 41 se observa que todas las curvas están perfectamente restringidas.

El siguiente paso sería hacer el otro *spline* que reflejaría el flanco izquierdo del diente contiguo al anterior. He aquí el motivo por el que se ha creado la circunferencia primitiva, ya que nos va a servir de ayuda para crear la línea de simetría q nos ayudara a hacer dicha curva.

Como se introdujo al principio de este texto, sobre la circunferencia primitiva se definen los parámetros de mayor interés de un engranaje, como son el paso circular o el módulo. El paso circular no es más que la distancia sobre dicha circunferencia entre un punto, y el mismo punto correspondiente al siguiente diente. Esto es de gran interés

puesto que sobre dicha circunferencia, el espesor del diente es igual a la mitad del paso, es decir, espesor y espacio entre dientes tienen el mismo valor.

Por lo tanto vamos a usar el arco definido anteriormente para crear un punto medio gracias al cual podremos crear un eje de simetría. Para ello vamos a crear una circunferencia de radio igual a la cuerda correspondiente a un arco de circunferencia de valor la mitad del paso (que es igual a la mitad del espesor del diente) y de diámetro el diámetro primitivo. Dicha circunferencia tendrá por centro la intersección entre arco y el *spline* creados. La intersección de dicha circunferencia con el arco será un punto, el cual se corresponderá con el punto medio del perfil del diente a una distancia d1 del centro del engranaje.

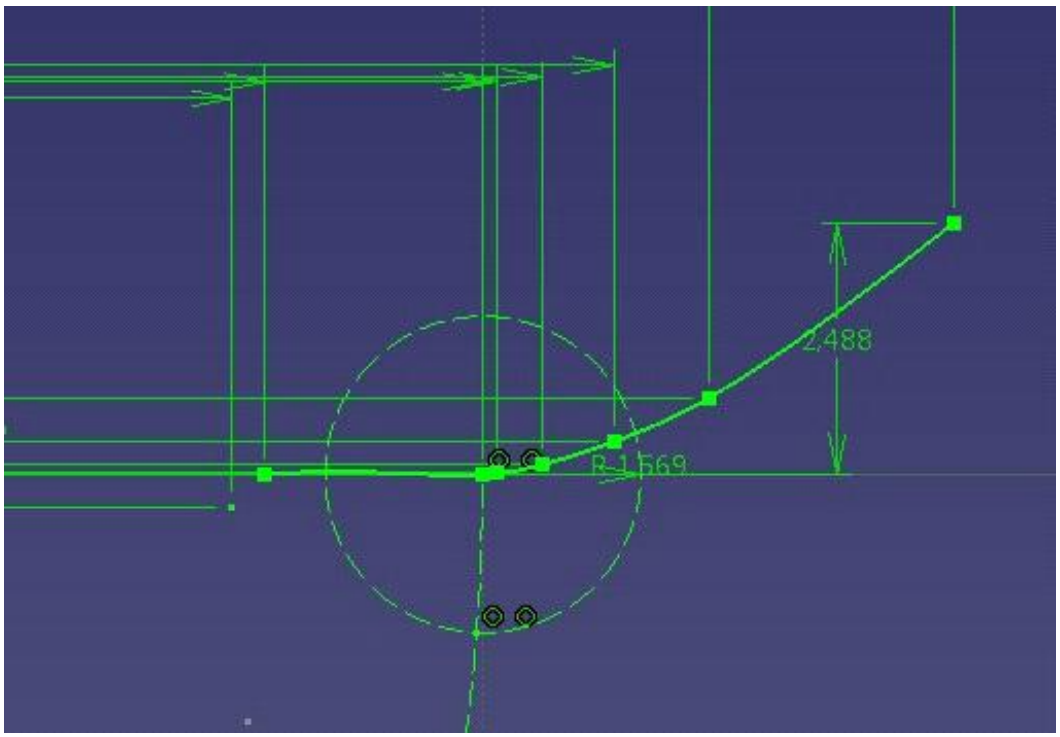

Figura 42. Creación de círculo auxiliar

Finalmente, hacemos una línea que una el origen de coordenadas y el punto creado por la intersección de las curvas implicadas que servirá como eje de simetría para la involuta creada con el *spline1*.

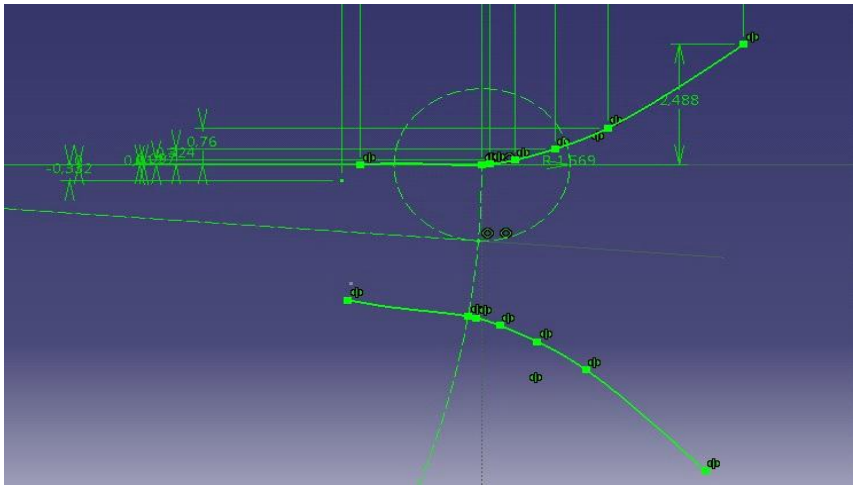

Figura 43. Creación de eje de simetría y realización de la simetría

Para finalizar el *sketch* solo resta cerrar el boceto. Para cerrar la parte superior se pueden seguir dos caminos: o bien creamos un arco de circunferencia que una los dos puntos finales de los *spline* 1 y 2 , o una recta.

Para hacerlo con un arco de circunferencia, habría que calcular el radio de la circunferencia con la ecuación característica:

$$x^2 + y^2 = r^2;$$

donde solo habría que introducir las coordenadas X y Y del último punto del *spline* y despejar el radio.

Sin embargo, aunque en primera estancia parece más correcto cerrar la curva con un arco de circunferencia, a la hora de realizar operaciones 3D más complejas, la geometría que se origina es muy crítica, induciendo errores en el programa que es incapaz de realizar ciertas operaciones. Por ello se va a crear una recta que una dichos dos puntos, quedando el diseño perfectamente cerrado en su parte superior.

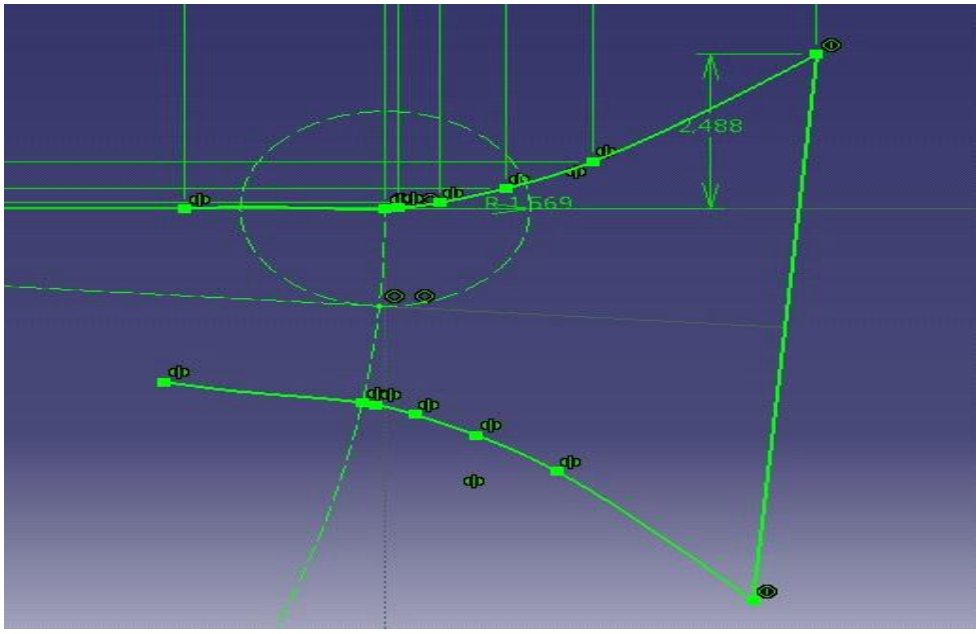

Figura 44. Unión de los últimos puntos de las envolventes

Finalmente se tiene que cerrar el sketch por la parte inferior del diente, es decir en el diámetro de fondo, teniendo en cuenta que la unión de los flancos con el diámetro de fondo no es recta, sino que existen dos suavizados de radio dados por la características geométricas del engranaje.

Por lo que, dado el radio de suavizado ri, tenemos que calcular tres puntos: el centro del arco de circunferencia de radio ri, el punto de inicio, y el punto final. Las coordenadas de dichos puntos serán:

```
‘Punto de inicio
'X0,y0
‘Punto final
Xri = rf
Yri = -ri
‘Centro de la circunferencia
Xri2 = rf + ri
Yri2 = -ri
```

Código 39. Puntos para la construcción de los arcos inferiores

El arco queda :

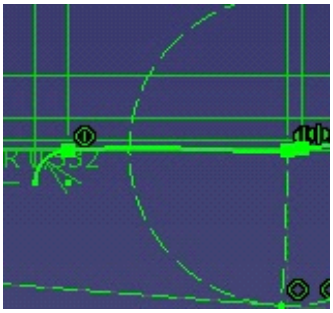

Figura 45. Arco inferior

Una vez realizado el suavizado el flanco superior, y de igual forma que con el *spline1*, hacemos el arco simétrico respecto al eje de simetría y finalmente, cerramos el boceto con un arco de radio el radio de fondo, y cuyos puntos inicial y final coinciden con los puntos de fin e inicio de los arcos creados anteriormente.

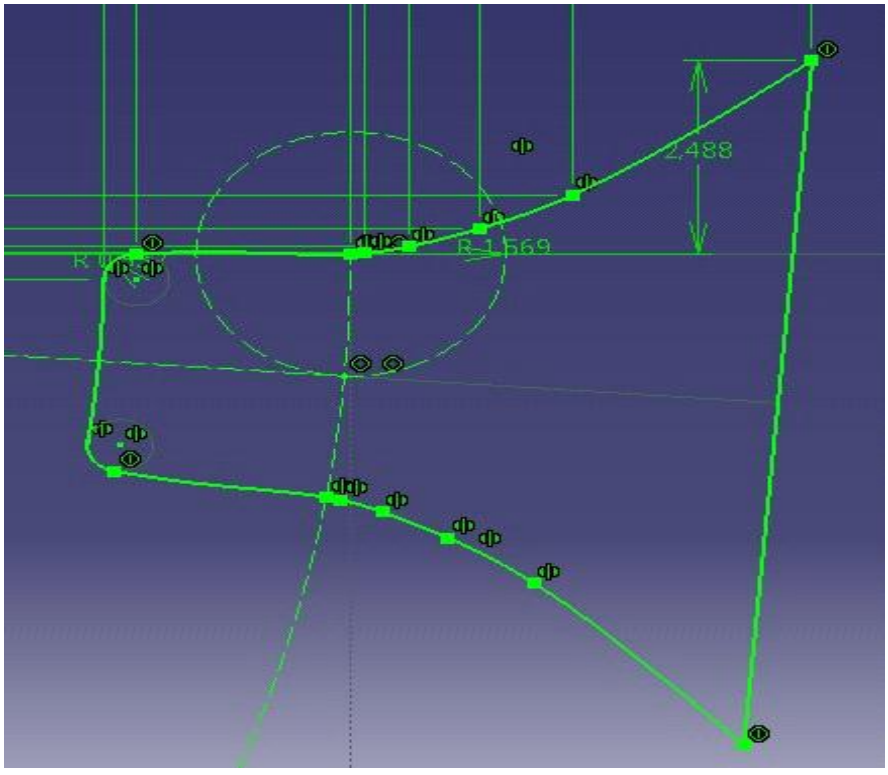

Figura 46. Sketch del hueco entre dientes

Se puede observar que el *sketch* queda perfectamente restringido (restricciones que se explicarán posteriormente) y que se obtiene un perfil perfectamente cerrado, lo cual se puede comprobar manualmente haciendo un análisis del *sketch* realizado desde CATIA(*Tools-->sketch analysis*).

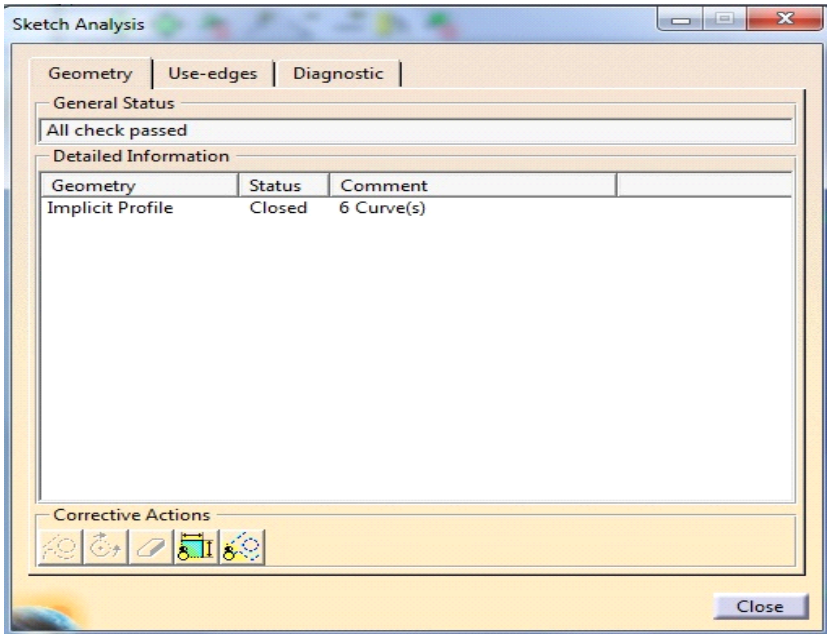

Figura 47. Sketch Analysis

Una vez realizado el *sketch* que dará la geometría del perfil de los dientes, el trabajo que se realizará en el módulo *sketcher* es prácticamente despreciable, y se limitará a crear las circunferencias correspondientes con el diámetro exterior del engranaje, y la circunferencia correspondiente con el eje central del engranaje.

A continuación se va a explicar el proceso de programación que se ha llevado a cabo para diseñar los diferentes engranajes que se plantean en este proyecto.

5.2 Engranaje cilíndrico recto de dientes involutos.

5.2.1 Engranajes cilíndricos.

Para desarrollar un engranaje cilíndrico necesitamos como datos de partida al menos 3 de 4 de los parámetros característicos de la geometría de los engranajes: diámetro primitivo (d1), módulo (m), número de dientes (Z) y ángulo de presión. Tanto d1,m,como Z están relacionados entre sí, de manera que solo tenemos que elegir dos de ellos. Normalmente se suelen usar el módulo y el número de dientes como parámetros, y d1 se obtiene mediante relaciones geométricas.

- **Engranaje cilíndrico recto de dientes involutos.**

El interfaz asociado al engranaje cilíndrico recto es:

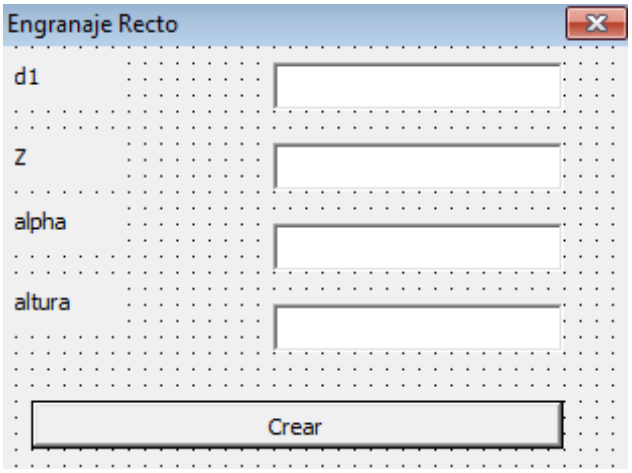

Figura 48. Formulario de engranaje recto

Como se dijo previamente, los argumentos de entrada son m,z y alpha. Lo primero que se va a realizar es un *sketch* en el que creemos una circunferencia de diámetro el diámetro exterior del engranaje, el cual como ya es sabido es de=rf+A+B.

Mediante el objeto shapefactory1 se añade un nuevo *pad* de altura la altura del engranaje, la cual viene dada por altura=10*m.

Ya se tiene el bloque de material macizo inicial sobre el cual se trabajará para todos los tipos de engranajes cilíndricos que se van a diseñar. A continuación se crea un nuevo *sketch*. En este sketch 2 se realiza la llamada a la función creada anteriormente que origina la geometría del perfil.

Sin embargo dicho sketch tiene una diferencia respecto a lo explicado en el manual de programación. En esta ocasión no se va a definir la referencia como un plano XY cualquiera sino que vamos a crear un plano con un offset de valor la altura la del engranaje en cuestión en el que se insertará el sketch.

Para realizar la llamada al módulo se tiene que escribir la siguiente línea de código:

```
Call diente.diente(d1, p, rf, pi, Z, db, h, geometricelements2, axis2d2, partDocument1, part1, sketch2, originElements1, factory2d2)
```

Código 40. Llamada del módulo

Se observa que para efectuar la llamada del módulo es necesario tanto el nombre del módulo en la columna de módulos y formularios, como la denominación de la *Public Sub* y es por ello que es necesario escribir “diente.diente”.

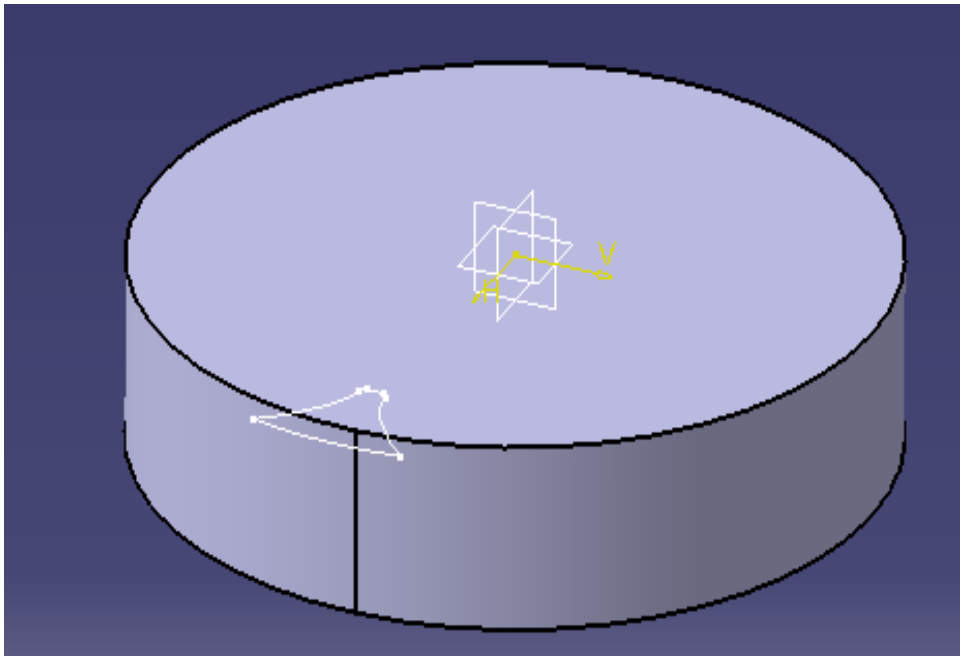

Figura 49. Engranaje recto (paso 1)

A continuación se va a proceder a eliminar material para obtener el hueco entre dientes consecutivos, procedimiento parecido al que realizan las fresadoras en la realidad.

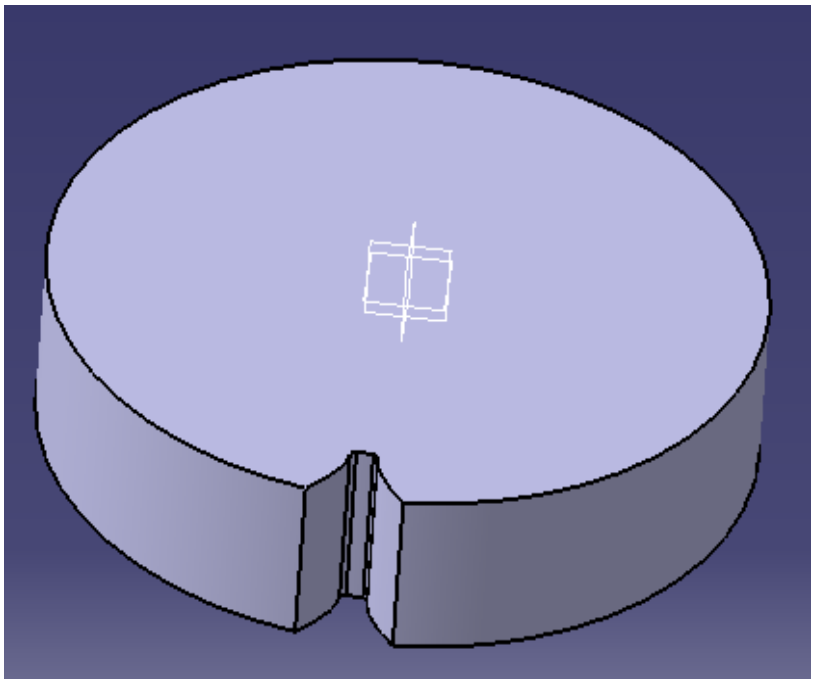

Figura 50. Engranaje recto (paso 2)

Una vez obtenido el hueco entre dientes, se realizará un patrón circular en torno al eje Z, donde el número de repeticiones será el número de dientes y el ángulo entre copias será el ángulo asociado al paso circular, el cual es esp = 2 * p / d1 * 180 / pi. El resultado final es:

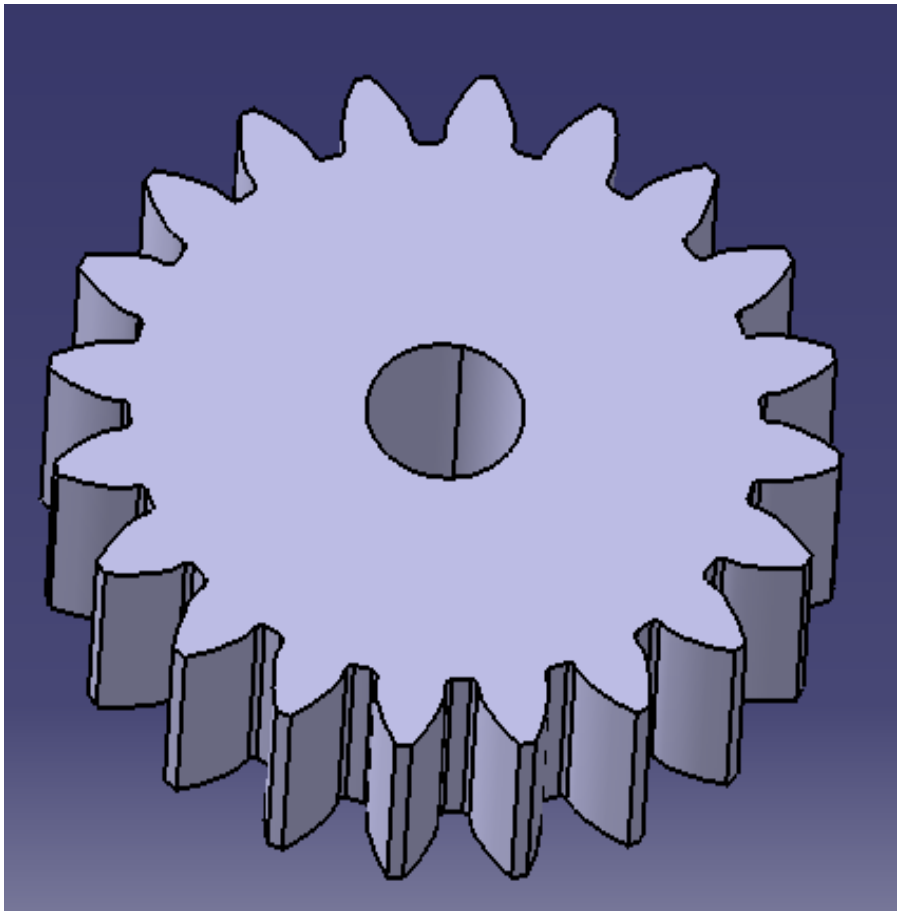

Figura 51. Engranaje recto (final)

donde se ha añadido un *pocket* central por el cual pasará el eje de transmisión del engranaje.

- **Engranaje cilíndrico helicoidal de dientes involutos.**

Se va a desarrollar ahora un engranaje cilíndrico de dientes helicoidales u oblicuos. Aunque la geometría de este tipo de diente cambia un poco respecto a los engranajes rectos, para simplificar el diseño se va a usar el mismo perfil involuto que en el caso de los engranajes cilíndricos de dientes rectos, salvo por el hecho de que los dientes no son rectos, si no que siguen la forma de una hélice a lo largo de su espesor.

El interfaz que da lugar a dichos engranajes es:

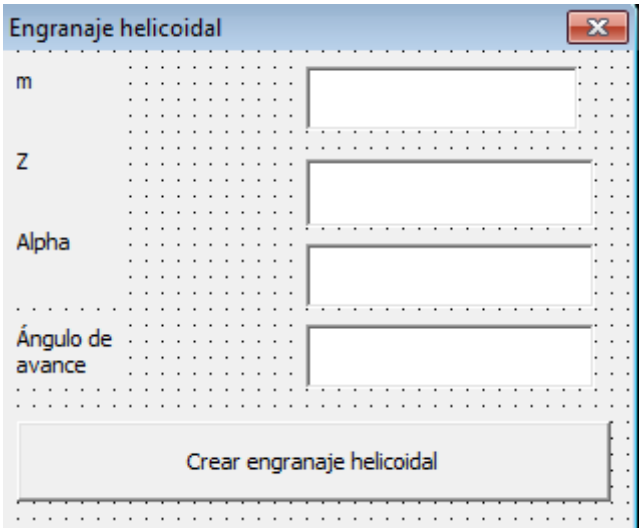

Figura 52. Formulario engranaje helicoidal

donde los datos de partida son el módulo, número de dientes, ángulo de presión y ángulo de avance.

El procedimiento para realizar dicho engranaje es similar al engranaje anterior. En primer lugar hacemos un *sketch* (*sketch1*) donde se realiza una circunferencia de diámetro de (diámetro exterior). Se realiza un *pad* (*pad1*) de espesor igual la altura del engranaje, y a continuación se realiza otro *sketch* en la cara superior del cilindro dónde se llama al módulo diente.

La principal diferencia de este engranaje con el engranaje anterior es que en este caso para realizar los dientes no se podrá utilizar la operación *pocket* para eliminar material, ya que dicho proceso de retirada de material sigue una determinada guía .

Por ello, es necesario crear en primer lugar la guía que va a seguir el perfil del *sketch2* .

Es sabido, como ya se explicó en el capítulo1, que los engranajes helicoidales tienen el dentado oblicuo con respecto al diente de rotación, es decir, la curva guía de los dientes forma un ángulo (ángulo de paso) con respecto al eje axial.

Se puede plantear el vaciado de material con varias operaciones, como son *slo y remove multisection solid* del módulo *PartDesign*, o realizar la operación en dos fases, una primera añadiendo material con alguna operación como *rib, multisection sólid (* Del módulo *Part Design o WireframeAndSurfaceDesign*), y la segunda en la que podemos eliminar el material añadido con operaciones booleanas.

Tras haberlo realizado de varias formas, se comprueba que la más rápida y eficiente es realizar un *remove multisection (loft)*, y por lo tanto es la que se va a mostrar a continuación.

Para realizar un *remove multisection* necesitamos al menos dos perfiles como secciones.

Una de las secciones es el *sketch2* creado anteriormente, y la otra, se obtendrá rotando un *sketch2* cierto ángulo, obtenido según las ecuaciones (4.1) a (4.3) del capítulo 1.

En primer lugar se crea un planoXY en la base del cilindro, en el que se realizará un *sketch*. Dicho *sketch* será idéntico al *sketch2*, es decir se realiza una llamada al módulo dienteh2 .

A continuación se rotará dicho *sketch3* el ángulo gamma y se ocultará el sketch3, para lo cual se tiene que usar las siguientes líneas de comando:

```
DimhybridshapeRotate1 As HybridShapeRotate
Set hybridShapeRotate1 = hshapefactory1.AddNewEmptyRotate()
Dim referencee2 As Reference
Set referencee2 = part1.CreateReferenceFromObject(sketch2)
hybridShapeRotate1.ElemToRotate = referencee2
hybridShapeRotate1.VolumeResult = False
hybridShapeRotate1.RotationType = 0
Dim referencee3 As Reference
Set referencee3 = part1.CreateReferenceFromObject(axis2D3)
hybridShapeRotate1.Axis = referencee3
hybridShapeRotate1.AngleValue = giro * 180 / pi
body1.InsertHybridShape hybridShapeRotate1
part1.InWorkObject = hybridShapeRotate1
part1.Update

""" borro el sketch 3
Dim selection1 As Selection
Set selection1 = partDocument1.Selection
Dim visPropertySet1 As VisPropertySet
Set visPropertySet1 = selection1.VisProperties
Dim bSTR1 As String
bSTR1 = sketch1.Name
selection1.Add sketch2
Set visPropertySet1 = visPropertySet1.Parent
Dim bSTR2 As String
bSTR2 = visPropertySet1.Nam
Dim bSTR3 As String
bSTR3 = visPropertySet1.Name
visPropertySet1.SetShow 1
selection1.Clear
part1.Update
```

Código 41. Rotación del sketch

Y se obtiene:

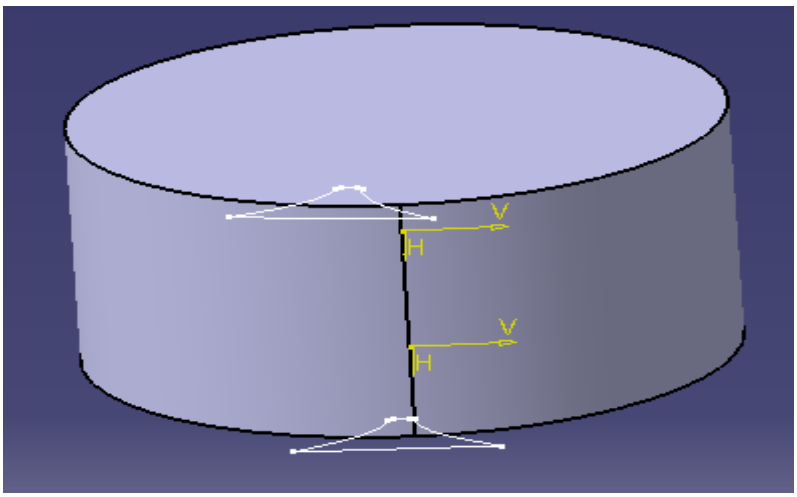

Figura 53.Secciones para el loft

Para añadir la guía se pueden seguir dos caminos, o bien se hace un plano que pase por tres puntos, realizando ahí el sketch en cuestión, o bien un spline3D a partir de tres puntos 3D creados previamente.

En esta ocasión se va a utilizar el segundo método para ilustrar este nuevo procedimiento. Para ello, se programan las siguientes líneas de código:

```
        Xgg1 = X0
        Ygg1 = Y0
        Xgg2 = (rf + ri) * Math.Cos(giro / 2)
        Ygg2 = (rf + ri) * Math.Sin(giro / 2)
        Xgg3 = (rf + ri) * Math.Cos(giro)
        Ygg3 = (rf + ri) * Math.Sin(giro)


        Dim hybridShapePointCoord11 As HybridShapePointCoord
        Set hybridShapePointCoord11 = hshapefactory1.AddNewPointCoord(Xgg1, Ygg1,
Zg1)


        body1.InsertHybridShape hybridShapePointCoord11

        part1.InWorkObject = hybridShapePointCoord11

        part1.Update

        Set hybridShapePointCoord22 As HybridShapePointCoord
        Set hybridShapePointCoord22 = hshapefactory1.AddNewPointCoord(Xgg2, Ygg2,
Zg2)

        body1.InsertHybridShape hybridShapePointCoord22
```

```
        part1.InWorkObject = hybridShapePointCoord22

        part1.Update

        Set hybridShapePointCoord33 As HybridShapePointCoord
        Set hybridShapePointCoord33 = hshapefactory1.AddNewPointCoord(Xgg3, Ygg3,
Zg3)

        body1.InsertHybridShape hybridShapePointCoord33

        part1.InWorkObject = hybridShapePointCoord33


        part1.Update

        Set hybridShapeSpline11 As HybridShapeSpline
        Set hybridShapeSpline11 = hshapefactory1.AddNewSpline()

        hybridShapeSpline11.SetSplineType 0

        hybridShapeSpline11.SetClosing 0 '

        Set refer11 As Reference
        Set refer11 = part1.CreateReferenceFromObject(hybridShapePointCoord11)

        hybridShapeSpline11.AddPointWithConstraintExplicit refer11, Nothing, -1#, 1,
Nothing, 0#

        Set refer22 As Reference
        Set refer22 = part1.CreateReferenceFromObject(hybridShapePointCoord22)

        hybridShapeSpline11.AddPointWithConstraintExplicit refer22, Nothing, -1#, 1,
Nothing, 0#

        Set refer33 As Reference
        Set refer33 = part1.CreateReferenceFromObject(hybridShapePointCoord33)

        hybridShapeSpline11.AddPointWithConstraintExplicit refer33, Nothing, -1#, 1,
Nothing, 0#

        body1.InsertHybridShape hybridShapeSpline11

        part1.InWorkObject = hybridShapeSpline11

        part1.Update
```

Código 42.Creación de línea 3D

Y el spline queda:

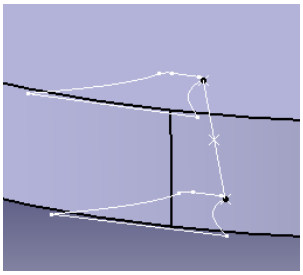

Figura 54. Guía opcional para el loft (Remove multisection)

Con la ayuda de la herramienta *loft* del objeto *shapefactory*, se elimina el material, y finalmente, se realiza el patrón circular y el *pocket* para el paso del eje central, obteniendo:

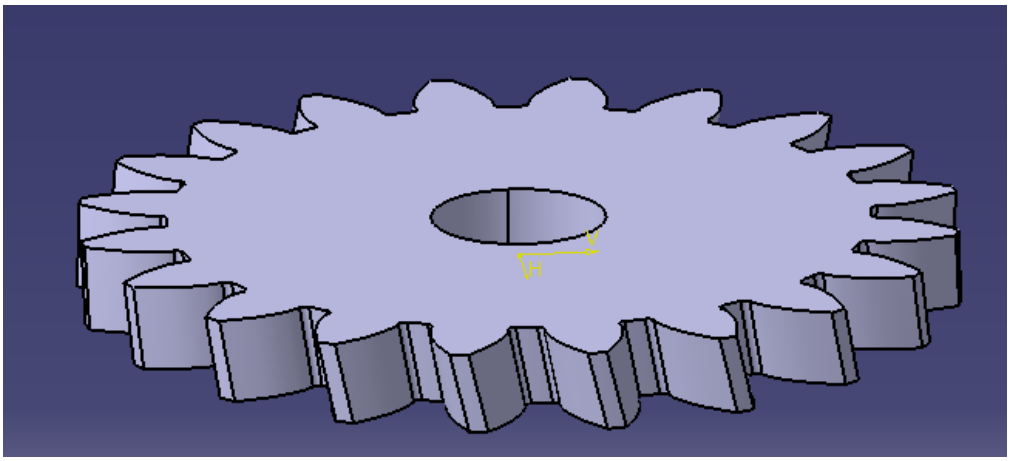

Figura 55. Engranaje helicoidal.

Es sabido que la forma de los dientes de un engranaje helicoidal se aproxima a una hélice, sin embargo en la figura no se aprecia dicha forma. Ello se debe o bien a que solo hemos definido dos secciones para realizar la operación, por lo que CATIA los aproxima linealmente con una recta, o bien a que el espesor del engranaje no se lo suficientemente grande como para apreciarlo. Si volvemos a realizar el engranaje aumentando el espesor y introduciendo tres secciones para realizar el *Remove Multisection-solid* se puede observar el perfil del diente experimenta cierta curvatura aproximándose a una hélice.

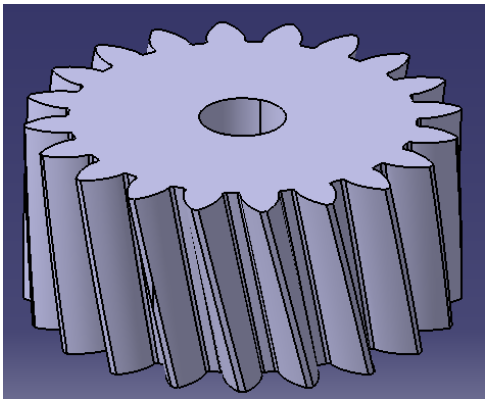

Figura 56. Engranaje helicoidal (2)

- **Engranaje doble helicoidal.**

El procedimiento es prácticamente igual al del engranaje cilíndrico helicoidal salvo por el hecho de que la variable altura con la que se trabaja durante prácticamente todo el código es la mitad de la altura correspondiente al engranaje cilíndrico helicoidal de mismos parámetros.

Se realizarán todas las operaciones necesarias para obtener un engranaje cilíndrico helicoidal de altura la mitad de la que le correspondería realmente (por lo que el código será idéntico al del subapartado anterior), añadiendo solo la operación *mirror* respecto a la cara superior del engranaje (planoXY), la cual nos aporta una simetría dando lugar a un dentado en forma de cola de pescado o "v".

Las líneas de comando que hay que añadir son:

```
        Dim referencemirror1 As Reference
        Setreferencemirror1=
part1.CreateReferenceFromBRepName("FSur:(Face:(Brp:(Pad.1;1);None:();Cf11:());WithTem
poraryBody;WithoutBuildError;WithInitialFeatureSupport;MonoFond;MFBRepVersion_CXR
15)", pad1)
        Dim mirror1 As Mirror
        Set mirror1 = shapeFactory1.AddNewMirror(referencemirror1)
        part1.Update
```

Código 43.Mirror

Y finalmente se obtiene:

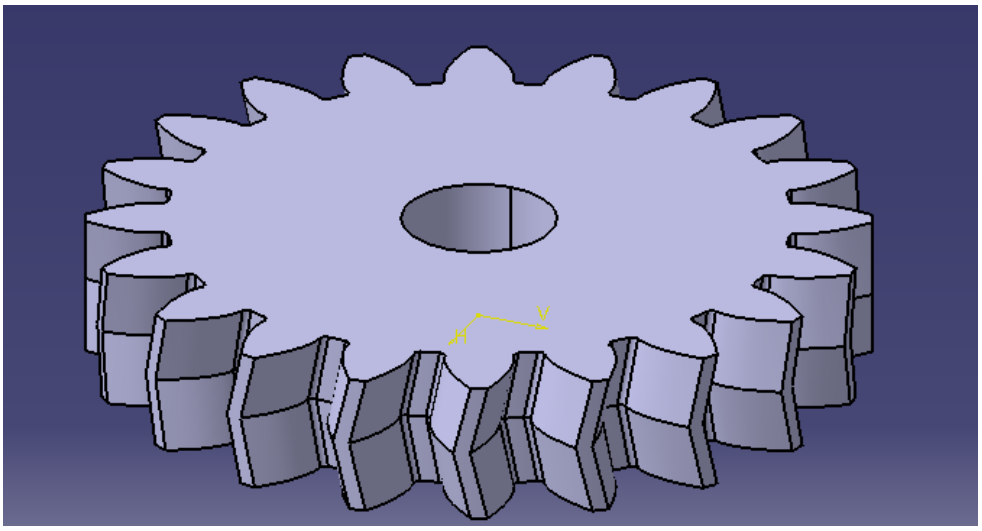

Figura 57. Engranaje helicoidal doble

- **Engranaje cilíndrico recto interno.**

La realización de un engranaje recto interno se puede obtener fácilmente a partir de un engrane recto exterior añadiendo solo algunas operaciones al programa realizado en la subsección 5.2.1.1.

La programación del engranaje interno se va a plantear a partir de operaciones booleanas, en particular la operación *remove*. Es decir, se va a crear un segundo sólido al que se le eliminará el engranaje recto exterior. Para ello es necesario insertar un *body2* (y crear *sketches2* por lo tanto) en el que trabajar y modelar el segundo sólido:

```
Dim body2 As Body
Set body2 = bodies1.Add()
Part1.update
Dim sketches2 As Sketches
Set sketches2 = body2.Sketches
```

Código 44. Definición del body2

En dicho nuevo *body* se creará un cilindro de diámetro mayor al diámetro exterior del engranaje, pero de igual altura.

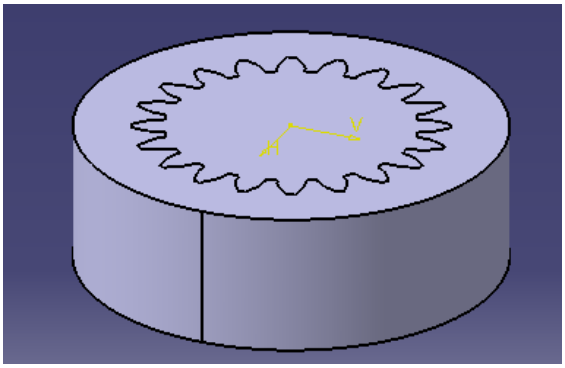

Figura 58. Creación del Body2

Una vez creado, ya podemos realizar la operación booleana entre los dos *body* existentes, donde hay que prestar especial interés en que *body* es el de trabajo, es decir, cual es al que se le elimina material.

```
part1.InWorkObject = body2
 Dim remove1 As Remove
Set remove1 = shapeFactory1.AddNewRemove(body1)
 part1.Update
```

Código 45. Operación booleana.Remove.

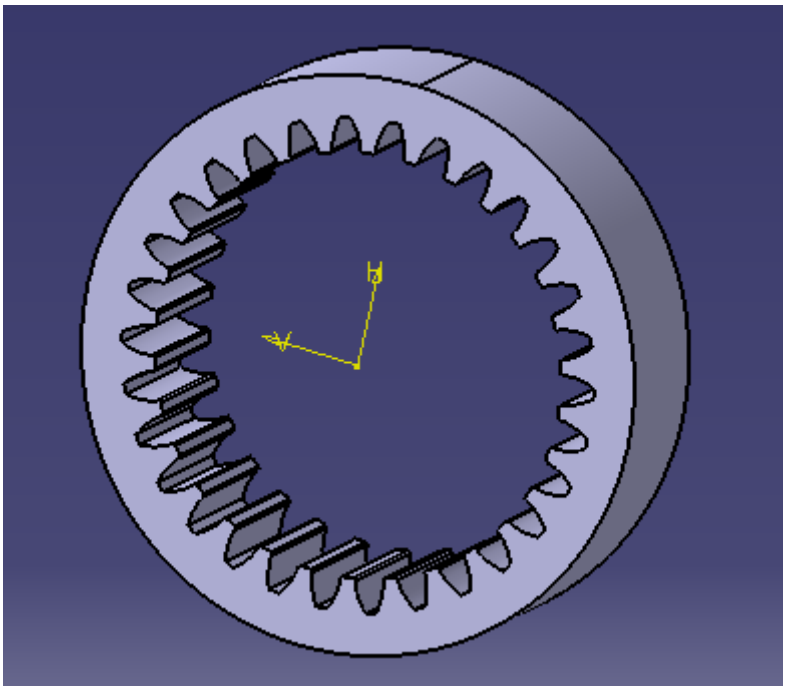

Figura 59. Engranaje recto interior

5.2.2 Otras aplicaciones.

Con las herramientas de programación que se han aprendido a lo largo del proyecto el lector debería ser capaz de programar engranajes y otros elementos mecánicos a la perfección.

En este apartado, se van a usar los conocimientos aprendidos para desarrollar un eje nervado, que podría servir de eje central a los engranajes desarrollados.

Los ejes acanalados o ejes nervados se utilizan cuando la potencia que se transmite es importante. Los ejes acanalados son el resultado de realizar unas ranuras sobre un eje, dando lugar a los nervios que cumplen la misma función que las chavetas.

Se va a programar un eje nervado de dientes rectos, los cuales son aptos para transmitir grandes pares, pero no son aptos para grandes velocidades de rotación. Sus dimensiones vienen definidas según las normas DIN 5461 , DIN 5462 (serie ligera),DIN 5463 (serie media) y DIN 5464 (serie pesada).

Se va a mostrar a continuación como se ha realizado en el eje nervado DIN5461. La geometría del eje viene dada por los datos de la figura 60 dónde como se ve, aparecen 8 datos de entrada: d1,d2,d3,b,e,f,g,r.

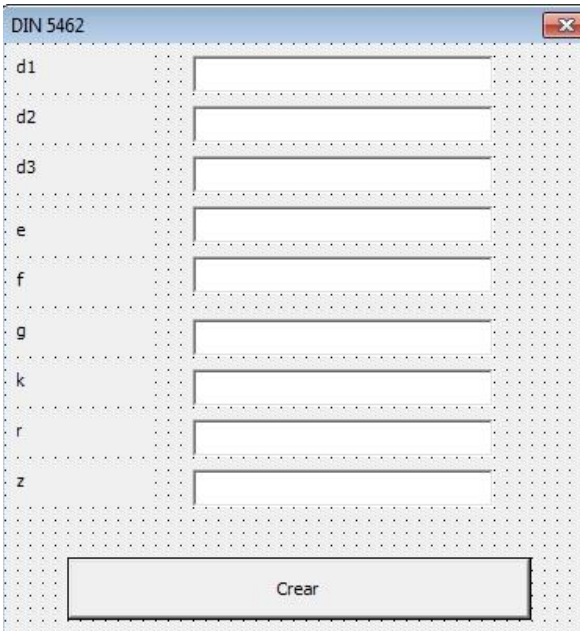

Figura 60. Parámetros del eje enervado

Y los datos de entrada necesarios se tomarán de la tabla dada por la norma DIN 5462 que aparece en la siguiente figura.

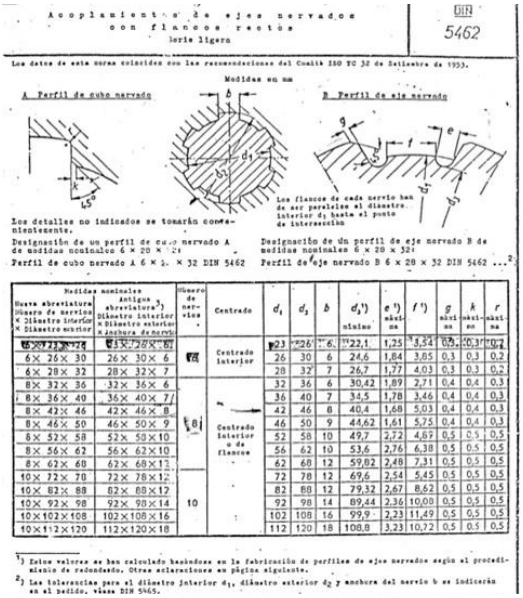

Figura 61. Normativa DIN5462

El primer paso para desarrollar el eje nervado será crear un cilindro de diámetro d2, para lo cual se realiza un *sketch* (*sketch1*) con una circunferencia de diámetro d2, y se hace un *pad*.

A continuación, se realiza el perfil del diente en un *sketch* a partir de los datos geométricos necesarios. Cabe destacar que no se ha incluido en este segundo *sketch* el suavizado que se puede observar en los extremos de la base de los nervios del eje.

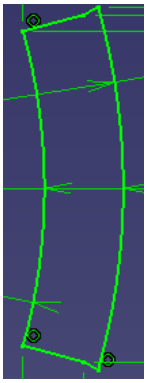

Figura 62.Perfil del nervio del eje

Una vez realizado el *sketch*, eliminamos material del cilindro inicial realizando un *pocket*.

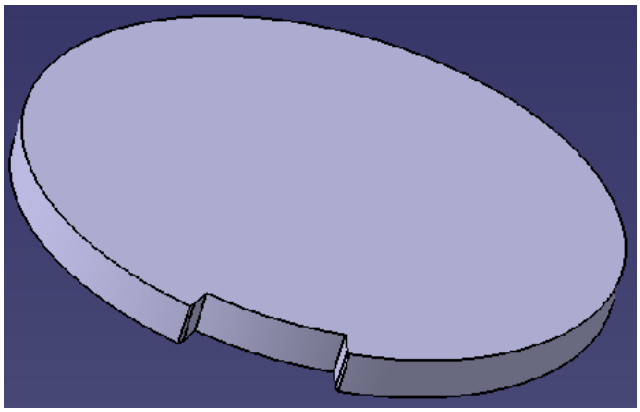

Figura 63. Creación de un nervio

Para terminar el nervio solo falta realizar el suavizado que presenta en sus esquinas inferiores. Para ello vamos a realizar un *sketch* con dos circunferencias de radio r centradas en el punto adecuado, y eliminamos material realizando un *pocket*.

Finalmente realizamos dos patrones circulares con cada uno de los *pockets* que se han realizado .El resultado final es el que se muestra en la figura 61.

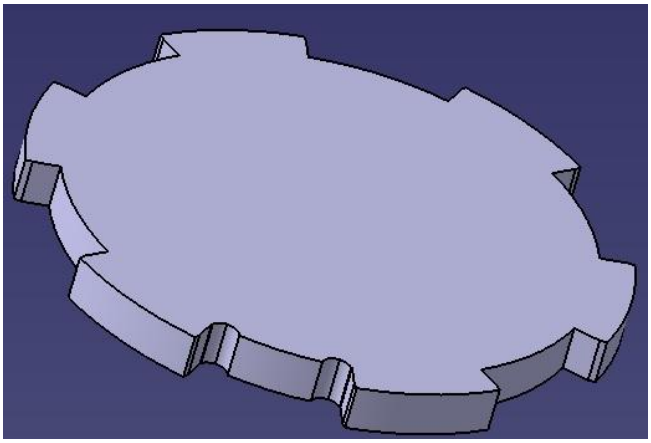

Figura 64.Suavizado del nervio y patrón circular

Una buena aplicación que se podría desarrollar es añadir en un *body (body2)* el procedimiento realizado para el eje nervado en cualquier de los engranajes realizados anteriormente. De esta manera, podríamos eliminar mediante operaciones booleanas el material correspondiente al *body2*, quedando los engranajes con un eje central nervado.

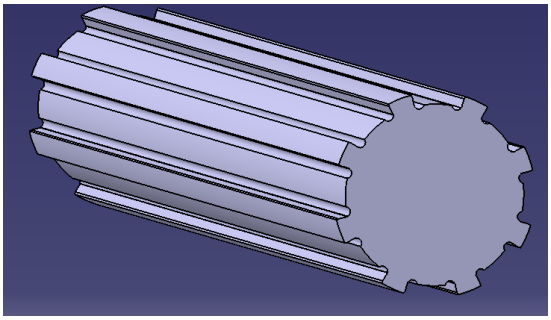

Figura 65.Eje nervado

Figura 66. Bujes nervados,

5.2.3 Aplicación en desarrollo. Engranajes cónicos rectos.

En este proyecto se va a presentar una aplicación para el diseño de engranajes cónicos la cual distinguimos de las demás puesto que está en vía de desarrollo y no se ha perfeccionado su funcionamiento. Ello se debe a diversos problemas que se han encontrado a la hora de definir la geometría correctamente y a que su funcionamiento no es al cien por cien fiable.

La aplicación diseñará engranajes cónicos de tipo inglete, es decir, engranajes cónicos que tienen el mismo paso diametral o módulo, ángulo de presión y número de dientes.

Las herramientas de programación utilizadas son las mismas que para los engranajes anteriores y el resultado es el que se muestra en las figuras 66 y 67.

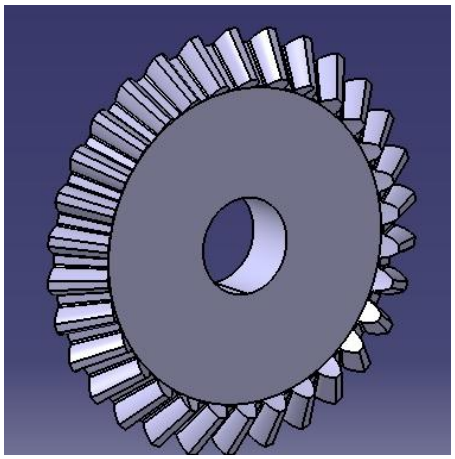

Figura 67.Engranaje cónico (1)

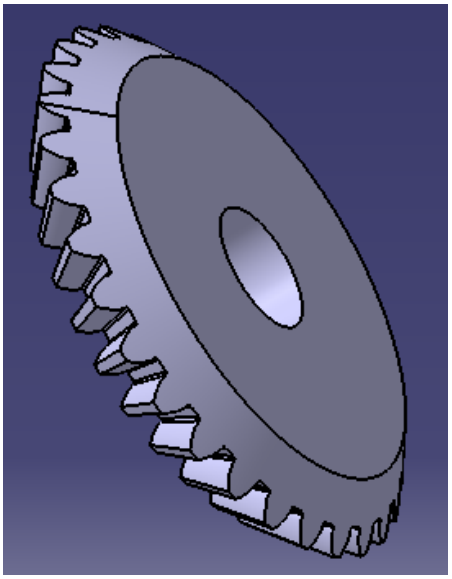

Figura 68.Engranaje cónico (2).

6 Aplicación Engranator : manual de usuario.

Una vez explicado el software desarrollado para realizar engranajes, se va a ilustrar a continuación los pasos que debe seguir el lector para el uso del mismo.

En primer lugar el usuario debe arrancar CATIA V5 . Debe abrir la pestaña Tools de la barra de herramientas, y acceder a macros→Visual Basic editor.

Una vez abierto el interfaz de VBA, para arrancar la macro se debe clicar sobre el formulario Engranator abriéndose la siguiente ventana:

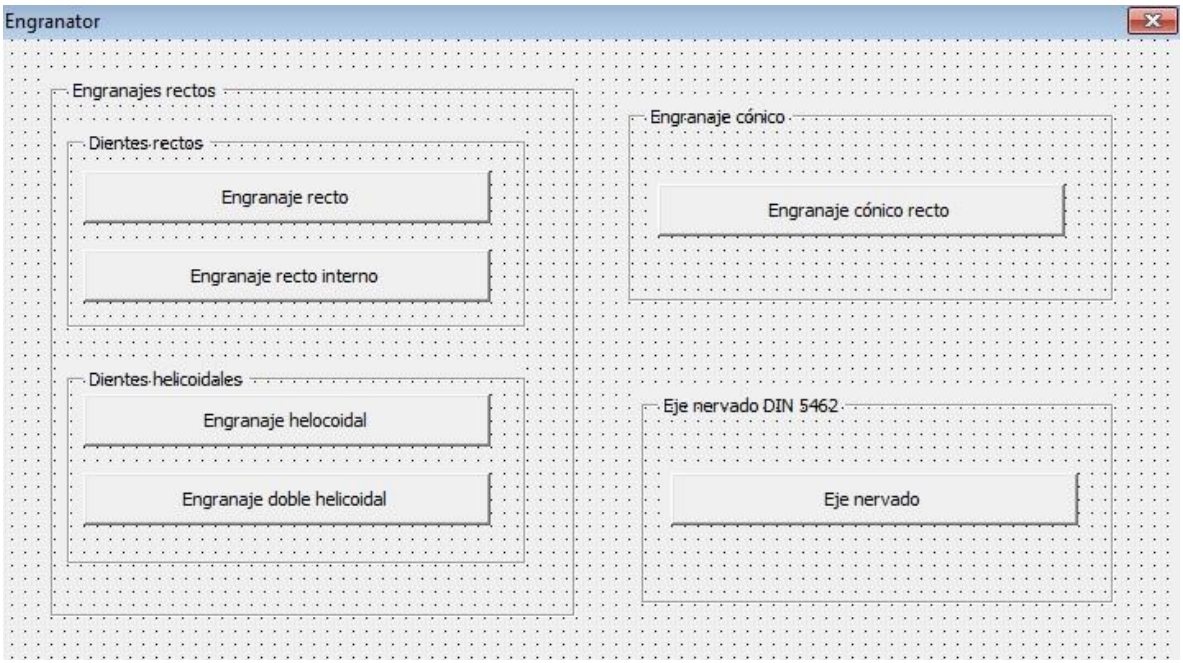

Figura 69. Interfaz de la aplicación Engranator,

Se deberá seleccionar el tipo de engranaje que desea realizar. Para ello se hará clic sobre la pestaña, abriéndose el interfaz correspondiente al engranaje en cuestión. Se van a explicar uno a uno los interfaces correspondientes a cada uno de los engranajes programados, para que el usuario pueda usar el software desarrollado sin ningún problema.

En general, los engranajes vienen definidos por tres parámetros característicos: módulo o diametral pitch y el ángulo de presión. En nuestro caso se va a trabajar con unidades del sistema internacional, por lo que se usará el módulo, el cual viene dado en mm en la mayoría de tablas y documentos aportados por la normativa. Se necesita un parámetro adicional para definir completamente la geometría del engranaje. Para ello se usará el diámetro primitivo o el número de dientes, siendo más útil desde el punto de vista del diseño mecánico de mecanismo el diámetro primitivo.

6.1.1 Engranaje recto.

Para generar un engranaje recto sólo son necesarios cuatro datos de entrada: el diámetro primitivo, módulo, y ángulo de presión. Podemos introducir como cuarto dato de interés la longitud del diente (espesor engranaje), aunque dicho parámetro viene dado por la norma en la mayoría de las ocasiones.

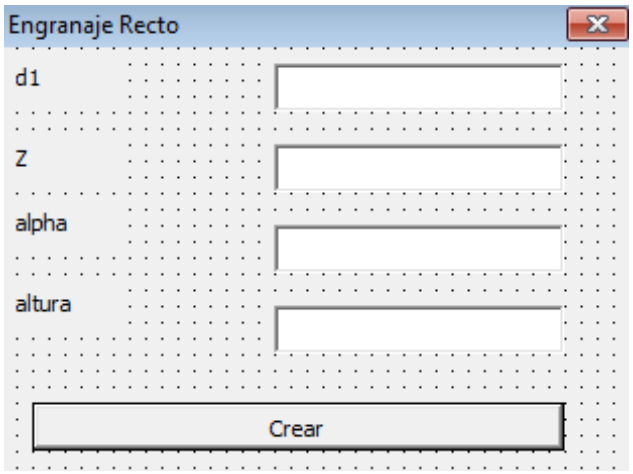

Figura 70.Formulario del engranaje Recto

6.1.2 Engranaje recto interno.

En el caso de un engranaje interior, añadimos un parámetro extra en comparación con el engranaje recto normal, el diámetro exterior.

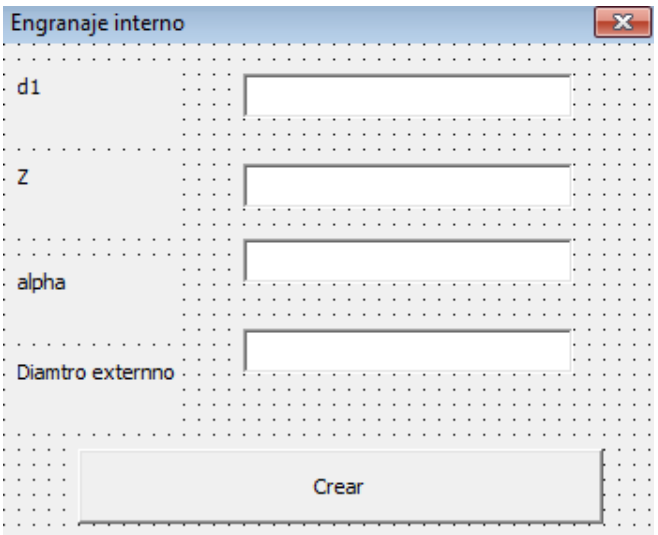

Figura 71. Formulario del engranaje recto interno.

6.1.3 Engranaje helicoidal.

Como es sabido, la característica que diferencia engranajes rectos y helicoidales es que este último posee dientes oblicuos, es decir, que forman un ángulo respecto al eje axial del engranaje. Por lo que el parámetro adicional para definir este tipo de engranaje será el ángulo de avance.

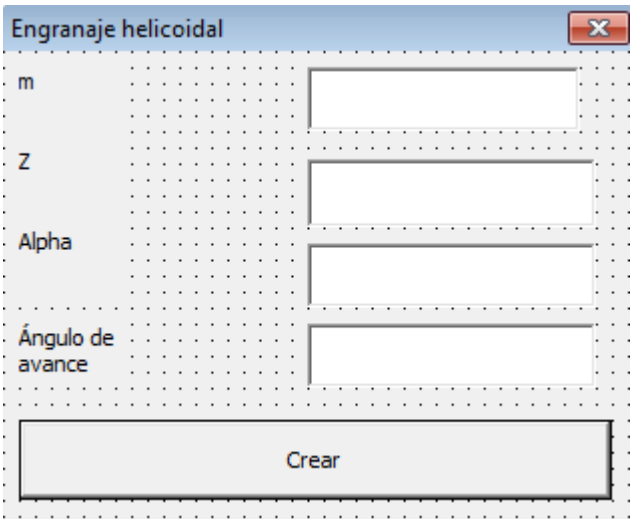

Figura 72.Formulario del engranaje recto helicoidal

6.1.4 Engranaje doble helicoidal.

Como en el caso anterior, los parámetros de entrada del engranaje helicoidal doble son módulo, número de dientes, ángulo de presión y ángulo de avance.

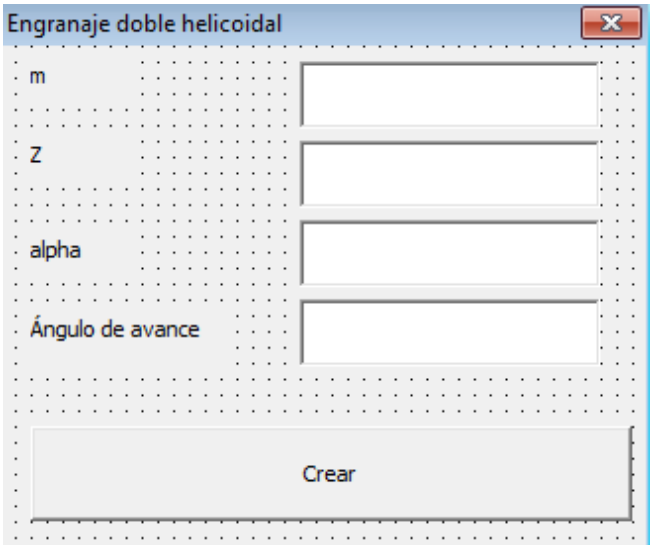

Figura 73. Formulario del engranaje doble helicoidal

6.1.5 Engranaje cónico.

El interfaz que se ha desarrollado hasta el momento para el engranaje cónico recto es :

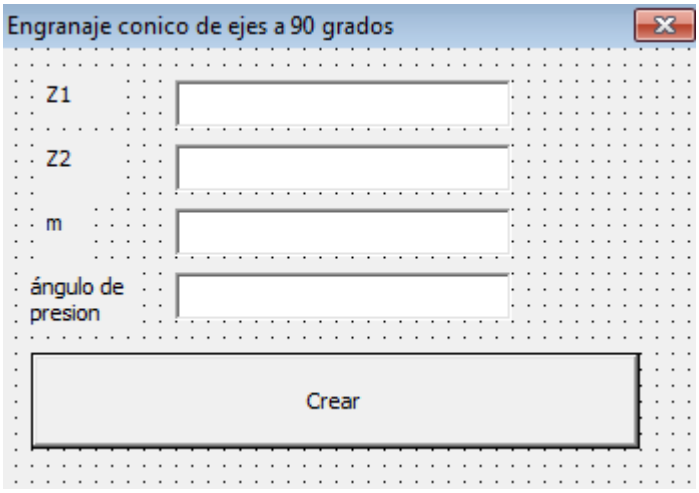

Figura 74.Formulario del engranaje cónico

6.1.6 Eje nervado.

Los parámetros necesarios para el diseño de ejes nervados vienen dado por la normativa DIN 5462, la cual se puede revisar en la figura 61.

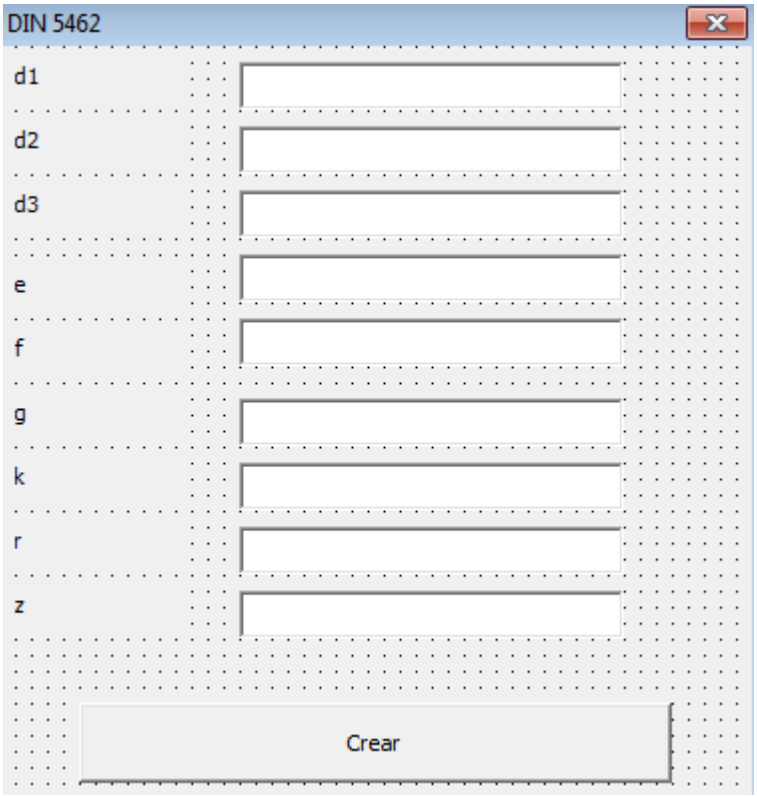

Figura 75. Formulario del eje nervado.

7 Conclusiones

CATIA V5 (Computer Aided three Dimensional Interactive Aplication) es un programa informático de alto nivel que viene siendo utilizado sobre todo en granes y relevantes áreas de la ingeniería como son la aeronáutica y la automoción, ya que permite realizar tanto el diseño como y el análisis de los modelos, y finalmente la fabricación de los mismos.

Sin embargo, a diferencia de otros software CAD/CAM, no posee módulos ni herramientas específicas para el diseño de diferentes piezas y elementos mecánicos. Es por ello que este proyecto se ha querido realizar una introducción al lenguaje de programación Visual Basic for Aplications (VBA), el cual permite crear macros y aplicaciones para CATIAV5.

A la hora de realizar el presente proyecto se pensó que era buena idea realizar una aplicación para CATIA V5 que permitiera realizar algún tipo de pieza o elemento mecánico perteneciente a una familia de piezas, es decir, aquellas que se definen a partir de la similitud de los atributo de diseño y de fabricación de sus miembros. Tras varias deliberaciones y estudiar el amplio abanico de posibilidades existentes, se optó por realizar una aplicación que permitiera realizar familias de engranajes.

La primera aplicación que se realizó fue una que permitiera definir la geometría del diente de perfil involuto, puesto que es la geometría de dientes de engranes más comúnmente usada. En una primera aproximación se decidió generar los engranajes añadiendo material en lugar de eliminándolo, que es el procedimiento que se usa en la industria. Este fue el primer gran error que se cometió en el transcurso del proyecto ya que, transcurrido el tiempo y realizadas aplicaciones para varios tipos distintos de engranajes, se pudo comprobar que a pesar de que la aplicación desarrollada diseñaba los engranajes aparentemente bien, existían errores en el dimensionado del mismo, ya que era prácticamente imposible imponer las restricciones adecuadas sobre la geometría de la involuta para que el dimensionado fuera correcto. A pesar de que los errores eran del orden de un milímetro, se decidió comenzar de nuevo.

Entonces se decidió realizar los engranajes de la misma forma de la que se realiza en la industria, es decir, eliminando material a partir de un cilindro macizo.

Las aplicaciones desarrolladas se centran en los engranajes estándar de ejes paralelos con dientes de perfil involuta, y se tratan tanto engranajes rectos como los helicoidales simples y dobles. También se incluyen otras aplicaciones que están en vías de desarrollo para realizar engranajes cónicos y ejes nervados según diferentes normativas.

En lo referente a la programación en VBA de la aplicación, se han pasado por varias etapas y han tenido lugar varios impedimentos.

En una primera aproximación, como se comentó anteriormente se planteó realizar los engranajes añadiendo material a pesar que por experiencia se sabía que no era buena idea. Una vez rechazada la idea inicial, se comenzó a programar todo de nuevo, encontrándonos con

que algunas operaciones usadas comúnmente en CATIA como la realización de multi-secciones o patrones circulares eran complejas de programar, por lo que hubo que dedicarle mucho tiempo al entendimiento de las mismas y al análisis de los errores que iban apareciendo.

Una vez finalizadas las aplicaciones para el diseño de engranajes rectos y helicoidales, se planteó la posibilidad de realizar una aplicación para generar engranajes cónicos de ejes perpendiculares. Sin embargo, varios problemas a la hora de desarrollar los engranajes acorde a la normativa dieron lugar a la no inclusión de dichos engranajes en el proyecto , dejándolos en un segundo plano como una aplicación en vía de desarrollo.

También se ha realizado un estudio a fondo del módulo de superficies de CATIA para el diseño de engranajes, y a pesar de que se obtienen buenos resultados, es más complejo y tedioso el procedimiento de programación con dicho módulo.

Finalmente lo que se obtiene es una macro para CATIA V5 programada en VBA con la cual a partir de ciertos parámetros iniciales característicos de la geometría de los engranajes, la aplicación es capaz de construir el engranaje que se desea.

8 Anexo A: código de la aplicación.

MODULO DIENTE.

```
Public axis2D3 As Axis2D
Public geometricElements3 As GeometricElements
Public sketch3 As Sketch
Public factory2D3 As Factory2D

Public Sub dienteh2(d1, p, rf, pi, Z, db, h, geometricElements3, axis2D3, partDocument1, part1, sketch3, originElements1, factory2D3)

pi = 4 * Math.Atn(1)
'betha = (pi / 2) / Z " Z=numero de dientes
p = pi * d1 / Z "paso circunferencial
m = p / pi
e = pi * m / 2 "" Espesor de un diente curvilineo
'rb = d1 / 2
'rc = rb * Math.Cos(betha)
A = m 'addendum
B = 1.25 * m 'dedenddum
C = 0.25 * m 'db
ri = 0.166 * m 'radio de pie/union
h = A + B


'"radios y diametros
db = d1 * Math.Cos(alpha * pi / 180)
df = d1 - 2 * B
rb = db / 2
rf = df / 2


'Dim line2d7 As Line2D
Set line2D7 = axis2D3.GetItem("HDirection")
line2D7.ReportName = 1
'Dim line2d8 As Line2D
Set line2D8 = axis2D3.GetItem("VDirection")
line2D8.ReportName = 2


"""""DEFINIMOS LA PRIMERA INVOLUTA

theta1 = 0
x1 = (rf + 2 * ri) * (Math.Cos(theta1) + theta1 * Math.Sin(theta1))
y1 = (rf + 2 * ri) * (Math.Sin(theta1) - theta1 * Math.Cos(theta1))

theta2 = 7 * pi / 180
x2 = (rf + 2 * ri) * (Math.Cos(theta2) + theta2 * Math.Sin(theta2))
y2 = (rf + 2 * ri) * (Math.Sin(theta2) - theta2 * Math.Cos(theta2))

theta3 = 14 * pi / 180
x3 = (rf + 2 * ri) * (Math.Cos(theta3) + theta3 * Math.Sin(theta3))
y3 = (rf + 2 * ri) * (Math.Sin(theta3) - theta3 * Math.Cos(theta3))
```

```
theta4 = 21 * pi / 180
x4 = (rf + 2 * ri) * (Math.Cos(theta4) + theta4 * Math.Sin(theta4))
y4 = (rf + 2 * ri) * (Math.Sin(theta4) - theta4 * Math.Cos(theta4))

theta5 = 28 * pi / 180
X5 = (rf + 2 * ri) * (Math.Cos(theta5) + theta5 * Math.Sin(theta5))
Y5 = (rf + 2 * ri) * (Math.Sin(theta5) - theta5 * Math.Cos(theta5))

theta6 = 35 * pi / 180
X6 = (rf + 2 * ri) * (Math.Cos(theta6) + theta6 * Math.Sin(theta6))
Y6 = rb * (Math.Sin(theta6) - theta6 * Math.Cos(theta6))

theta7 = 65 * pi / 180
x7 = (rf + 2 * ri) * (Math.Cos(theta7) + theta7 * Math.Sin(theta7))
y7 = (rf + 2 * ri) * (Math.Sin(theta7) - theta7 * Math.Cos(theta7))

theta8 = 65 * pi / 180  "49  54
x8 = (rf + 2 * ri) * (Math.Cos(theta8) + theta8 * Math.Sin(theta8))
y8 = (rf + 2 * ri) * (Math.Sin(theta8) - theta8 * Math.Cos(theta8))

'theta9 = 56 * pi / 180
'x9 = rf * (Math.Cos(theta9) + theta9 * Math.Sin(theta9))
'Y9 = rf * (Math.Sin(theta9) - theta9 * Math.Cos(theta9))

X0 = rf + ri

Dim point00 As ControlPoint2D
Set point00 = factory2D3.CreateControlPoint(X0, 0)

Dim point1 As ControlPoint2D
Set point1 = factory2D3.CreateControlPoint(x1, y1)

Dim point2 As ControlPoint2D
Set point2 = factory2D3.CreateControlPoint(x2, y2)

Dim point3 As ControlPoint2D
Set point3 = factory2D3.CreateControlPoint(x3, y3)

Dim point4 As ControlPoint2D
Set point4 = factory2D3.CreateControlPoint(x4, y4)

Dim point5 As ControlPoint2D
Set point5 = factory2D3.CreateControlPoint(X5, Y5)

Dim point6 As ControlPoint2D
Set point6 = factory2D3.CreateControlPoint(X6, Y6)

Dim point7 As ControlPoint2D
Set point7 = factory2D3.CreateControlPoint(x7, y7)

'Dim point8 As ControlPoint2D
'Set point8 = factory2d3.CreateControlPoint(x8, y8)

'Dim point9 As ControlPoint2D
'Set point9 = factory2D2.CreateControlPoint(x9, Y9)
```

```
        Dim arrayOfObject1(7)

        Set arrayOfObject1(0) = point00
        Set arrayOfObject1(1) = point1
        Set arrayOfObject1(2) = point2
        Set arrayOfObject1(3) = point3
        Set arrayOfObject1(4) = point4
        Set arrayOfObject1(5) = point5
        Set arrayOfObject1(6) = point6
        Set arrayOfObject1(7) = point7
        'Set arrayOfObject1(8) = point8
        'Set arrayOfObject1(8) = point9

        Dim spline2D1 As Spline2D
        Set factory2D1temp = factory2D3
        Set spline2D1 = factory2D1temp.CreateSpline(arrayOfObject1)

        Dim point0 As Point2D
        Set point0 = axis2D3.GetItem("Origin")

        Dim constraints1 As Constraints
        Set constraints1 = sketch3.Constraints
        """"arco de circulo primitivo del que obtendremos el espline simetrico
        Dim circle1 As Circle2D
        Set circle1 = factory2D3.CreateCircle(0, 0, d1 / 2, 3 * pi / 2, 2 * pi)
        Dim pointcircle1 As Point2D
        Set pointcircle1 = factory2D3.CreatePoint(d1 / 2 * Math.Cos(3 * pi / 2), d1 / 2 *
Math.Sin(3 * pi / 2))
        Dim pointcircle2 As Point2D
        Set pointcircle2 = factory2D3.CreatePoint(d1 / 2 * Math.Cos(2 * pi), d1 / 2 *
Math.Sin(2 * pi))
        Dim constraintc1 As Constraint
        Dim refc1 As Reference
        Set refc1 = part1.CreateReferenceFromObject(pointcircle1)
        Dim refc2 As Reference
        Set refc2 = part1.CreateReferenceFromObject(line2D7)
        Set constraintc1 = constraints1.AddBiEltCst(catCstTypeDistance, refc1, refc2)
        constraintc1.Mode = catCstModeDrivingDimension
        Dim lengthc1 As Length
        Set lengthc1 = constraintc1.Dimension
        lengthc1.Value = d1 / 2 * Math.Sin(2 * pi)
        Dim constraintc2 As Constraint
        Dim refc3 As Reference
        Set refc3 = part1.CreateReferenceFromObject(pointcircle1)
        Dim refc4 As Reference
        Set refc4 = part1.CreateReferenceFromObject(line2D8)
        Set constraintc2 = constraints1.AddBiEltCst(catCstTypeDistance, refc3, refc4)
        constraintc2.Mode = catCstModeDrivingDimension
        Dim lengthc2 As Length
        Set lengthc2 = constraintc2.Dimension
        lengthc2.Value = d1 / 2 * Math.Cos(2 * pi)
        'Dim constraintc3 As Constraint
```

```
'Dim refc5 As Reference
'Set refc5 = part1.CreateReferenceFromObject(pointcircle2)
'Dim refc6 As Reference
'Set refc6 = part1.CreateReferenceFromObject(line2d7)
'Set constraintc3 = constraints1.AddBiEltCst(catCstTypeDistance, refc5, refc6)
'constraintc3.Mode = catCstModeDrivingDimension
'Dim lengthc3 As Length
'Set lengthc3 = constraintc3.Dimension
'lengthc3.Value = d1 / 2 * Math.Sin(3 / 2 * pi)
'Dim constraintc4 As Constraint
'Dim refc7 As Reference
'Set refc7 = part1.CreateReferenceFromObject(pointcircle2)
'Dim refc8 As Reference
'Set refc8 = part1.CreateReferenceFromObject(line2d8)
'Set constraintc4 = constraints1.AddBiEltCst(catCstTypeDistance, refc7, refc8)
'constraintc4.Mode = catCstModeDrivingDimension
'Dim lengthc4 As Length
'Set lengthc4 = constraintc4.Dimension
'lengthc4.Value = d1 / 2 * Math.Cos(3 / 2 * pi)

circle1.CenterPoint = point0
circle1.StartPoint = pointcircle1
circle1.EndPoint = pointcircle2


X = d1 / 2 * Math.Cos(pi / 10)
Y = d1 / 2 * Math.Sin(pi / 10)
Dim pointict As Point2D
Set pointict = factory2D3.CreatePoint(X, Y)
'pointict.Construction = False

"""fijamos puntos del spline


'p00
Dim constraint00 As Constraint
Dim refcloti1 As Reference
Set refcloti1 = part1.CreateReferenceFromObject(point00)
Dim refcloti2 As Reference
Set refcloti2 = part1.CreateReferenceFromObject(line2D7)
Set constraint00 = constraints1.AddBiEltCst(catCstTypeStDistance, refcloti1, refcloti2)
constraint00.Mode = catCstModeDrivingDimension
Dim length00 As Length
Set length00 = constraint00.Dimension
length00.Value = 0

Dim constraint000 As Constraint
Dim refcloti3 As Reference
Set refcloti3 = part1.CreateReferenceFromObject(point00)
Dim refcloti4 As Reference
Set refcloti4 = part1.CreateReferenceFromObject(line2D8)
Set constraint000 = constraints1.AddBiEltCst(catCstTypeStDistance, refcloti3,
refcloti4)
constraint000.Mode = catCstModeDrivingDimension
Dim length000 As Length
```

```
Set length000 = constraint000.Dimension
length000.Value = X0


'p6
Dim constraint4 As Constraint
Dim ref5 As Reference
Set ref5 = part1.CreateReferenceFromObject(point6)
Dim ref6 As Reference
Set ref6 = part1.CreateReferenceFromObject(line2D7)
Set constraint4 = constraints1.AddBiEltCst(catCstTypeStDistance, ref5, ref6)
constraint4.Mode = catCstModeDrivingDimension
Dim length4 As Length
Set length4 = constraint4.Dimension
length4.Value = Y6



Dim constraint5 As Constraint
Dim ref7 As Reference
Set ref7 = part1.CreateReferenceFromObject(point6)
Dim ref8 As Reference
Set ref8 = part1.CreateReferenceFromObject(line2D8)
Set constraint5 = constraints1.AddBiEltCst(catCstTypeStDistance, ref7, ref8)
constraint5.Mode = catCstModeDrivingDimension
Dim length5 As Length
Set length5 = constraint5.Dimension
length5.Value = X6


'p7
Dim constraint6 As Constraint
Dim ref9 As Reference
Set ref9 = part1.CreateReferenceFromObject(point7)
Dim ref10 As Reference
Set ref10 = part1.CreateReferenceFromObject(line2D7)
Set constraint6 = constraints1.AddBiEltCst(catCstTypeStDistance, ref9, ref10)
constraint6.Mode = catCstModeDrivingDimension
Dim length6 As Length
Set length6 = constraint4.Dimension
length4.Value = y7



Dim constraint7 As Constraint
Dim ref11 As Reference
Set ref11 = part1.CreateReferenceFromObject(point7)
Dim ref12 As Reference
Set ref12 = part1.CreateReferenceFromObject(line2D8)
Set constraint7 = constraints1.AddBiEltCst(catCstTypeStDistance, ref11, ref12)
constraint7.Mode = catCstModeDrivingDimension
Dim length7 As Length
Set length7 = constraint5.Dimension
length7.Value = x7
```

```
'p5
Dim constraint8 As Constraint
Dim ref13 As Reference
Set ref13 = part1.CreateReferenceFromObject(point5)
Dim ref14 As Reference
Set ref14 = part1.CreateReferenceFromObject(line2D7)
Set constraint8 = constraints1.AddBiEltCst(catCstTypeStDistance, ref13, ref14)
constraint8.Mode = catCstModeDrivingDimension
Dim length8 As Length
Set length8 = constraint8.Dimension
length8.Value = Y5


Dim constraint9 As Constraint
Dim ref15 As Reference
Set ref15 = part1.CreateReferenceFromObject(point5)
Dim ref16 As Reference
Set ref16 = part1.CreateReferenceFromObject(line2D8)
Set constraint9 = constraints1.AddBiEltCst(catCstTypeStDistance, ref15, ref16)
constraint9.Mode = catCstModeDrivingDimension
Dim length9 As Length
Set length9 = constraint9.Dimension
length9.Value = X5

'p4
Dim constraint10 As Constraint
Dim ref17 As Reference
Set ref17 = part1.CreateReferenceFromObject(point4)
Dim ref18 As Reference
Set ref18 = part1.CreateReferenceFromObject(line2D7)
Set constraint10 = constraints1.AddBiEltCst(catCstTypeStDistance, ref17, ref18)
constraint10.Mode = catCstModeDrivingDimension
Dim length10 As Length
Set length10 = constraint10.Dimension
length10.Value = y4


Dim constraint11 As Constraint
Dim ref19 As Reference
Set ref19 = part1.CreateReferenceFromObject(point4)
Dim ref20 As Reference
Set ref20 = part1.CreateReferenceFromObject(line2D8)
Set constraint11 = constraints1.AddBiEltCst(catCstTypeStDistance, ref19, ref20)
constraint11.Mode = catCstModeDrivingDimension
Dim length11 As Length
Set length11 = constraint11.Dimension
length11.Value = x4


'p3
Dim constraint12 As Constraint
Dim ref21 As Reference
```

```
Set ref21 = part1.CreateReferenceFromObject(point3)
Dim ref22 As Reference
Set ref22 = part1.CreateReferenceFromObject(line2D7)
Set constraint12 = constraints1.AddBiEltCst(catCstTypeStDistance, ref21, ref22)
constraint12.Mode = catCstModeDrivingDimension
Dim length12 As Length
Set length12 = constraint12.Dimension
length12.Value = y3


Dim constraint13 As Constraint
Dim ref23 As Reference
Set ref23 = part1.CreateReferenceFromObject(point3)
Dim ref24 As Reference
Set ref24 = part1.CreateReferenceFromObject(line2D8)
Set constraint13 = constraints1.AddBiEltCst(catCstTypeStDistance, ref23, ref24)
constraint13.Mode = catCstModeDrivingDimension
Dim length13 As Length
Set length13 = constraint13.Dimension
length13.Value = x3


'p2
Dim constraint14 As Constraint
Dim ref25 As Reference
Set ref25 = part1.CreateReferenceFromObject(point2)
Dim ref26 As Reference
Set ref26 = part1.CreateReferenceFromObject(line2D7)
Set constraint14 = constraints1.AddBiEltCst(catCstTypeStDistance, ref25, ref26)
constraint14.Mode = catCstModeDrivingDimension
Dim length14 As Length
Set length14 = constraint14.Dimension
length14.Value = y2


Dim constraint15 As Constraint
Dim ref27 As Reference
Set ref27 = part1.CreateReferenceFromObject(point2)
Dim ref28 As Reference
Set ref28 = part1.CreateReferenceFromObject(line2D8)
Set constraint15 = constraints1.AddBiEltCst(catCstTypeStDistance, ref27, ref28)
constraint15.Mode = catCstModeDrivingDimension
Dim length15 As Length
Set length15 = constraint15.Dimension
length15.Value = x2


'p1
Dim constraint16 As Constraint
Dim ref29 As Reference
Set ref29 = part1.CreateReferenceFromObject(point1)
```

```
Dim ref30 As Reference
Set ref30 = part1.CreateReferenceFromObject(line2D7)
Set constraint16 = constraints1.AddBiEltCst(catCstTypeStDistance, ref29, ref30)
constraint16.Mode = catCstModeDrivingDimension
Dim length16 As Length
Set length16 = constraint16.Dimension
length16.Value = y1



Dim constraint17 As Constraint
Dim ref31 As Reference
Set ref31 = part1.CreateReferenceFromObject(point1)
Dim ref32 As Reference
Set ref32 = part1.CreateReferenceFromObject(line2D8)
Set constraint17 = constraints1.AddBiEltCst(catCstTypeStDistance, ref31, ref32)
constraint17.Mode = catCstModeDrivingDimension
Dim length17 As Length
Set length17 = constraint17.Dimension
length17.Value = x1



'p8

'Dim constraint101 As Constraint
'Dim ref102 As Reference
'Set ref102 = part1.CreateReferenceFromObject(point8)
'Dim ref103 As Reference
'Set ref103 = part1.CreateReferenceFromObject(line2D7)
'Set constraint101 = constraints1.AddBiEltCst(catCstTypeStDistance, ref102, ref103)
'constraint101.Mode = catCstModeDrivingDimension
'Dim length101 As Length
'Set length101 = constraint101.Dimension
'length101.Value = y8


'Dim constraint100 As Constraint
'Dim ref100 As Reference
'Set ref100 = part1.CreateReferenceFromObject(point8)
'Dim ref101 As Reference
'Set ref101 = part1.CreateReferenceFromObject(line2D8)
'Set constraint100 = constraints1.AddBiEltCst(catCstTypeStDistance, ref100, ref101)
'constraint100.Mode = catCstModeDrivingDimension
'Dim length100 As Length
'Set length100 = constraint100.Dimension
'length100.Value = x8


'p9

'Dim constraint103 As constraint
'Dim ref104 As Reference
'Set ref104 = part1.CreateReferenceFromObject(point9)
'Dim ref105 As Reference
```

```
'Set ref105 = part1.CreateReferenceFromObject(line2d7)
'Set constraint103 = constraints1.AddBiEltCst(catCstTypeStDistance, ref104, ref105)
'constraint103.Mode = catCstModeDrivingDimension
'Dim length103 As Length
'Set length103 = constraint103.Dimension
'length103.Value = Y9



'Dim constraint104 As constraint
'Dim ref106 As Reference
'Set ref106 = part1.CreateReferenceFromObject(point9)
'Dim ref107 As Reference
'Set ref107 = part1.CreateReferenceFromObject(line2d8)
'Set constraint104 = constraints1.AddBiEltCst(catCstTypeStDistance, ref106, ref107)
'constraint104.Mode = catCstModeDrivingDimension
'Dim length104 As Length
'Set length104 = constraint104.Dimension
'length104.Value = x9

Dim reference20 As Reference
Set reference20 = part1.CreateReferenceFromObject(spline2D1)
Dim reference30 As Reference
Set reference30 = part1.CreateReferenceFromObject(pointict)
Dim constraint200 As Constraint
Set constraint200 = constraints1.AddBiEltCst(catCstTypeOn, reference20, reference30)

'Dim reference40 As Reference
'Set reference40 = part1.CreateReferenceFromObject(circle1)
'Dim constraint201 As Constraint
'Set constraint201 = constraints1.AddMonoEltCst(catCstTypeRadius, reference40)
'constraint201.Mode = catCstModeDrivingDimension
'Dim length201 As Length
'Set length201 = constraint201.Dimension
'length201.Value = d1 / 2


Dim reference70 As Reference
Set reference70 = part1.CreateReferenceFromObject(circle1)
Dim reference50 As Reference
Set reference50 = part1.CreateReferenceFromObject(pointict)
Dim constraint300 As Constraint
Set constraint300 = constraints1.AddBiEltCst(catCstTypeOn, reference70, reference50)

""""" hemos restringido todos los puntos, ahora hallamos la linea de simetria, para lo cual
usaremos una circunferencia
'cuyo radio sera la cuerda del arco que define el espesor maximo de un perfil
'alpha1 = Math.Atn(Y / X)
betha = p / 2 / d1
co = (d1 / 2) * Math.Sin(betha)

Dim circleaux As Circle2D
Set circleaux = factory2D3.CreateClosedCircle(x1, y1, co)
circleaux.CenterPoint = pointict
```

```
        Dim pointaux As Point2D
        Set pointaux = factory2D3.CreatePoint(d1 / 2 * Math.Cos(-2 * pi / Z), d1 / 2 *
Math.Sin(-2 * pi / Z))
        'pointaux.Construction = False


        Dim constraintaux1 As Constraint
        Dim refaux1 As Reference
        Dim refaux2 As Reference
        Set refaux1 = part1.CreateReferenceFromObject(circle1)
        Set refaux2 = part1.CreateReferenceFromObject(pointaux)
        Set constraintaux1 = constraints1.AddBiEltCst(catCstTypeOn, refaux1, refaux2)

        Dim constraintaux2 As Constraint
        Dim refaux3 As Reference
        Dim refaux4 As Reference
        Set refaux3 = part1.CreateReferenceFromObject(circleaux)
        Set refaux4 = part1.CreateReferenceFromObject(pointaux)
        Set constraintaux2 = constraints1.AddBiEltCst(catCstTypeOn, refaux3, refaux4)

        Dim referenceaux As Reference
        Set referenceaux = part1.CreateReferenceFromObject(circleaux)
        Dim constraintaux3 As Constraint
        Set constraintaux3 = constraints1.AddMonoEltCst(catCstTypeRadius, referenceaux)
        constraintaux3.Mode = catCstModeDrivingDimension
        Dim lengthaux3 As Length
        Set lengthaux3 = constraintaux3.Dimension
        lengthaux3.Value = co


        Dim linemirror As line2D
        Set linemirror = factory2D3.CreateLine(0, 0, Xaux, Yaux)
        linemirror.StartPoint = point0
        linemirror.EndPoint = pointaux

        """" hacemos el mirror del spline2D1
        '" en primer lugar tenemos que crear otro spline, al cual le impondremos que sea
simétrico del primero.
        """""" involuta2
        angulo2 = -pi / Z

        X11 = rb * (Math.Cos(theta1 - angulo2) + theta1 * Math.Sin(theta1 - angulo2))
        Y11 = -rb * (Math.Sin(theta1 - angulo2) - theta1 * Math.Cos(theta1 - angulo2))

        theta2 = 7 * pi / 180
        X22 = rb * (Math.Cos(theta2 - angulo2) + theta2 * Math.Sin(theta2 - angulo2))
        Y22 = -rb * (Math.Sin(theta2 - angulo2) - theta2 * Math.Cos(theta2 - angulo2))

        theta3 = 14 * pi / 180
        X33 = rb * (Math.Cos(theta3 - angulo2) + theta3 * Math.Sin(theta3 - angulo2))
        Y33 = -rb * (Math.Sin(theta3 - angulo2) - theta3 * Math.Cos(theta3 - angulo2))

        theta4 = 21 * pi / 180
        X44 = rb * (Math.Cos(theta4 - angulo2) + theta4 * Math.Sin(theta4 - angulo2))
```

```
Y44 = -rb * (Math.Sin(theta4 - angulo2) - theta4 * Math.Cos(theta4 - angulo2))

theta5 = 28 * pi / 180
X55 = rb * (Math.Cos(theta5 - angulo2) + theta5 * Math.Sin(theta5 - angulo2))
Y55 = -rb * (Math.Sin(theta5 - angulo2) - theta5 * Math.Cos(theta5 - angulo2))

theta6 = 35 * pi / 180
X66 = rb * (Math.Cos(theta6 - angulo2) + theta6 * Math.Sin(theta6 - angulo2))
Y66 = -rb * (Math.Sin(theta6 - angulo2) - theta6 * Math.Cos(theta6 - angulo2))

theta7 = 42 * pi / 180
x77 = rb * (Math.Cos(theta7 - angulo2) + theta7 * Math.Sin(theta7 - angulo2))
y77 = -rb * (Math.Sin(theta7 - angulo2) - theta7 * Math.Cos(theta7 - angulo2))

theta8 = 49 * pi / 180
x88 = rb * (Math.Cos(theta8 - angulo2) + theta8 * Math.Sin(theta8 - angulo2))
y88 = -rb * (Math.Sin(theta8 - angulo2) - theta8 * Math.Cos(theta8 - angulo2))

'theta9 = 56 * pi / 180
'X99 = rf * (Math.Cos(theta9 - angulo2) + theta9 * Math.Sin(theta9 - angulo2))
'Y99 = -rf * (Math.Sin(theta9 - angulo2) - theta9 * Math.Cos(theta9 - angulo2))

Dim point000 As ControlPoint2D
Set point000 = factory2D3.CreateControlPoint(C, g)

Dim point11 As ControlPoint2D
Set point11 = factory2D3.CreateControlPoint(X11, Y11)

Dim point22 As ControlPoint2D
Set point22 = factory2D3.CreateControlPoint(X22, Y22)

Dim point33 As ControlPoint2D
Set point33 = factory2D3.CreateControlPoint(X33, Y33)

Dim point44 As ControlPoint2D
Set point44 = factory2D3.CreateControlPoint(X44, Y44)

Dim point55 As ControlPoint2D
Set point55 = factory2D3.CreateControlPoint(X55, Y55)

Dim point66 As ControlPoint2D
Set point66 = factory2D3.CreateControlPoint(X66, Y66)

Dim point77 As ControlPoint2D
Set point77 = factory2D3.CreateControlPoint(x77, y77)

Dim point88 As ControlPoint2D
Set point88 = factory2D3.CreateControlPoint(x88, y88)

'Dim point99 As ControlPoint2D
'Set point99 = factory2D2.CreateControlPoint(X99, Y99)

Dim arrayOfObject2(7)
Set arrayOfObject2(0) = point000
Set arrayOfObject2(1) = point11
```

```
Set arrayOfObject2(2) = point22
Set arrayOfObject2(3) = point33
Set arrayOfObject2(4) = point44
Set arrayOfObject2(5) = point55
Set arrayOfObject2(6) = point66
Set arrayOfObject2(7) = point77
'Set arrayOfObject2(8) = point88
'Set arrayOfObject2(8) = point99


Dim spline2D2 As Spline2D
Set factory2D1temp = factory2D3
Set spline2D2 = factory2D1temp.CreateSpline(arrayOfObject2)


Dim contraintmirror1 As Constraint
Dim refsim1 As Reference
Dim refsim2 As Reference
Dim refsim3 As Reference
Set refsim1 = part1.CreateReferenceFromObject(point1)
Set refsim2 = part1.CreateReferenceFromObject(point11)
Set refsim3 = part1.CreateReferenceFromObject(linemirror)
Set contrainstmirror1 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim1,
refsim2, refsim3)

Dim contraintmirror2 As Constraint
Dim refsim4 As Reference
Dim refsim5 As Reference
Set refsim4 = part1.CreateReferenceFromObject(point2)
Set refsim5 = part1.CreateReferenceFromObject(point22)
Set contrainstmirror2 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim4,
refsim5, refsim3)



Dim contraintmirror3 As Constraint
Dim refsim6 As Reference
Dim refsim7 As Reference
Set refsim6 = part1.CreateReferenceFromObject(point3)
Set refsim7 = part1.CreateReferenceFromObject(point33)
Set contrainstmirror3 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim6,
refsim7, refsim3)

Dim contraintmirror4 As Constraint
Dim refsim8 As Reference
Dim refsim9 As Reference
Set refsim8 = part1.CreateReferenceFromObject(point3)
Set refsim9 = part1.CreateReferenceFromObject(point33)
Set contrainstmirror4 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim8,
refsim9, refsim3)

Dim contraintmirror5 As Constraint
Dim refsim10 As Reference
Dim refsim11 As Reference
Set refsim10 = part1.CreateReferenceFromObject(point4)
```

```
        Set refsim11 = part1.CreateReferenceFromObject(point44)
        Set contrainstmirror5 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim10, refsim11, refsim3)

        Dim contraintmirror6 As Constraint
        Dim refsim12 As Reference
        Dim refsim13 As Reference
        Set refsim12 = part1.CreateReferenceFromObject(point5)
        Set refsim13 = part1.CreateReferenceFromObject(point55)
        Set contrainstmirror6 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim12, refsim13, refsim3)

        Dim contraintmirror7 As Constraint
        Dim refsim14 As Reference
        Dim refsim15 As Reference
        Set refsim14 = part1.CreateReferenceFromObject(point6)
        Set refsim15 = part1.CreateReferenceFromObject(point66)
        Set contrainstmirror7 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim14, refsim15, refsim3)

        Dim contraintmirror8 As Constraint
        Dim refsim16 As Reference
        Dim refsim17 As Reference
        Set refsim16 = part1.CreateReferenceFromObject(point7)
        Set refsim17 = part1.CreateReferenceFromObject(point77)
        Set contrainstmirror8 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim16, refsim17, refsim3)

        'Dim contraintmirror9 As Constraint
        'Dim refsim18 As Reference
        'Dim refsim19 As Reference
        'Set refsim18 = part1.CreateReferenceFromObject(point8)
        'Set refsim19 = part1.CreateReferenceFromObject(point88)
        'Set contrainstmirror5 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim18, refsim19, refsim3)

        'Dim contraintmirror10 As constraint
        'Dim refsim20 As Reference
        'Dim refsim21 As Reference
        'Set refsim20 = part1.CreateReferenceFromObject(point9)
        'Set refsim21 = part1.CreateReferenceFromObject(point99)
        'Set contrainstmirror10 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim20, refsim21, refsim3)

        Dim contraintmirror21 As Constraint
        Dim refsim24 As Reference
        Dim refsim25 As Reference
        Set refsim24 = part1.CreateReferenceFromObject(point00)
        Set refsim25 = part1.CreateReferenceFromObject(point000)
        Set contrainstmirror21 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim24, refsim25, refsim3)

        Dim contraintmirror20 As Constraint
        Dim refsim22 As Reference
        Dim refsim23 As Reference
```

```
        Set refsim22 = part1.CreateReferenceFromObject(spline2D1)
        Set refsim23 = part1.CreateReferenceFromObject(spline2D2)
        Set contrainstmirror20 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsim22,
refsim23, refsim3)



        """ cerramos el arco superior

        'r = Math.Sqr((X8 * X8) + (Y8 * Y8))

        'Dim arcsup As Circle2D
        'Set arcsup = factory2d3.CreateCircle(0, 0, r, -pi / Z, pi / Z)
        'arcsup.StartPoint = point88
        'arcsup.EndPoint = point8
        Dim linesup As line2D
        Set linesup = factory2D3.CreateLine(x7, y7, x77, y77)
        linesup.StartPoint = point7
        linesup.EndPoint = point77


        'Dim referencearcsup As Reference
        'Set referencearcsup = part1.CreateReferenceFromObject(arcsup)
        'Dim constraintarcsup As Constraint
        'Set constraintarcsup = constraints1.AddMonoEltCst(catCstTypeRadius,
referencearcsup)
        'constraintarcsup.Mode = catCstModeDrivingDimension
        'Dim lengtharcsup As Length
        'Set lengtharcsup = constraintarcsup.Dimension
        'lengtharcsup.Value = r


        circle1.Construction = True
        circleaux.Construction = True
        linemirror.Construction = True



        """arco inferior y esquinas

        'anguliti = (pi * ri) / (2 * rf)

        Xri = rf
        Yri = -ri

        Xri2 = rf + ri
        Yri2 = -ri

        Dim pointri As Point2D
        Set pointri = factory2D3.CreatePoint(Xri, Yri)

        Dim pointri2 As Point2D
        Set pointri2 = factory2D3.CreatePoint(Xri2, Yri2)
```

```
'pri
Dim constraintri1 As Constraint
Dim refer1 As Reference
Set refer1 = part1.CreateReferenceFromObject(pointri)
Dim refer2 As Reference
Set refer2 = part1.CreateReferenceFromObject(line2D7)
Set constraintri1 = constraints1.AddBiEltCst(catCstTypeStDistance, refer1, refer2)
constraintri1.Mode = catCstModeDrivingDimension
Dim lengthe1 As Length
Set lengthe1 = constraintri1.Dimension
lengthe1.Value = Yri


Dim constraintri2 As Constraint
Dim refer3 As Reference
Set refer3 = part1.CreateReferenceFromObject(pointri)
Dim refer4 As Reference
Set refer4 = part1.CreateReferenceFromObject(line2D8)
Set constraintri2 = constraints1.AddBiEltCst(catCstTypeStDistance, refer3, refer4)
constraintri2.Mode = catCstModeDrivingDimension
Dim lengthe2 As Length
Set lengthe2 = constraintri2.Dimension
lengthe2.Value = Xri


'pr2
'Dim constraintri3 As Constraint
'Dim refer5 As Reference
'Set refer5 = part1.CreateReferenceFromObject(pointri2)
'Dim refer6 As Reference
'Set refer6 = part1.CreateReferenceFromObject(line2d7)
'Set constraintri3 = constraints1.AddBiEltCst(catCstTypeStDistance, refer5, refer6)
'constraintri3.Mode = catCstModeDrivingDimension
'Dim lengthe3 As Length
'Set lengthe3 = constraintri3.Dimension
'lengthe3.Value = Yri2


'Dim constraintri4 As Constraint
'Dim refer7 As Reference
'Set refer7 = part1.CreateReferenceFromObject(pointri2)
'Dim refer8 As Reference
'Set refer8 = part1.CreateReferenceFromObject(line2d8)
'Set constraintri4 = constraints1.AddBiEltCst(catCstTypeStDistance, refer7, refer8)
'constraintri4.Mode = catCstModeDrivingDimension
'Dim lengthe4 As Length
'Set lengthe4 = constraintri4.Dimension
'lengthe4.Value = Xri2


Dim circle5 As Circle2D
Set circle5 = factory2D3.CreateCircle(Xri2, Yri2, ri, pi / 2, pi)
circle5.CenterPoint = pointri2
circle5.StartPoint = point00
circle5.EndPoint = pointri
```

```
        Dim constraintcircle5 As Constraint
        Dim referencecircle5 As Reference
        Set referencecircle5 = part1.CreateReferenceFromObject(circle5)
        Set constraintcircle5 = constraints1.AddMonoEltCst(catCstTypeRadius,
referencecircle5)
        constraintcircle5.Mode = catCstModeDrivingDimension
        Dim lengthcircle5 As Length
        Set lengthcircle5 = constraintcircle5.Dimension
        lengthcircle5.Value = ri

        """ hacemos mirror a los dos puntos del arco y al arco
        Dim pointb3 As Point2D
        Set pointb3 = factory2D3.CreatePoint(h, j)
        Dim pointb4 As Point2D
        Set pointb4 = factory2D3.CreatePoint(l, m)

        Dim constraintmirrorl As Constraint
        Dim refsiml1 As Reference
        Dim refsiml2 As Reference
        Set refsiml1 = part1.CreateReferenceFromObject(pointri)
        Set refsiml2 = part1.CreateReferenceFromObject(pointb3)
        Set constraintmirrorl = constraints1.AddTriEltCst(catCstTypeSymmetry, refsiml1,
refsiml2, refsim3)

        Dim constraintmirrorl1 As Constraint
        Dim refsiml11 As Reference
        Dim refsiml22 As Reference
        Set refsiml11 = part1.CreateReferenceFromObject(pointri2)
        Set refsiml22 = part1.CreateReferenceFromObject(pointb4)
        Set constraintmirrorl1 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsiml11,
refsiml22, refsim3)

        Dim circle6 As Circle2D
        Set circle6 = factory2D3.CreateCircle(0, 0, ri, pi, 3 * pi / 2)


        Dim constraintmirrorl11 As Constraint
        Dim refsiml111 As Reference
        Dim refsiml222 As Reference
        Set refsiml111 = part1.CreateReferenceFromObject(circle5)
        Set refsiml222 = part1.CreateReferenceFromObject(circle6)
        Set constraintmirrorl11 = constraints1.AddTriEltCst(catCstTypeSymmetry, refsiml111,
refsiml222, refsim3)
        circle6.StartPoint = pointb3
        circle6.EndPoint = point000

        Dim circle7 As Circle2D
        Set circle7 = factory2D3.CreateCircle(0, 0, rf, -pi / Z, pi / Z)
        circle7.CenterPoint = point0
        circle7.StartPoint = pointb3
        circle7.EndPoint = pointri


End Sub
```

ENGRANAJE RECTO

```
Private Sub CommandButton1_Click()


Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part

Dim bodies1 As Bodies
Dim body1 As Body
Dim sketches1 As Sketches
Dim originElements1 As OriginElements


Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part

Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
Set sketches1 = body1.Sketches
Set originElements1 = part1.OriginElements

Dim sketch2 As Sketch
Dim reference2 As Reference

Set reference2 = originElements1.PlaneXY
Set sketch2 = sketches1.Add(reference2)

Dim arrayOfVariantOfDouble2(8)
arrayOfVariantOfDouble2(0) = 0#
arrayOfVariantOfDouble2(1) = 0#
arrayOfVariantOfDouble2(2) = 0#
arrayOfVariantOfDouble2(3) = 1#
arrayOfVariantOfDouble2(4) = 0#
arrayOfVariantOfDouble2(5) = 0#
arrayOfVariantOfDouble2(6) = 0#
arrayOfVariantOfDouble2(7) = 1#
arrayOfVariantOfDouble2(8) = 0#
Set sketch2Variant = sketch2
sketch2Variant.SetAbsoluteAxisData arrayOfVariantOfDouble2
part1.InWorkObject = sketch2



Dim factory2d2 As Factory2D
Set factory2d2 = sketch2.OpenEdition()
Dim geometricelements2 As GeometricElements
Set geometricelements2 = part1.GeometricElements
Dim axis2d2 As Axis2D
Set axis2d2 = geometricelements2.Item("AbsoluteAxis")
Dim line2D7 As line2D
Set line2D7 = axis2d2.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D8 As line2D
```

```
Set line2D8 = axis2d2.GetItem("VDirection")
line2D8.ReportName = 2

""""""""" DATOS DE PARTIDA Y RELACIONES

d1 = TextBox1  "" diametro primitivo
Z = TextBox2
alpha = TextBox3 "" angulo de presion
altura = TextBox4

pi = 4 * Math.Atn(1)
'betha = (pi / 2) / Z " Z=numero de dientes
p = pi * d1 / Z "paso circunferencial
m = p / pi
e = pi * m / 2 "" Espesor de un diente curvilineo
rb = d1 / 2
rc = rb * Math.Cos(betha)
A = m 'addendum
B = 1.25 * m 'dedenddum
C = 0.25 * m 'db
ri = 0.166 * m 'radio de pie/union
h = A + B


'"radios y diametros
db = d1 * Math.Cos(alpha * pi / 180)
df = d1 - 2 * B
rb = db / 2
rf = df / 2


Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricelements2, axis2d2,
partDocument1, part1, sketch2, originElements1, factory2d2)

sketch2.CloseEdition
part1.InWorkObject = sketch2
part1.Update



Dim sketch1 As Sketch
Dim reference1 As Reference
Set reference1 = originElements1.PlaneXY
Set sketch1 = sketches1.Add(reference1)

Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0#
arrayOfVariantOfDouble1(1) = 0#
arrayOfVariantOfDouble1(2) = 0#
arrayOfVariantOfDouble1(3) = 1#
arrayOfVariantOfDouble1(4) = 0#
arrayOfVariantOfDouble1(5) = 0#
arrayOfVariantOfDouble1(6) = 0#
arrayOfVariantOfDouble1(7) = 1#
arrayOfVariantOfDouble1(8) = 0#
```

```
Set sketch1Variant = sketch1
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
part1.InWorkObject = sketch1



Dim factory2d1 As Factory2D
Set factory2d1 = sketch1.OpenEdition()
Dim geometricelements1 As GeometricElements
Set geometricelements1 = part1.GeometricElements
Dim axis2d1 As Axis2D
Set axis2d1 = geometricelements1.Item("AbsoluteAxis")
Dim line2D1 As line2D
Set line2D1 = axis2d1.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D2 As line2D
Set line2D2 = axis2d1.GetItem("VDirection")
line2D8.ReportName = 2

''' sketch
Dim circle2d1 As Circle2D
Set circle2d1 = factory2d1.CreateClosedCircle(0, 0, rf + h)

sketch1.CloseEdition
part1.InWorkObject = sketch1
part1.Update


Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory
Dim pad1 As Pad
Set pad1 = shapeFactory1.AddNewPad(sketch1, altura)
Dim limit1 As Limit
Set limit1 = pad1.FirstLimit
Dim lengthpad1 As Length
Set lengthpad1 = limit1.Dimension
lengthpad1.Value = -altura
part1.Update


'part1.InWorkObject = sketch2
Dim pocket1 As Pocket
Set pocket1 = shapeFactory1.AddNewPocket(sketch2, altura)
Dim limit2 As Limit
Set limit2 = pocket1.FirstLimit
Dim lengthpocket1 As Length
Set lengthpocket1 = limit2.Dimension
lengthpocket1.Value = altura
part1.Update



"""""""""""circular pattern

""""" Vamos a hacer el circular pattern
```

```
Dim refcirc5 As Reference
Set refcirc5 = part1.CreateReferenceFromName("")

Dim refcirc6 As Reference
Set refcirc6 = part1.CreateReferenceFromName("")



esp = 2 * p / d1 * 180 / pi
Dim circPattern1 As CircPattern
Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, esp, 1, 1,
refcirc5, refcirc6, True, 0#, True)

'parametro 5=angulo del diente
'parametro4=numero de dientes

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

Dim angularRepartition1 As AngularRepartition
Set angularRepartition1 = circPattern1.AngularRepartition

Dim intParam1 As IntParam
Set intParam1 = angularRepartition1.InstancesCount
intParam1.Value = Z

'Dim intParam2 As IntParam
'Set intParam2 = angularRepartition1.AngularSpacing

'intParam2.Value = 360 / N


Dim shapes1 As Shapes
Set shapes1 = body1.Shapes


Set pocket2 = shapes1.Item("Pocket.1")
circPattern1.ItemToCopy = pocket1


Dim refrot7 As Reference
Set refrot7 = part1.CreateReferenceFromObject(axis2d2)

circPattern1.SetRotationAxis refrot7

part1.UpdateObject circPattern1


""""""" hueco interno

Dim sketch3 As Sketch
Dim reference10 As Reference
Set reference10 = originElements1.PlaneXY
```

```
Set sketch3 = sketches1.Add(reference10)

Dim arrayOfVariantOfDouble3(8)
arrayOfVariantOfDouble3(0) = 0#
arrayOfVariantOfDouble3(1) = 0#
arrayOfVariantOfDouble3(2) = 0#
arrayOfVariantOfDouble3(3) = 1#
arrayOfVariantOfDouble3(4) = 0#
arrayOfVariantOfDouble3(5) = 0#
arrayOfVariantOfDouble3(6) = 0#
arrayOfVariantOfDouble3(7) = 1#
arrayOfVariantOfDouble3(8) = 0#
Set sketch3Variant = sketch3
sketch3Variant.SetAbsoluteAxisData arrayOfVariantOfDouble3
part1.InWorkObject = sketch3



Dim factory2D3 As Factory2D
Set factory2D3 = sketch3.OpenEdition()
Dim geometricElements3 As GeometricElements
Set geometricElements3 = part1.GeometricElements
Dim axis2D3 As Axis2D
Set axis2D3 = geometricElements3.Item("AbsoluteAxis")
Dim line2D4 As line2D
Set line2D4 = axis2D3.GetItem("HDirection")
line2D4.ReportName = 1
Dim line2D5 As line2D
Set line2D5 = axis2D3.GetItem("VDirection")
line2D5.ReportName = 2


'"sketch3
Dim circlec As Circle2D
Set circlec = factory2D3.CreateClosedCircle(0, 0, rf / 4)
sketch3.CloseEdition
part1.InWorkObject = sketch3
part1.Update


""hueeco central pocket
Dim pocket3 As Pocket
Set pocket3 = shapeFactory1.AddNewPocket(sketch3, altura)
Dim limitp2 As Limit
Set limitp2 = pocket3.FirstLimit
Dim lengthp2 As Length
Set lengthp2 = limitp2.Dimension
lengthp2.Value = altura
 part1.Update



End Sub
```

- **RECTO INTERNO**

```
Private Sub CommandButton1_Click()


Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part

Dim bodies1 As Bodies
Dim body1 As Body
Dim sketches1 As Sketches
Dim originElements1 As OriginElements


Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part

Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
Set sketches1 = body1.Sketches
Set originElements1 = part1.OriginElements

Dim sketch2 As Sketch
Dim reference2 As Reference

Set reference2 = originElements1.PlaneXY
Set sketch2 = sketches1.Add(reference2)

Dim arrayOfVariantOfDouble2(8)
arrayOfVariantOfDouble2(0) = 0#
arrayOfVariantOfDouble2(1) = 0#
arrayOfVariantOfDouble2(2) = 0#
arrayOfVariantOfDouble2(3) = 1#
arrayOfVariantOfDouble2(4) = 0#
arrayOfVariantOfDouble2(5) = 0#
arrayOfVariantOfDouble2(6) = 0#
arrayOfVariantOfDouble2(7) = 1#
arrayOfVariantOfDouble2(8) = 0#
Set sketch2Variant = sketch2
sketch2Variant.SetAbsoluteAxisData arrayOfVariantOfDouble2
part1.InWorkObject = sketch2



Dim factory2d2 As Factory2D
Set factory2d2 = sketch2.OpenEdition()
Dim geometricelements2 As GeometricElements
Set geometricelements2 = part1.GeometricElements
Dim axis2d2 As Axis2D
Set axis2d2 = geometricelements2.Item("AbsoluteAxis")
Dim line2D7 As line2D
Set line2D7 = axis2d2.GetItem("HDirection")
```

```
line2D7.ReportName = 1
Dim line2D8 As line2D
Set line2D8 = axis2d2.GetItem("VDirection")
line2D8.ReportName = 2

""""""" DATOS DE PARTIDA Y RELACIONES

d1 = TextBox1  "" diametro primitivo
Z = TextBox2
alpha = TextBox3 "" angulo de presion
altura = TextBox4

pi = 4 * Math.Atn(1)
'betha = (pi / 2) / Z " Z=numero de dientes
p = pi * d1 / Z "paso circunferencial
m = p / pi
e = pi * m / 2 "" Espesor de un diente curvilineo
rb = d1 / 2
rc = rb * Math.Cos(betha)
A = m 'addendum
B = 1.25 * m 'dedenddum
C = 0.25 * m 'db
ri = 0.166 * m 'radio de pie/union
h = A + B


'"radios y diametros
db = d1 * Math.Cos(alpha * pi / 180)
df = d1 - 2 * B
rb = db / 2
rf = df / 2


Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricelements2, axis2d2,
partDocument1, part1, sketch2, originElements1, factory2d2)

sketch2.CloseEdition
part1.InWorkObject = sketch2
part1.Update



Dim sketch1 As Sketch
Dim reference1 As Reference
Set reference1 = originElements1.PlaneXY
Set sketch1 = sketches1.Add(reference1)

Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0#
arrayOfVariantOfDouble1(1) = 0#
arrayOfVariantOfDouble1(2) = 0#
arrayOfVariantOfDouble1(3) = 1#
arrayOfVariantOfDouble1(4) = 0#
arrayOfVariantOfDouble1(5) = 0#
arrayOfVariantOfDouble1(6) = 0#
```

```
arrayOfVariantOfDouble1(7) = 1#
arrayOfVariantOfDouble1(8) = 0#
Set sketch1Variant = sketch1
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
part1.InWorkObject = sketch1



Dim factory2d1 As Factory2D
Set factory2d1 = sketch1.OpenEdition()
Dim geometricelements1 As GeometricElements
Set geometricelements1 = part1.GeometricElements
Dim axis2d1 As Axis2D
Set axis2d1 = geometricelements1.Item("AbsoluteAxis")
Dim line2D1 As line2D
Set line2D1 = axis2d1.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D2 As line2D
Set line2D2 = axis2d1.GetItem("VDirection")
line2D8.ReportName = 2

''' sketch
Dim circle2d1 As Circle2D
Set circle2d1 = factory2d1.CreateClosedCircle(0, 0, rf + h)

sketch1.CloseEdition
part1.InWorkObject = sketch1
part1.Update


Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory
Dim pad1 As Pad
Set pad1 = shapeFactory1.AddNewPad(sketch1, altura)
Dim limit1 As Limit
Set limit1 = pad1.FirstLimit
Dim lengthpad1 As Length
Set lengthpad1 = limit1.Dimension
lengthpad1.Value = -altura
part1.Update


'part1.InWorkObject = sketch2
Dim pocket1 As Pocket
Set pocket1 = shapeFactory1.AddNewPocket(sketch2, altura)
Dim limit2 As Limit
Set limit2 = pocket1.FirstLimit
Dim lengthpocket1 As Length
Set lengthpocket1 = limit2.Dimension
lengthpocket1.Value = altura
part1.Update



""""""""""circular pattern
```

```
"""" Vamos a hacer el circular pattern

Dim refcirc5 As Reference
Set refcirc5 = part1.CreateReferenceFromName("")

Dim refcirc6 As Reference
Set refcirc6 = part1.CreateReferenceFromName("")



esp = 2 * p / d1 * 180 / pi
Dim circPattern1 As CircPattern
Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, esp, 1, 1,
refcirc5, refcirc6, True, 0#, True)

'parametro 5=angulo del diente
'parametro4=numero de dientes

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

Dim angularRepartition1 As AngularRepartition
Set angularRepartition1 = circPattern1.AngularRepartition

Dim intParam1 As IntParam
Set intParam1 = angularRepartition1.InstancesCount
intParam1.Value = Z

'Dim intParam2 As IntParam
'Set intParam2 = angularRepartition1.AngularSpacing

'intParam2.Value = 360 / N


Dim shapes1 As Shapes
Set shapes1 = body1.Shapes


Set pocket2 = shapes1.Item("Pocket.1")
circPattern1.ItemToCopy = pocket1


Dim refrot7 As Reference
Set refrot7 = part1.CreateReferenceFromObject(axis2d2)

circPattern1.SetRotationAxis refrot7

part1.UpdateObject circPattern1


"""""""" hueco interno

Dim sketch3 As Sketch
```

```
Dim reference10 As Reference
Set reference10 = originElements1.PlaneXY
Set sketch3 = sketches1.Add(reference10)

Dim arrayOfVariantOfDouble3(8)
arrayOfVariantOfDouble3(0) = 0#
arrayOfVariantOfDouble3(1) = 0#
arrayOfVariantOfDouble3(2) = 0#
arrayOfVariantOfDouble3(3) = 1#
arrayOfVariantOfDouble3(4) = 0#
arrayOfVariantOfDouble3(5) = 0#
arrayOfVariantOfDouble3(6) = 0#
arrayOfVariantOfDouble3(7) = 1#
arrayOfVariantOfDouble3(8) = 0#
Set sketch3Variant = sketch3
sketch3Variant.SetAbsoluteAxisData arrayOfVariantOfDouble3
part1.InWorkObject = sketch3



Dim factory2D3 As Factory2D
Set factory2D3 = sketch3.OpenEdition()
Dim geometricElements3 As GeometricElements
Set geometricElements3 = part1.GeometricElements
Dim axis2D3 As Axis2D
Set axis2D3 = geometricElements3.Item("AbsoluteAxis")
Dim line2D4 As line2D
Set line2D4 = axis2D3.GetItem("HDirection")
line2D4.ReportName = 1
Dim line2D5 As line2D
Set line2D5 = axis2D3.GetItem("VDirection")
line2D5.ReportName = 2


'"sketch3
Dim circlec As Circle2D
Set circlec = factory2D3.CreateClosedCircle(0, 0, rf / 4)
sketch3.CloseEdition
part1.InWorkObject = sketch3
part1.Update


""hueeco central pocket
Dim pocket3 As Pocket
Set pocket3 = shapeFactory1.AddNewPocket(sketch3, altura)
Dim limitp2 As Limit
Set limitp2 = pocket3.FirstLimit
Dim lengthp2 As Length
Set lengthp2 = limitp2.Dimension
lengthp2.Value = altura
 part1.Update



End Sub
```

```
'''''''''''''''''''' body2

Dim body2 As Body
Set body2 = bodies1.Add()

part1.Update

Dim sketches2 As Sketches
Set sketches2 = body2.Sketches


Dim referenceb2 As Reference
Set referenceb2 = originElements1.PlaneXY


Dim sketch4 As Sketch
Set sketch4 = sketches2.Add(referenceb2)

Dim arrayOfVariantOfDouble4(8)
arrayOfVariantOfDouble4(0) = 0#
arrayOfVariantOfDouble4(1) = 0#
arrayOfVariantOfDouble4(2) = 0#
arrayOfVariantOfDouble4(3) = 1#
arrayOfVariantOfDouble4(4) = 0#
arrayOfVariantOfDouble4(5) = 0#
arrayOfVariantOfDouble4(6) = 0#
arrayOfVariantOfDouble4(7) = 1#
arrayOfVariantOfDouble4(8) = 0#
Set sketch4Variant = sketch4
sketch4Variant.SetAbsoluteAxisData arrayOfVariantOfDouble4
part1.InWorkObject = sketch4



Dim factory2D4 As Factory2D
Set factory2D4 = sketch4.OpenEdition()
Dim geometricElements4 As GeometricElements
Set geometricElements4 = part1.GeometricElements
Dim axis2D4 As Axis2D
Set axis2D4 = geometricElements4.Item("AbsoluteAxis")
Dim line2D17 As line2D
Set line2D17 = axis2D4.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D18 As line2D
Set line2D18 = axis2D4.GetItem("VDirection")
line2D8.ReportName = 2

Dim circleext As Circle2D
Set circleext = factory2D4.CreateClosedCircle(0, 0, dexterno / 2)

sketch4.CloseEdition
part1.InWorkObject = sketch4
```

```
part1.Update


''''padbody2
Dim pad2 As Pad
Set pad2 = shapeFactory1.AddNewPad(sketch4, altura)
Dim limitp20 As Limit
Set limitp20 = pad2.FirstLimit
Dim lengthp20 As Length
Set lengthp20 = limitp20.Dimension
lengthp20.Value = -altura

part1.Update

part1.InWorkObject = body2

Dim remove1 As Remove
Set remove1 = shapeFactory1.AddNewRemove(body1)

part1.Update



End Sub
```

HELICOIDAL.

```
Private Sub CommandButton1_Click()
Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part

Dim bodies1 As Bodies
Dim body1 As Body
Dim sketches1 As Sketches
Dim originElements1 As OriginElements


Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part

Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
Set sketches1 = body1.Sketches
Set originElements1 = part1.OriginElements

Dim sketch2 As Sketch
Dim reference2 As Reference

Set reference2 = originElements1.PlaneXY
```

```
Set sketch2 = sketches1.Add(reference2)

Dim arrayOfVariantOfDouble2(8)
arrayOfVariantOfDouble2(0) = 0#
arrayOfVariantOfDouble2(1) = 0#
arrayOfVariantOfDouble2(2) = 0#
arrayOfVariantOfDouble2(3) = 1#
arrayOfVariantOfDouble2(4) = 0#
arrayOfVariantOfDouble2(5) = 0#
arrayOfVariantOfDouble2(6) = 0#
arrayOfVariantOfDouble2(7) = 1#
arrayOfVariantOfDouble2(8) = 0#
Set sketch2Variant = sketch2
sketch2Variant.SetAbsoluteAxisData arrayOfVariantOfDouble2
part1.InWorkObject = sketch2



Dim factory2d2 As Factory2D
Set factory2d2 = sketch2.OpenEdition()
Dim geometricelements2 As GeometricElements
Set geometricelements2 = part1.GeometricElements
Dim axis2d2 As Axis2D
Set axis2d2 = geometricelements2.Item("AbsoluteAxis")
Dim line2D7 As line2D
Set line2D7 = axis2d2.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D8 As line2D
Set line2D8 = axis2d2.GetItem("VDirection")
line2D8.ReportName = 2

''''''''' DATOS DE PARTIDA Y RELACIONES

''''''''' DATOS DE PARTIDA Y RELACIONES

m = TextBox1 'd1 = TextBox1  '''' diametro primitivo
Z = TextBox2
alpha = TextBox3 '''' angulo de presion
altura = 5 * m 'altura = TextBox4
phi = TextBox4

pi = 4 * Math.Atn(1)
p = pi * m '* d1 / Z ''paso circunferencial
d1 = (p * Z) / (pi * m)
'Z = pi * m / p
'm = p / pi
e = pi * m / 2 '''' Espesor de un diente curvilineo
rb = d1 / 2
rc = rb * Math.Cos(betha)
A = m 'addendum
B = 1.25 * m 'dedenddum
C = 0.25 * m 'db
```

```
ri = 0.166 * m 'radio de pie/union
h = A + B


'''radios y diametros
db = d1 * Math.Cos(alpha * pi / 180)
df = d1 - 2 * B
rb = db / 2
rf = df / 2

de = rf + 2 * h
giro = -altura * Math.Tan(phi * pi / 180) / de


Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricelements2, axis2d2, partDocument1, part1, sketch2, originElements1, factory2d2)


sketch2.CloseEdition
part1.InWorkObject = sketch2
part1.Update

Dim sketch1 As Sketch
Dim reference1 As Reference
Set reference1 = originElements1.PlaneXY
Set sketch1 = sketches1.Add(reference1)

Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0#
arrayOfVariantOfDouble1(1) = 0#
arrayOfVariantOfDouble1(2) = 0#
arrayOfVariantOfDouble1(3) = 1#
arrayOfVariantOfDouble1(4) = 0#
arrayOfVariantOfDouble1(5) = 0#
arrayOfVariantOfDouble1(6) = 0#
arrayOfVariantOfDouble1(7) = 1#
arrayOfVariantOfDouble1(8) = 0#
Set sketch1Variant = sketch1
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
part1.InWorkObject = sketch1


Dim factory2d1 As Factory2D
Set factory2d1 = sketch1.OpenEdition()
Dim geometricelements1 As GeometricElements
Set geometricelements1 = part1.GeometricElements
Dim axis2d1 As Axis2D
Set axis2d1 = geometricelements1.Item("AbsoluteAxis")
Dim line2D1 As line2D
Set line2D1 = axis2d1.GetItem("HDirection")
line2D7.ReportName = 1
```

```
Dim line2D2 As line2D
Set line2D2 = axis2d1.GetItem("VDirection")
line2D8.ReportName = 2

''' sketch
Dim circle2d1 As Circle2D
Set circle2d1 = factory2d1.CreateClosedCircle(0, 0, rf + h)

sketch1.CloseEdition
part1.InWorkObject = sketch1
part1.Update


Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory
Dim pad1 As Pad
Set pad1 = shapeFactory1.AddNewPad(sketch1, altura)
Dim limit1 As Limit
Set limit1 = pad1.FirstLimit
Dim lengthpad1 As Length
Set lengthpad1 = limit1.Dimension
lengthpad1.Value = -altura
part1.Update




''''''plano para sketch3
'''''Dim hybridBodies1 As HybridBodies
'''''Set hybridBodies1 = part1.HybridBodies

''''Dim hybridBody1 As HybridBody
'''''Set hybridBody1 = hybridBodies1.Add()

Dim hshapefactory1 As HybridShapeFactory
Set hshapefactory1 = part1.HybridShapeFactory

Dim hybridShapePlaneExplicit1 As HybridShapePlaneExplicit
Set hybridShapePlaneExplicit1 = originElements1.PlaneXY

Dim reference10 As Reference
Set reference10 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit1)

Dim hybridShapePlaneOffset1 As HybridShapePlaneOffset
Set hybridShapePlaneOffset1 = hshapefactory1.AddNewPlaneOffset(reference10, -
altura, False)


body1.InsertHybridShape hybridShapePlaneOffset1

part1.InWorkObject = hybridShapePlaneOffset1
```

```
part1.Update

Dim hybridShapes1 As HybridShapes
Set hybridShapes1 = body1.HybridShapes

'''''sketch3

Dim reference20 As Reference
Set reference20 = hybridShapes1.Item("Plane.1")

Dim sketch3 As Sketch
Set sketch3 = sketches1.Add(reference20)

Dim arrayOfVariantOfDouble3(8)
arrayOfVariantOfDouble3(0) = 0#
arrayOfVariantOfDouble3(1) = 0#
arrayOfVariantOfDouble3(2) = -altura
arrayOfVariantOfDouble3(3) = 1#
arrayOfVariantOfDouble3(4) = 0#
arrayOfVariantOfDouble3(5) = 0#
arrayOfVariantOfDouble3(6) = 0#
arrayOfVariantOfDouble3(7) = 1#
arrayOfVariantOfDouble3(8) = 0#
Set sketch3Variant = sketch3
sketch3Variant.SetAbsoluteAxisData arrayOfVariantOfDouble3

part1.InWorkObject = sketch3

Dim factory2D3 As Factory2D
Set factory2D3 = sketch3.OpenEdition()
Dim geometricElements3 As GeometricElements
Set geometricElements3 = sketch3.GeometricElements
Dim axis2D3 As Axis2D
Set axis2D3 = geometricElements3.Item("AbsoluteAxis")
Dim line2D10 As line2D
Set line2D10 = axis2D3.GetItem("HDirection")
Dim line2D11 As line2D
Set line2D11 = axis2D3.GetItem("VDirection")

Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricElements3, axis2D3,
partDocument1, part1, sketch3, originElements1, factory2D3)
sketch3.CloseEdition
part1.InWorkObject = sketch3
part1.Update


Dim hybridShapeRotate1 As HybridShapeRotate
Set hybridShapeRotate1 = hshapefactory1.AddNewEmptyRotate()

Dim referencee2 As Reference
Set referencee2 = part1.CreateReferenceFromObject(sketch2)
```

```
hybridShapeRotate1.ElemToRotate = referencee2

hybridShapeRotate1.VolumeResult = False

hybridShapeRotate1.RotationType = 0



'Dim hybridShapeLineExplicit1 As HybridShapeLineExplicit
'Set hybridShapeLineExplicit1 = hybridShapes1.Item("Z Axis")

Dim referencee3 As Reference
Set referencee3 = part1.CreateReferenceFromObject(axis2D3)

hybridShapeRotate1.Axis = referencee3

hybridShapeRotate1.AngleValue = giro * 180 / pi
body1.InsertHybridShape hybridShapeRotate1

part1.InWorkObject = hybridShapeRotate1

part1.Update

''''' borro el 1
Dim selection1 As Selection
Set selection1 = partDocument1.Selection

Dim visPropertySet1 As VisPropertySet
Set visPropertySet1 = selection1.VisProperties

Dim bSTR1 As String
bSTR1 = sketch1.Name

selection1.Add sketch2

Set visPropertySet1 = visPropertySet1.Parent

Dim bSTR2 As String
bSTR2 = visPropertySet1.Name

Dim bSTR3 As String
bSTR3 = visPropertySet1.Name

visPropertySet1.SetShow 1

selection1.Clear

part1.Update
```

```
''''plano medio

Dim hybridShapePlaneExplicit2 As HybridShapePlaneExplicit
Set hybridShapePlaneExplicit2 = originElements1.PlaneXY

Dim reference100 As Reference
Set reference100 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit2)

Dim hybridShapePlaneOffset2 As HybridShapePlaneOffset
Set hybridShapePlaneOffset2 = hshapefactory1.AddNewPlaneOffset(reference10, -
altura / 2, False)


body1.InsertHybridShape hybridShapePlaneOffset2

part1.InWorkObject = hybridShapePlaneOffset2

part1.Update

'Dim hybridShapes1 As HybridShapes
'Set hybridShapes1 = body1.HybridShapes

'''''sketch4

Dim reference200 As Reference
Set reference200 = hybridShapes1.Item("Plane.2")

Dim sketch4 As Sketch
Set sketch4 = sketches1.Add(reference200)

Dim arrayOfVariantOfDouble4(8)
arrayOfVariantOfDouble4(0) = 0#
arrayOfVariantOfDouble4(1) = 0#
arrayOfVariantOfDouble4(2) = -altura
arrayOfVariantOfDouble4(3) = 1#
arrayOfVariantOfDouble4(4) = 0#
arrayOfVariantOfDouble4(5) = 0#
arrayOfVariantOfDouble4(6) = 0#
arrayOfVariantOfDouble4(7) = 1#
arrayOfVariantOfDouble4(8) = 0#
Set sketch4Variant = sketch4
sketch4Variant.SetAbsoluteAxisData arrayOfVariantOfDouble4

part1.InWorkObject = sketch4

Dim factory2D4 As Factory2D
Set factory2D4 = sketch4.OpenEdition()
Dim geometricElements4 As GeometricElements
Set geometricElements4 = sketch4.GeometricElements
Dim axis2D4 As Axis2D
Set axis2D4 = geometricElements4.Item("AbsoluteAxis")
```

```
Dim line2D12 As line2D
Set line2D12 = axis2D4.GetItem("HDirection")
Dim line2D13 As line2D
Set line2D13 = axis2D4.GetItem("VDirection")



Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricElements4, axis2D4, partDocument1, part1, sketch4, originElements1, factory2D4)
sketch4.CloseEdition
part1.InWorkObject = sketch4
part1.Update


Dim hybridShapeRotate10 As HybridShapeRotate
Set hybridShapeRotate10 = hshapefactory1.AddNewEmptyRotate()

Dim referencee20 As Reference
Set referencee20 = part1.CreateReferenceFromObject(sketch4)

hybridShapeRotate10.ElemToRotate = referencee20

hybridShapeRotate10.VolumeResult = False

hybridShapeRotate10.RotationType = 0



'Dim hybridShapeLineExplicit1 As HybridShapeLineExplicit
'Set hybridShapeLineExplicit1 = hybridShapes1.Item("Z Axis")

Dim referencee30 As Reference
Set referencee30 = part1.CreateReferenceFromObject(axis2D4)

hybridShapeRotate10.Axis = referencee30

hybridShapeRotate10.AngleValue = giro * 180 / pi / 2
body1.InsertHybridShape hybridShapeRotate10

part1.InWorkObject = hybridShapeRotate10

part1.Update

''''' borro el 1
Dim selection10 As Selection
Set selection10 = partDocument1.Selection

'Dim visPropertySet10 As VisPropertySet
'Set visPropertySet10 = selection10.VisProperties

'Dim bSTR10 As String
```

```
'bSTR10 = sketch1.Name

selection10.Add sketch4

'Set visPropertySet1 = visPropertySet1.Parent

'Dim bSTR20 As String
'bSTR20 = visPropertySet10.Name

'Dim bSTR30 As String
'bSTR30 = visPropertySet10.Name

'visPropertySet1.SetShow 10

selection10.Clear

part1.Update




''''''''''''''''''guias''''''''''''''''''

''''''''''''''''''GUIA1
'theta7 = 42 * pi / 180
'x7 = rb * (Math.Cos(theta7) + theta7 * Math.Sin(theta7))
'y7 = rb * (Math.Sin(theta7) - theta7 * Math.Cos(theta7))


'Xg1 = rb * (Math.Cos(theta7) + theta7 * Math.Sin(theta7))
'Xg2 = rb * (Math.Cos(theta7 + giro / 2) + theta7 * Math.Sin(theta7 + giro / 2))
'Xg3 = rb * (Math.Cos(theta7 + giro) + theta7 * Math.Sin(theta7 + giro))
'Yg1 = rb * (Math.Sin(theta7) - theta7 * Math.Cos(theta7))
'Yg2 = rb * (Math.Sin(theta7 + giro / 2) - theta7 * Math.Cos(theta7 + giro / 2))
'Yg3 = rb * (Math.Sin(theta7 + giro) - theta7 * Math.Cos(theta7 + giro))
'Zg1 = -altura
'Zg2 = -altura / 2
'Zg3 = 0

'''''Dim hybridShapeFactory1 As HybridShapeFactory
'''''Set hybridShapeFactory1 = part1.HybridShapeFactory

'Dim hybridShapePointCoord1 As HybridShapePointCoord
'Set hybridShapePointCoord1 = hshapefactory1.AddNewPointCoord(Xg1, Yg1, Zg1)


'body1.InsertHybridShape hybridShapePointCoord1

'part1.InWorkObject = hybridShapePointCoord1

'part1.Update
```

```
'Dim hybridShapePointCoord2 As HybridShapePointCoord
'Set hybridShapePointCoord2 = hshapefactory1.AddNewPointCoord(Xg2, Yg2, Zg2)

'body1.InsertHybridShape hybridShapePointCoord2

'part1.InWorkObject = hybridShapePointCoord2

'part1.Update

'Dim hybridShapePointCoord3 As HybridShapePointCoord
'Set hybridShapePointCoord3 = hshapefactory1.AddNewPointCoord(Xg3, Yg3, Zg3)

'body1.InsertHybridShape hybridShapePointCoord3

'part1.InWorkObject = hybridShapePointCoord3


'part1.Update

'Dim hybridShapeSpline1 As HybridShapeSpline
'Set hybridShapeSpline1 = hshapefactory1.AddNewSpline()'

'hybridShapeSpline1.SetSplineType 0

'hybridShapeSpline1.SetClosing 0 '

'Dim refer1 As Reference
'Set refer1 = part1.CreateReferenceFromObject(hybridShapePointCoord1)

'hybridShapeSpline1.AddPointWithConstraintExplicit refer1, Nothing, -1#, 1, Nothing,
0#

'Dim refer2 As Reference
'Set refer2 = part1.CreateReferenceFromObject(hybridShapePointCoord2)

'hybridShapeSpline1.AddPointWithConstraintExplicit refer2, Nothing, -1#, 1, Nothing,
0#

'Dim refer3 As Reference
'Set refer3 = part1.CreateReferenceFromObject(hybridShapePointCoord3)

'hybridShapeSpline1.AddPointWithConstraintExplicit refer3, Nothing, -1#, 1, Nothing,
0#

'body1.InsertHybridShape hybridShapeSpline1

'part1.InWorkObject = hybridShapeSpline1

'part1.Update
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''
X0 = rf + ri
Y0 = 0

Xgg1 = X0
Ygg1 = Y0
Xgg2 = (rf + ri) * Math.Cos(giro / 2)
Ygg2 = (rf + ri) * Math.Sin(giro / 2)
Xgg3 = (rf + ri) * Math.Cos(giro)
Ygg3 = (rf + ri) * Math.Sin(giro)


'Dim hybridShapePointCoord11 As HybridShapePointCoord
'Set hybridShapePointCoord11 = hshapefactory1.AddNewPointCoord(Xgg1, Ygg1, Zg1)


'body1.InsertHybridShape hybridShapePointCoord11

'part1.InWorkObject = hybridShapePointCoord11

'part1.Update

'Dim hybridShapePointCoord22 As HybridShapePointCoord
'Set hybridShapePointCoord22 = hshapefactory1.AddNewPointCoord(Xgg2, Ygg2, Zg2)

'body1.InsertHybridShape hybridShapePointCoord22

'part1.InWorkObject = hybridShapePointCoord22

'part1.Update

'Dim hybridShapePointCoord33 As HybridShapePointCoord
'Set hybridShapePointCoord33 = hshapefactory1.AddNewPointCoord(Xgg3, Ygg3, Zg3)

'body1.InsertHybridShape hybridShapePointCoord33

'part1.InWorkObject = hybridShapePointCoord33


'part1.Update

'Dim hybridShapeSpline11 As HybridShapeSpline
'Set hybridShapeSpline11 = hshapefactory1.AddNewSpline()

'hybridShapeSpline11.SetSplineType 0

'hybridShapeSpline11.SetClosing 0 '

'Dim refer11 As Reference
'Set refer11 = part1.CreateReferenceFromObject(hybridShapePointCoord11)
```

```
'hybridShapeSpline11.AddPointWithConstraintExplicit refer11, Nothing, -1#, 1, Nothing, 0#

'Dim refer22 As Reference
'Set refer22 = part1.CreateReferenceFromObject(hybridShapePointCoord22)

'hybridShapeSpline11.AddPointWithConstraintExplicit refer22, Nothing, -1#, 1, Nothing, 0#

'Dim refer33 As Reference
'Set refer33 = part1.CreateReferenceFromObject(hybridShapePointCoord33)

'hybridShapeSpline11.AddPointWithConstraintExplicit refer33, Nothing, -1#, 1, Nothing, 0#

'body1.InsertHybridShape hybridShapeSpline11

'part1.InWorkObject = hybridShapeSpline11

'part1.Update

'''''''''''''''''''' hacemos el multisection remove

Dim loft1 As Loft
Set loft1 = shapeFactory1.AddNewRemovedLoft()

Dim hybridShapeLoft1 As HybridShapeLoft
Set hybridShapeLoft1 = loft1.HybridShape

hybridShapeLoft1.SectionCoupling = 3

hybridShapeLoft1.Relimitation = 1

hybridShapeLoft1.CanonicalDetection = 2


'Dim reference1000 As Reference
'Set reference1000 = part1.CreateReferenceFromObject(hybridShapeSpline11)

'hybridShapeLoft1.AddGuide reference1000


Dim reference3000 As Reference
Set reference3000 = part1.CreateReferenceFromObject(sketch3)

Dim reference4000 As Reference
'                                    reference4                                    =
part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(GSMRot
ate.1;(Brp:(Sketch.1;6)));None:();Cf11:());Face:(Brp:(GSMRotate.1;(Brp:(Sketch.1;1)));None:();Cf
11:()));Cf11:());WithPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVer
sion_CXR15)", hybridShapeRotate1)
```

```
hybridShapeLoft1.AddSectionToLoft reference3000, 1, Nothing

Dim reference5500 As Reference
Set reference5500 = part1.CreateReferenceFromObject(hybridShapeRotate10)

Dim reference6000 As Reference
'Set reference6 = part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(Sketch.3;6);None:();Cf11:());Face:(Brp:(Sketch.3;1);None:();Cf11:()));Cf11:());WithPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVersion_CXR15)", sketch1)

hybridShapeLoft1.AddSectionToLoft reference5500, 1, Nothing


Dim reference5000 As Reference
Set reference5000 = part1.CreateReferenceFromObject(hybridShapeRotate1)

'Dim reference6000 As Reference
'Set reference6 = part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(Sketch.3;6);None:();Cf11:());Face:(Brp:(Sketch.3;1);None:();Cf11:()));Cf11:());WithPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVersion_CXR15)", sketch1)

hybridShapeLoft1.AddSectionToLoft reference5000, 1, Nothing

part1.InWorkObject = hybridShapeLoft1

part1.Update

'''''' Vamos a hacer el circular pattern

Dim ref5 As Reference
Set ref5 = part1.CreateReferenceFromName("")

Dim ref6 As Reference
Set ref6 = part1.CreateReferenceFromName("")

esp = 2 * p / d1 * 180 / pi

Dim circPattern1 As CircPattern
Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, esp, 1, 1, ref5, ref6, True, 0#, True)

'parametro 5=angulo del diente
'parametro4=numero de dientes

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

Dim angularRepartition1 As AngularRepartition
Set angularRepartition1 = circPattern1.AngularRepartition
```

```
Dim intParam1 As IntParam
Set intParam1 = angularRepartition1.InstancesCount
intParam1.Value = Z

'Dim intParam2 As IntParam
'Set intParam2 = angularRepartition1.AngularSpacing

'intParam2.Value = 360 / N


Dim shapes1 As Shapes
Set shapes1 = body1.Shapes


'Set pad2 = shapes1.Item("Pad.1")

'Set sketch2 = pad2.Sketch

circPattern1.ItemToCopy = loft1
Dim ref7 As Reference
Set ref7 = part1.CreateReferenceFromObject(axis2D3)

circPattern1.SetRotationAxis ref7

part1.UpdateObject circPattern1


''''''''''hueco interno

''''''''' hueco interno

Dim sketch5 As Sketch
Dim refernce10 As Reference
Set refernce10 = originElements1.PlaneXY
Set sketch5 = sketches1.Add(refernce10)

Dim arrayOfVariantOfDouble5(8)
arrayOfVariantOfDouble5(0) = 0#
arrayOfVariantOfDouble5(1) = 0#
arrayOfVariantOfDouble5(2) = 0#
arrayOfVariantOfDouble5(3) = 1#
arrayOfVariantOfDouble5(4) = 0#
arrayOfVariantOfDouble5(5) = 0#
arrayOfVariantOfDouble5(6) = 0#
arrayOfVariantOfDouble5(7) = 1#
arrayOfVariantOfDouble5(8) = 0#
Set sketch5Variant = sketch5
sketch5Variant.SetAbsoluteAxisData arrayOfVariantOfDouble5
part1.InWorkObject = sketch5
```

```
Dim factory2D5 As Factory2D
Set factory2D5 = sketch5.OpenEdition()
Dim geometricElements5 As GeometricElements
Set geometricElements5 = part1.GeometricElements
Dim axis2D5 As Axis2D
Set axis2D5 = geometricElements5.Item("AbsoluteAxis")
Dim line13 As line2D
Set line13 = axis2D5.GetItem("HDirection")
'line2D4.ReportName = 1
Dim line14 As line2D
Set line14 = axis2D5.GetItem("VDirection")
'line2D5.ReportName = 2


'''sketch3
Dim circlec As Circle2D
Set circlec = factory2D5.CreateClosedCircle(0, 0, rf / 4)
sketch5.CloseEdition
part1.InWorkObject = sketch5
part1.Update


''''hueeco central pocket
Dim pocket1 As Pocket
Set pocket1 = shapeFactory1.AddNewPocket(sketch5, altura)
Dim limitp2 As Limit
Set limitp2 = pocket1.FirstLimit
Dim lengthp2 As Length
Set lengthp2 = limitp2.Dimension
lengthp2.Value = altura
 part1.Update


  ' Error = MsgBox("No se puede reproducir el engranaje con los parámetros de entrada", False, "Pruebe otro valor")


End Sub
```

DOBLE HELICOIDAL

```
Private Sub CommandButton1_Click()
Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part

Dim bodies1 As Bodies
Dim body1 As Body
Dim sketches1 As Sketches
Dim originElements1 As OriginElements
```

```
Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part

Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
Set sketches1 = body1.Sketches
Set originElements1 = part1.OriginElements

Dim sketch2 As Sketch
Dim reference2 As Reference

Set reference2 = originElements1.PlaneXY
Set sketch2 = sketches1.Add(reference2)

Dim arrayOfVariantOfDouble2(8)
arrayOfVariantOfDouble2(0) = 0#
arrayOfVariantOfDouble2(1) = 0#
arrayOfVariantOfDouble2(2) = 0#
arrayOfVariantOfDouble2(3) = 1#
arrayOfVariantOfDouble2(4) = 0#
arrayOfVariantOfDouble2(5) = 0#
arrayOfVariantOfDouble2(6) = 0#
arrayOfVariantOfDouble2(7) = 1#
arrayOfVariantOfDouble2(8) = 0#
Set sketch2Variant = sketch2
sketch2Variant.SetAbsoluteAxisData arrayOfVariantOfDouble2
part1.InWorkObject = sketch2



Dim factory2d2 As Factory2D
Set factory2d2 = sketch2.OpenEdition()
Dim geometricelements2 As GeometricElements
Set geometricelements2 = part1.GeometricElements
Dim axis2d2 As Axis2D
Set axis2d2 = geometricelements2.Item("AbsoluteAxis")
Dim line2D7 As line2D
Set line2D7 = axis2d2.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D8 As line2D
Set line2D8 = axis2d2.GetItem("VDirection")
line2D8.ReportName = 2

''''''''' DATOS DE PARTIDA Y RELACIONES

''''''''' DATOS DE PARTIDA Y RELACIONES

m = TextBox1 'd1 = TextBox1  '''' diametro primitivo
Z = TextBox2
```

```
alpha = TextBox3 '''' angulo de presion
altura = 10 * m / 10 'altura = TextBox4
phi = TextBox4

pi = 4 * Math.Atn(1)
p = pi * m '* d1 / Z ''paso circunferencial
d1 = (p * Z) / (pi * m)
'm = p / pi
e = pi * m / 2 '''' Espesor de un diente curvilineo
rb = d1 / 2
rc = rb * Math.Cos(betha)
A = m 'addendum
B = 1.25 * m 'dedenddum
C = 0.25 * m 'db
ri = 0.166 * m 'radio de pie/union
h = A + B


'''radios y diametros
db = d1 * Math.Cos(alpha * pi / 180)
df = d1 - 2 * B
rb = db / 2
rf = df / 2

de = rf + 2 * h
giro = -altura * Math.Tan(phi * pi / 180) / de


Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricelements2, axis2d2,
partDocument1, part1, sketch2, originElements1, factory2d2)


sketch2.CloseEdition
part1.InWorkObject = sketch2
part1.Update

Dim sketch1 As Sketch
Dim reference1 As Reference
Set reference1 = originElements1.PlaneXY
Set sketch1 = sketches1.Add(reference1)

Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0#
arrayOfVariantOfDouble1(1) = 0#
arrayOfVariantOfDouble1(2) = 0#
arrayOfVariantOfDouble1(3) = 1#
arrayOfVariantOfDouble1(4) = 0#
arrayOfVariantOfDouble1(5) = 0#
arrayOfVariantOfDouble1(6) = 0#
arrayOfVariantOfDouble1(7) = 1#
arrayOfVariantOfDouble1(8) = 0#
Set sketch1Variant = sketch1
```

```
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
part1.InWorkObject = sketch1



Dim factory2d1 As Factory2D
Set factory2d1 = sketch1.OpenEdition()
Dim geometricelements1 As GeometricElements
Set geometricelements1 = part1.GeometricElements
Dim axis2d1 As Axis2D
Set axis2d1 = geometricelements1.Item("AbsoluteAxis")
Dim line2D1 As line2D
Set line2D1 = axis2d1.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D2 As line2D
Set line2D2 = axis2d1.GetItem("VDirection")
line2D8.ReportName = 2

''' sketch
Dim circle2d1 As Circle2D
Set circle2d1 = factory2d1.CreateClosedCircle(0, 0, rf + h)

sketch1.CloseEdition
part1.InWorkObject = sketch1
part1.Update


Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory
Dim pad1 As Pad
Set pad1 = shapeFactory1.AddNewPad(sketch1, altura)
Dim limit1 As Limit
Set limit1 = pad1.FirstLimit
Dim lengthpad1 As Length
Set lengthpad1 = limit1.Dimension
lengthpad1.Value = -altura
part1.Update




''''''plano para sketch3
'''''Dim hybridBodies1 As HybridBodies
'''''Set hybridBodies1 = part1.HybridBodies

''''Dim hybridBody1 As HybridBody
'''''Set hybridBody1 = hybridBodies1.Add()

Dim hshapefactory1 As HybridShapeFactory
Set hshapefactory1 = part1.HybridShapeFactory

Dim hybridShapePlaneExplicit1 As HybridShapePlaneExplicit
```

```
Set hybridShapePlaneExplicit1 = originElements1.PlaneXY

Dim reference10 As Reference
Set reference10 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit1)

Dim hybridShapePlaneOffset1 As HybridShapePlaneOffset
Set hybridShapePlaneOffset1 = hshapefactory1.AddNewPlaneOffset(reference10, -
altura, False)


body1.InsertHybridShape hybridShapePlaneOffset1

part1.InWorkObject = hybridShapePlaneOffset1

part1.Update

Dim hybridShapes1 As HybridShapes
Set hybridShapes1 = body1.HybridShapes

'''''sketch3

Dim reference20 As Reference
Set reference20 = hybridShapes1.Item("Plane.1")

Dim sketch3 As Sketch
Set sketch3 = sketches1.Add(reference20)

Dim arrayOfVariantOfDouble3(8)
arrayOfVariantOfDouble3(0) = 0#
arrayOfVariantOfDouble3(1) = 0#
arrayOfVariantOfDouble3(2) = -altura
arrayOfVariantOfDouble3(3) = 1#
arrayOfVariantOfDouble3(4) = 0#
arrayOfVariantOfDouble3(5) = 0#
arrayOfVariantOfDouble3(6) = 0#
arrayOfVariantOfDouble3(7) = 1#
arrayOfVariantOfDouble3(8) = 0#
Set sketch3Variant = sketch3
sketch3Variant.SetAbsoluteAxisData arrayOfVariantOfDouble3

part1.InWorkObject = sketch3

Dim factory2D3 As Factory2D
Set factory2D3 = sketch3.OpenEdition()
Dim geometricElements3 As GeometricElements
Set geometricElements3 = sketch3.GeometricElements
Dim axis2D3 As Axis2D
Set axis2D3 = geometricElements3.Item("AbsoluteAxis")
Dim line2D10 As line2D
Set line2D10 = axis2D3.GetItem("HDirection")
Dim line2D11 As line2D
Set line2D11 = axis2D3.GetItem("VDirection")
```

```
Call dienteh2.dienteh2(d1, p, rf, pi, Z, db, h, geometricElements3, axis2D3, partDocument1, part1, sketch3, originElements1, factory2D3)
sketch3.CloseEdition
part1.InWorkObject = sketch3
part1.Update


Dim hybridShapeRotate1 As HybridShapeRotate
Set hybridShapeRotate1 = hshapefactory1.AddNewEmptyRotate()

Dim referencee2 As Reference
Set referencee2 = part1.CreateReferenceFromObject(sketch2)

hybridShapeRotate1.ElemToRotate = referencee2

hybridShapeRotate1.VolumeResult = False

hybridShapeRotate1.RotationType = 0



'Dim hybridShapeLineExplicit1 As HybridShapeLineExplicit
'Set hybridShapeLineExplicit1 = hybridShapes1.Item("Z Axis")

Dim referencee3 As Reference
Set referencee3 = part1.CreateReferenceFromObject(axis2D3)

hybridShapeRotate1.Axis = referencee3

hybridShapeRotate1.AngleValue = giro * 180 / pi
body1.InsertHybridShape hybridShapeRotate1

part1.InWorkObject = hybridShapeRotate1

part1.Update

''''' borro el 1
Dim selection1 As Selection
Set selection1 = partDocument1.Selection

Dim visPropertySet1 As VisPropertySet
Set visPropertySet1 = selection1.VisProperties

Dim bSTR1 As String
bSTR1 = sketch1.Name

selection1.Add sketch2

Set visPropertySet1 = visPropertySet1.Parent

Dim bSTR2 As String
```

```
bSTR2 = visPropertySet1.Name

Dim bSTR3 As String
bSTR3 = visPropertySet1.Name

visPropertySet1.SetShow 1

selection1.Clear

part1.Update



''''plano medio

Dim hybridShapePlaneExplicit2 As HybridShapePlaneExplicit
Set hybridShapePlaneExplicit2 = originElements1.PlaneXY

Dim reference100 As Reference
Set reference100 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit2)

Dim hybridShapePlaneOffset2 As HybridShapePlaneOffset
Set hybridShapePlaneOffset2 = hshapefactory1.AddNewPlaneOffset(reference10, -
altura / 2, False)


body1.InsertHybridShape hybridShapePlaneOffset2

part1.InWorkObject = hybridShapePlaneOffset2

part1.Update

'Dim hybridShapes1 As HybridShapes
'Set hybridShapes1 = body1.HybridShapes

'''''sketch4

Dim reference200 As Reference
Set reference200 = hybridShapes1.Item("Plane.2")

Dim sketch4 As Sketch
Set sketch4 = sketches1.Add(reference200)

Dim arrayOfVariantOfDouble4(8)
arrayOfVariantOfDouble4(0) = 0#
arrayOfVariantOfDouble4(1) = 0#
arrayOfVariantOfDouble4(2) = -altura
arrayOfVariantOfDouble4(3) = 1#
arrayOfVariantOfDouble4(4) = 0#
arrayOfVariantOfDouble4(5) = 0#
```

```
arrayOfVariantOfDouble4(6) = 0#
arrayOfVariantOfDouble4(7) = 1#
arrayOfVariantOfDouble4(8) = 0#
Set sketch4Variant = sketch4
sketch4Variant.SetAbsoluteAxisData arrayOfVariantOfDouble4

part1.InWorkObject = sketch4

Dim factory2D4 As Factory2D
Set factory2D4 = sketch4.OpenEdition()
Dim geometricElements4 As GeometricElements
Set geometricElements4 = sketch4.GeometricElements
Dim axis2D4 As Axis2D
Set axis2D4 = geometricElements4.Item("AbsoluteAxis")
Dim line2D12 As line2D
Set line2D12 = axis2D4.GetItem("HDirection")
Dim line2D13 As line2D
Set line2D13 = axis2D4.GetItem("VDirection")

'Call dienteh.dienteh(d1, p, rf, pi, Z, db, h, geometricelements4, axis2d4,
partDocument1, part1, sketch4, originElements1, factory2d4)
'sketch4.CloseEdition
'part1.InWorkObject = sketch4
'part1.Update


'Dim hybridShapeRotate2 As HybridShapeRotate
'Set hybridShapeRotate2 = hshapefactory1.AddNewEmptyRotate()

'Dim referencee20 As Reference
'Set referencee20 = part1.CreateReferenceFromObject(sketch4)

'hybridShapeRotate2.ElemToRotate = referencee20

'hybridShapeRotate2.VolumeResult = False

'hybridShapeRotate2.RotationType = 0



''Dim hybridShapeLineExplicit1 As HybridShapeLineExplicit
''Set hybridShapeLineExplicit1 = hybridShapes1.Item("Z Axis")

'Dim referencee30 As Reference
'Set referencee30 = part1.CreateReferenceFromObject(axis2d3)

'hybridShapeRotate2.Axis = referencee3

'hybridShapeRotate2.AngleValue = giro * 180 / pi / 2

'body1.InsertHybridShape hybridShapeRotate2
```

```
'part1.InWorkObject = hybridShapeRotate2

'part1.Update


''' borro el 2

'Dim selection2 As Selection
'Set selection2 = partDocument1.Selection

'Dim visPropertySet2 As VisPropertySet
'Set visPropertySet2 = selection2.VisProperties

'Dim bSTR6 As String
'bSTR6 = sketch2.Name

'selection2.Add sketch4

'Set visPropertySet2 = visPropertySet2.Parent

'Dim bSTR4 As String
'bSTR4 = visPropertySet2.Name

'Dim bSTR5 As String
'bSTR5 = visPropertySet1.Name

'visPropertySet1.SetShow 1

'selection2.Clear

'part1.Update




''''''''''''''''''guias''''''''''''''''''

''''''''''''''''''GUIA1
theta7 = 42 * pi / 180
x7 = rb * (Math.Cos(theta7) + theta7 * Math.Sin(theta7))
y7 = rb * (Math.Sin(theta7) - theta7 * Math.Cos(theta7))


Xg1 = rb * (Math.Cos(theta7) + theta7 * Math.Sin(theta7))
Xg2 = rb * (Math.Cos(theta7 + giro / 2) + theta7 * Math.Sin(theta7 + giro / 2))
Xg3 = rb * (Math.Cos(theta7 + giro) + theta7 * Math.Sin(theta7 + giro))
Yg1 = rb * (Math.Sin(theta7) - theta7 * Math.Cos(theta7))
Yg2 = rb * (Math.Sin(theta7 + giro / 2) - theta7 * Math.Cos(theta7 + giro / 2))
Yg3 = rb * (Math.Sin(theta7 + giro) - theta7 * Math.Cos(theta7 + giro))
Zg1 = -altura
Zg2 = -altura / 2
Zg3 = 0
```

```
'Dim hybridShapeFactory1 As HybridShapeFactory
'Set hybridShapeFactory1 = part1.HybridShapeFactory

Dim hybridShapePointCoord1 As HybridShapePointCoord
Set hybridShapePointCoord1 = hshapefactory1.AddNewPointCoord(Xg1, Yg1, Zg1)


body1.InsertHybridShape hybridShapePointCoord1

part1.InWorkObject = hybridShapePointCoord1

part1.Update

Dim hybridShapePointCoord2 As HybridShapePointCoord
Set hybridShapePointCoord2 = hshapefactory1.AddNewPointCoord(Xg2, Yg2, Zg2)

body1.InsertHybridShape hybridShapePointCoord2

part1.InWorkObject = hybridShapePointCoord2

part1.Update

Dim hybridShapePointCoord3 As HybridShapePointCoord
Set hybridShapePointCoord3 = hshapefactory1.AddNewPointCoord(Xg3, Yg3, Zg3)

body1.InsertHybridShape hybridShapePointCoord3

part1.InWorkObject = hybridShapePointCoord3


part1.Update

Dim hybridShapeSpline1 As HybridShapeSpline
Set hybridShapeSpline1 = hshapefactory1.AddNewSpline()

hybridShapeSpline1.SetSplineType 0

hybridShapeSpline1.SetClosing 0 '

Dim refer1 As Reference
Set refer1 = part1.CreateReferenceFromObject(hybridShapePointCoord1)

hybridShapeSpline1.AddPointWithConstraintExplicit refer1, Nothing, -1#, 1, Nothing,
0#

Dim refer2 As Reference
Set refer2 = part1.CreateReferenceFromObject(hybridShapePointCoord2)

hybridShapeSpline1.AddPointWithConstraintExplicit refer2, Nothing, -1#, 1, Nothing,
0#
```

```
Dim refer3 As Reference
Set refer3 = part1.CreateReferenceFromObject(hybridShapePointCoord3)

hybridShapeSpline1.AddPointWithConstraintExplicit refer3, Nothing, -1#, 1, Nothing,
0#

body1.InsertHybridShape hybridShapeSpline1

part1.InWorkObject = hybridShapeSpline1

part1.Update


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
X0 = rf + ri
Y0 = 0

Xgg1 = X0
Ygg1 = Y0
Xgg2 = (rf + ri) * Math.Cos(giro / 2)
Ygg2 = (rf + ri) * Math.Sin(giro / 2)
Xgg3 = (rf + ri) * Math.Cos(giro)
Ygg3 = (rf + ri) * Math.Sin(giro)


Dim hybridShapePointCoord11 As HybridShapePointCoord
Set hybridShapePointCoord11 = hshapefactory1.AddNewPointCoord(Xgg1, Ygg1, Zg1)


body1.InsertHybridShape hybridShapePointCoord11

part1.InWorkObject = hybridShapePointCoord11

part1.Update

Dim hybridShapePointCoord22 As HybridShapePointCoord
Set hybridShapePointCoord22 = hshapefactory1.AddNewPointCoord(Xgg2, Ygg2, Zg2)

body1.InsertHybridShape hybridShapePointCoord22

part1.InWorkObject = hybridShapePointCoord22

part1.Update

Dim hybridShapePointCoord33 As HybridShapePointCoord
Set hybridShapePointCoord33 = hshapefactory1.AddNewPointCoord(Xgg3, Ygg3, Zg3)

body1.InsertHybridShape hybridShapePointCoord33

part1.InWorkObject = hybridShapePointCoord33
```

```
part1.Update

Dim hybridShapeSpline11 As HybridShapeSpline
Set hybridShapeSpline11 = hshapefactory1.AddNewSpline()

hybridShapeSpline11.SetSplineType 0

hybridShapeSpline11.SetClosing 0 '

Dim refer11 As Reference
Set refer11 = part1.CreateReferenceFromObject(hybridShapePointCoord11)

hybridShapeSpline11.AddPointWithConstraintExplicit refer11, Nothing, -1#, 1, Nothing,
0#

Dim refer22 As Reference
Set refer22 = part1.CreateReferenceFromObject(hybridShapePointCoord22)

hybridShapeSpline11.AddPointWithConstraintExplicit refer22, Nothing, -1#, 1, Nothing,
0#

Dim refer33 As Reference
Set refer33 = part1.CreateReferenceFromObject(hybridShapePointCoord33)

hybridShapeSpline11.AddPointWithConstraintExplicit refer33, Nothing, -1#, 1, Nothing,
0#

body1.InsertHybridShape hybridShapeSpline11

part1.InWorkObject = hybridShapeSpline11

part1.Update

'''''''''''''''''''''' hacemos el multisection remove

Dim loft1 As Loft
Set loft1 = shapeFactory1.AddNewRemovedLoft()

Dim hybridShapeLoft1 As HybridShapeLoft
Set hybridShapeLoft1 = loft1.HybridShape

hybridShapeLoft1.SectionCoupling = 3

hybridShapeLoft1.Relimitation = 1

hybridShapeLoft1.CanonicalDetection = 2


Dim reference1000 As Reference
Set reference1000 = part1.CreateReferenceFromObject(hybridShapeSpline11)

hybridShapeLoft1.AddGuide reference1000
```

```
Dim reference3000 As Reference
Set reference3000 = part1.CreateReferenceFromObject(sketch3)

Dim reference4000 As Reference
' reference4 = part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(GSMRotate.1;(Brp:(Sketch.1;6)));None:();Cf11:());Face:(Brp:(GSMRotate.1;(Brp:(Sketch.1;1)));None:();Cf11:()));Cf11:());WithPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVersion_CXR15)", hybridShapeRotate1)

hybridShapeLoft1.AddSectionToLoft reference3000, 1, Nothing


Dim reference5000 As Reference
Set reference5000 = part1.CreateReferenceFromObject(hybridShapeRotate1)

Dim reference6000 As Reference
'Set reference6 = part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(Sketch.3;6);None:();Cf11:());Face:(Brp:(Sketch.3;1);None:();Cf11:()));Cf11:());WithPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVersion_CXR15)", sketch1)

hybridShapeLoft1.AddSectionToLoft reference5000, 1, Nothing

part1.InWorkObject = hybridShapeLoft1

part1.Update

'''''' Vamos a hacer el circular pattern

Dim ref5 As Reference
Set ref5 = part1.CreateReferenceFromName("")

Dim ref6 As Reference
Set ref6 = part1.CreateReferenceFromName("")

esp = 2 * p / d1 * 180 / pi

Dim circPattern1 As CircPattern
Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, esp, 1, 1, ref5, ref6, True, 0#, True)

'parametro 5=angulo del diente
'parametro4=numero de dientes

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

Dim angularRepartition1 As AngularRepartition
```

```
Set angularRepartition1 = circPattern1.AngularRepartition

Dim intParam1 As IntParam
Set intParam1 = angularRepartition1.InstancesCount
intParam1.Value = Z

'Dim intParam2 As IntParam
'Set intParam2 = angularRepartition1.AngularSpacing

'intParam2.Value = 360 / N


Dim shapes1 As Shapes
Set shapes1 = body1.Shapes


'Set pad2 = shapes1.Item("Pad.1")

'Set sketch2 = pad2.Sketch

circPattern1.ItemToCopy = loft1
Dim ref7 As Reference
Set ref7 = part1.CreateReferenceFromObject(axis2D3)

circPattern1.SetRotationAxis ref7

part1.UpdateObject circPattern1

''''' Hacemos un mirror
'Dim referencemirror1 As Reference
'Set referencemirror1 = part1.CreateReferenceFromGeometry(pad1)
Dim referencemirror1 As Reference
Set referencemirror1 =
part1.CreateReferenceFromBRepName("FSur:(Face:(Brp:(Pad.1;1);None:();Cf11:());WithTempo
raryBody;WithoutBuildError;WithInitialFeatureSupport;MonoFond;MFBRepVersion_CXR15)",
pad1)



Dim mirror1 As Mirror
Set mirror1 = shapeFactory1.AddNewMirror(referencemirror1)
part1.Update


''''''''' hueco interno

Dim sketch5 As Sketch
Dim refernce10 As Reference
Set refernce10 = originElements1.PlaneXY
Set sketch5 = sketches1.Add(refernce10)

Dim arrayOfVariantOfDouble5(8)
```

```
arrayOfVariantOfDouble5(0) = 0#
arrayOfVariantOfDouble5(1) = 0#
arrayOfVariantOfDouble5(2) = 0#
arrayOfVariantOfDouble5(3) = 1#
arrayOfVariantOfDouble5(4) = 0#
arrayOfVariantOfDouble5(5) = 0#
arrayOfVariantOfDouble5(6) = 0#
arrayOfVariantOfDouble5(7) = 1#
arrayOfVariantOfDouble5(8) = 0#
Set sketch5Variant = sketch5
sketch5Variant.SetAbsoluteAxisData arrayOfVariantOfDouble5
part1.InWorkObject = sketch5



Dim factory2D5 As Factory2D
Set factory2D5 = sketch5.OpenEdition()
Dim geometricElements5 As GeometricElements
Set geometricElements5 = part1.GeometricElements
Dim axis2D5 As Axis2D
Set axis2D5 = geometricElements5.Item("AbsoluteAxis")
Dim line13 As line2D
Set line13 = axis2D5.GetItem("HDirection")
'line2D4.ReportName = 1
Dim line14 As line2D
Set line14 = axis2D5.GetItem("VDirection")
'line2D5.ReportName = 2


'''sketch3
Dim circlec As Circle2D
Set circlec = factory2D5.CreateClosedCircle(0, 0, rf / 4)
sketch5.CloseEdition
part1.InWorkObject = sketch5
part1.Update


''''hueeco central pocket
Dim pocket1 As Pocket
Set pocket1 = shapeFactory1.AddNewPocket(sketch5, altura)
Dim limitp2 As Limit
Set limitp2 = pocket1.FirstLimit
Dim lengthp2 As Length
Set lengthp2 = limitp2.Dimension
lengthp2.Value = altura
 part1.Update

Dim pocket2 As Pocket
Set pocket2 = shapeFactory1.AddNewPocket(sketch5, altura)
Dim limitp3 As Limit
Set limitp3 = pocket2.FirstLimit
Dim lengthp3 As Length
```

```
Set lengthp3 = limitp3.Dimension
lengthp3.Value = -altura
 part1.Update
End Sub



   EJE NERVADO.
Private Sub CommandButton1_Click()


Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part

Dim bodies1 As Bodies
Dim body1 As Body
Dim sketches1 As Sketches
Dim originElements1 As OriginElements
Dim reference1 As Reference
Dim sketch1 As Sketch

Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part

Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
Set sketches1 = body1.Sketches
Set originElements1 = part1.OriginElements
Set reference1 = originElements1.PlaneXY
Set sketch1 = sketches1.Add(reference1)

Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0#
arrayOfVariantOfDouble1(1) = 0#
arrayOfVariantOfDouble1(2) = 0#
arrayOfVariantOfDouble1(3) = 1#
arrayOfVariantOfDouble1(4) = 0#
arrayOfVariantOfDouble1(5) = 0#
arrayOfVariantOfDouble1(6) = 0#
arrayOfVariantOfDouble1(7) = 1#
arrayOfVariantOfDouble1(8) = 0#
Set sketch1Variant = sketch1
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
part1.InWorkObject = sketch1
```

```
Dim factory2d1 As Factory2D
Set factory2d1 = sketch1.OpenEdition()
Dim geometricelements1 As GeometricElements
Set geometricelements1 = part1.GeometricElements
Dim axis2d1 As Axis2D
Set axis2d1 = geometricelements1.Item("AbsoluteAxis")
Dim line2D1 As line2D
Set line2D1 = axis2d1.GetItem("HDirection")
Dim line2D2 As line2D
Set line2D2 = axis2d1.GetItem("VDirection")


''''''''' DATOS DE PARTIDA Y RELACIONES

d1 = TextBox1  '''' diametro primitivo
d2 = TextBox2
d3 = TextBox3 '''' angulo de presion
e = TextBox4
f = TextBox5
g = TextBox6
k = TextBox7
r = TextBox8


pi = 4 * Math.Atn(1)

Dim point2D1 As Point2D
Set point2D1 = axis2d1.GetItem("Origin")

angulo2 = (f + (2 * e)) / 2 / (d1 / 2)
theta = g / d2 / 2

Dim circle2d1 As Circle2D
Set circle2d1 = factory2d1.CreateCircle(0, 0, d1 / 2, -angulo2, angulo2)
circle2d1.CenterPoint = point2D1

Dim circle2d2 As Circle2D
Set circle2d2 = factory2d1.CreateCircle(0, 0, d2 / 2, -angulo2 - theta, angulo2 + theta)
circle2d2.CenterPoint = point2D1

px1 = d2 / 2 * Math.Cos(angulo2 + theta)
py1 = d2 / 2 * Math.Sin(angulo2 + theta)
px4 = d2 / 2 * Math.Cos(-angulo2 - theta)
py4 = d2 / 2 * Math.Sin(-angulo2 - theta)
px2 = d1 / 2 * Math.Cos(angulo2)
py2 = d1 / 2 * Math.Sin(angulo2)
px5 = d1 / 2 * Math.Cos(-angulo2)
py5 = d1 / 2 * Math.Sin(-angulo2)
px3 = (d2 / 2 - k) * Math.Cos(angulo2)
py3 = (d2 / 2 - k) * Math.Sin(angulo2)
px6 = (d2 / 2 - k) * Math.Cos(-angulo2)
py6 = (d2 / 2 - k) * Math.Sin(-angulo2)
```

```
Dim point1 As Point2D
Set point1 = factory2d1.CreatePoint(px1, py1)
Dim point2 As Point2D
Set point2 = factory2d1.CreatePoint(px2, py2)
Dim point3 As Point2D
Set point3 = factory2d1.CreatePoint(px3, py3)
Dim point4 As Point2D
Set point4 = factory2d1.CreatePoint(px4, py4)
Dim point5 As Point2D
Set point5 = factory2d1.CreatePoint(px5, py5)
Dim point6 As Point2D
Set point6 = factory2d1.CreatePoint(px6, py6)


Dim line1 As line2D
Set line1 = factory2d1.CreateLine(px1, py1, px3, py3)
line1.StartPoint = point1
line1.EndPoint = point3

Dim line2 As line2D
Set line2 = factory2d1.CreateLine(px4, py4, px6, py6)
line2.StartPoint = point4
line2.EndPoint = point6

Dim line3 As line2D
Set line3 = factory2d1.CreateLine(px3, py3, px2, py2)
line3.StartPoint = point3
line3.EndPoint = point2

Dim line4 As line2D
Set line4 = factory2d1.CreateLine(px6, py6, px5, py5)
line4.StartPoint = point6
line4.EndPoint = point5

Dim constraints1 As Constraints
Set constraints1 = part1.Constraints

Dim ref11 As Reference
Set ref11 = part1.CreateReferenceFromObject(circle2d1)
Dim constraint7 As Constraint
Set constraint7 = constraints1.AddMonoEltCst(catCstTypeRadius, ref11)
constraint7.Mode = catCstModeDrivingDimension
Dim length7 As Length
Set length7 = constraint7.Dimension
length7.Value = d1 / 2

Dim ref12 As Reference
Set ref12 = part1.CreateReferenceFromObject(circle2d2)
Dim constraint8 As Constraint
Set constraint8 = constraints1.AddMonoEltCst(catCstTypeRadius, ref12)
```

```
constraint8.Mode = catCstModeDrivingDimension
Dim length8 As Length
Set length8 = constraint8.Dimension
length8.Value = d2 / 2


sketch1.CloseEdition
part1.InWorkObject = sketch1
part1.Update


''''''''' cilindro central
Dim reference2 As Reference
Dim sketch2 As Sketch
Set reference2 = originElements1.PlaneXY
Set sketch2 = sketches1.Add(reference2)

Dim arrayOfVariantOfDouble2(8)
arrayOfVariantOfDouble2(0) = 0#
arrayOfVariantOfDouble2(1) = 0#
arrayOfVariantOfDouble2(2) = 0#
arrayOfVariantOfDouble2(3) = 1#
arrayOfVariantOfDouble2(4) = 0#
arrayOfVariantOfDouble2(5) = 0#
arrayOfVariantOfDouble2(6) = 0#
arrayOfVariantOfDouble2(7) = 1#
arrayOfVariantOfDouble2(8) = 0#
Set sketch2Variant = sketch2
sketch2Variant.SetAbsoluteAxisData arrayOfVariantOfDouble2
part1.InWorkObject = sketch2




Dim factory2d2 As Factory2D
Set factory2d2 = sketch2.OpenEdition()
Dim geometricelements2 As GeometricElements
Set geometricelements2 = part1.GeometricElements
Dim axis2d2 As Axis2D
Set axis2d2 = geometricelements2.Item("AbsoluteAxis")
Dim line2D3 As line2D
Set line2D3 = axis2d2.GetItem("HDirection")
Dim line2D4 As line2D
Set line2D4 = axis2d2.GetItem("VDirection")

Dim point00 As Point2D
Set point00 = factory2d2.CreatePoint(0, 0)
Dim circlep As Circle2D
Set circlep = factory2d2.CreateClosedCircle(0, 0, d2 / 2)
circlep.CenterPoint = point00
```

```
sketch2.CloseEdition
part1.InWorkObject = sketch2
part1.Update


''''''' pad1
Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory

Dim pad1 As Pad
Set pad1 = shapeFactory1.AddNewPad(sketch2, 2)
Dim limit1 As Limit
Set limit1 = pad1.FirstLimit
Dim lengthp As Length
Set lengthp = limit1.Dimension
lengthp.Value = 20

part1.Update

''''''''pocket1

Dim pocket1 As Pocket
Set pocket1 = shapeFactory1.AddNewPocket(sketch1, 2)
Dim limit2 As Limit
Set limit2 = pocket1.FirstLimit
Dim lenghtpk As Length
Set lenghtpk = limit1.Dimension
lenghtpk.Value = -2

part1.Update


Dim reference3 As Reference
Dim sketch3 As Sketch
Set reference3 = originElements1.PlaneXY
Set sketch3 = sketches1.Add(reference3)

Dim arrayOfVariantOfDouble3(8)
arrayOfVariantOfDouble3(0) = 0#
arrayOfVariantOfDouble3(1) = 0#
arrayOfVariantOfDouble3(2) = 0#
arrayOfVariantOfDouble3(3) = 1#
arrayOfVariantOfDouble3(4) = 0#
arrayOfVariantOfDouble3(5) = 0#
arrayOfVariantOfDouble3(6) = 0#
arrayOfVariantOfDouble3(7) = 1#
arrayOfVariantOfDouble3(8) = 0#
Set sketch3Variant = sketch3
sketch3Variant.SetAbsoluteAxisData arrayOfVariantOfDouble3
part1.InWorkObject = sketch3
```

```
Dim factory2D3 As Factory2D
Set factory2D3 = sketch3.OpenEdition()
Dim geometricElements3 As GeometricElements
Set geometricElements3 = part1.GeometricElements
Dim axis2D3 As Axis2D
Set axis2D3 = geometricElements3.Item("AbsoluteAxis")
Dim line2D5 As line2D
Set line2D5 = axis2D3.GetItem("HDirection")
Dim line2D6 As line2D
Set line2D6 = axis2D3.GetItem("VDirection")

angulo1 = (f / 2 + e / 2) / (d1 / 2)
X = (d1 / 2 + r) * Math.Cos(angulo1)
Y = (d1 / 2 + r) * Math.Sin(angulo1)
x1 = (d1 / 2 + r) * Math.Cos(-angulo1)
y1 = (d1 / 2 + r) * Math.Sin(-angulo1)

Dim pointaux1 As Point2D
Set pointaux1 = factory2D3.CreatePoint(X, Y)
Dim pointaux2 As Point2D
Set pointaux2 = factory2D3.CreatePoint(x1, y1)


Dim circle2d4 As Circle2D
Set circle2d4 = factory2D3.CreateClosedCircle(X, Y, e / 2)
circle2d4.CenterPoint = pointaux1

Dim circle2d5 As Circle2D
Set circle2d5 = factory2D3.CreateClosedCircle(x1, y1, e / 2)
circle2d5.CenterPoint = pointaux2


sketch3.CloseEdition
part1.InWorkObject = sketch3
part1.Update

''''' pocket2
Dim pocket2 As Pocket
Set pocket2 = shapeFactory1.AddNewPocket(sketch3, 2)
Dim limit3 As Limit
Set limit3 = pocket2.FirstLimit
Dim lengthpk2 As Length
Set lengthpk2 = limit3.Dimension
lengthpk2.Value = 2

part1.Update

Z = 6
```

```
''''''''''' circular pattern1
Dim ref5 As Reference
Set ref5 = part1.CreateReferenceFromName("")

Dim ref6 As Reference
Set ref6 = part1.CreateReferenceFromName("")



Dim circPattern1 As CircPattern
Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, 360 / Z, 1, 1,
ref5, ref6, True, 0#, True)

'parametro 5=angulo del diente
'parametro4=numero de dientes

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

Dim angularRepartition1 As AngularRepartition
Set angularRepartition1 = circPattern1.AngularRepartition

Dim intParam1 As IntParam
Set intParam1 = angularRepartition1.InstancesCount
intParam1.Value = Z

'Dim intParam2 As IntParam
'Set intParam2 = angularRepartition1.AngularSpacing

'intParam2.Value = 360 / N


Dim shapes1 As Shapes
Set shapes1 = body1.Shapes

Set pocket1 = shapes1.Item("Pocket.1")

circPattern1.ItemToCopy = pocket1


Dim ref7 As Reference
Set ref7 = part1.CreateReferenceFromObject(axis2D3)

circPattern1.SetRotationAxis ref7

part1.UpdateObject circPattern1


'''''' circular pattern2

Dim refe5 As Reference
```

```
        Set refe5 = part1.CreateReferenceFromName("")

        Dim refe6 As Reference
        Set refe6 = part1.CreateReferenceFromName("")


        Z = 6

        Dim circPattern2 As CircPattern
        Set circPattern2 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, 360 / Z, 1, 1,
refe5, refe6, True, 0#, True)

        'parametro 5=angulo del diente
        'parametro4=numero de dientes

        circPattern2.CircularPatternParameters = catInstancesandAngularSpacing

        circPattern2.CircularPatternParameters = catInstancesandAngularSpacing

        Dim angularRepartition2 As AngularRepartition
        Set angularRepartition2 = circPattern2.AngularRepartition

        Dim intParam2 As IntParam
        Set intParam2 = angularRepartition2.InstancesCount
        intParam2.Value = Z

        'Dim intParam2 As IntParam
        'Set intParam2 = angularRepartition1.AngularSpacing

        'intParam2.Value = 360 / N


        'Dim shapes1 As Shapes
        'Set shapes1 = body1.Shapes

        Set pocket2 = shapes1.Item("Pocket.2")

        circPattern2.ItemToCopy = pocket2


        Dim refe7 As Reference
        Set refe7 = part1.CreateReferenceFromObject(axis2D3)

        circPattern2.SetRotationAxis refe7

        part1.UpdateObject circPattern2
```

```
End Sub
```

CONICO.

```
Private Sub CommandButton1_Click()


' Arranque sin necesidad de tener un part abierto

Dim documents1 As Documents
Dim partDocument1 As PartDocument
Dim part1 As Part
Dim bodies1 As Bodies
Dim body1 As Body
Dim sketches1 As Sketches
Dim originElements1 As OriginElements
Dim reference1 As Reference
Dim sketch1 As Sketch

Set documents1 = CATIA.Documents
Set partDocument1 = documents1.Add("Part")
Set part1 = partDocument1.Part
Set bodies1 = part1.Bodies
Set body1 = bodies1.Item("PartBody")
Set sketches1 = body1.Sketches
Set originElements1 = part1.OriginElements
Set reference1 = originElements1.PlaneZX
Set sketch1 = sketches1.Add(reference1)

Dim arrayOfVariantOfDouble1(8)
arrayOfVariantOfDouble1(0) = 0#
arrayOfVariantOfDouble1(1) = 0#
arrayOfVariantOfDouble1(2) = 0#
arrayOfVariantOfDouble1(3) = 1#
arrayOfVariantOfDouble1(4) = 0#
arrayOfVariantOfDouble1(5) = 0#
arrayOfVariantOfDouble1(6) = 0#
arrayOfVariantOfDouble1(7) = 1#
arrayOfVariantOfDouble1(8) = 0#
Set sketch1Variant = sketch1
sketch1Variant.SetAbsoluteAxisData arrayOfVariantOfDouble1
part1.InWorkObject = sketch1

Dim factory2d1 As Factory2D
Set factory2d1 = sketch1.OpenEdition()
```

```
Dim geometricelements1 As GeometricElements
Set geometricelements1 = sketch1.GeometricElements
Dim axis2d1 As Axis2D
Set axis2d1 = geometricelements1.Item("AbsoluteAxis")
Dim line2D1 As line2D
Set line2D1 = axis2d1.GetItem("HDirection")
line2D1.ReportName = 1
Dim line2D2 As line2D
Set line2D2 = axis2d1.GetItem("VDirection")
line2D2.ReportName = 2


''''' geometria del engranaje
d1sup = TextBox1  '''' diametro primitivo
d1inf = TextBox2
Z = TextBox3 '''' angulo de presion
alpha = TextBox4
altura = TextBox5


'''''''''''' superior

pi = 4 * Math.Atn(1)
'betha = (pi / 2) / Z '' Z=numero de dientes
psup = pi * d1sup / Z ''paso circunferencial
msup = psup / pi
esup = pi * msup / 2 '''' Espesor de un diente curvilineo
rbsup = d1sup / 2
'rc = rb * Math.Cos(betha)
Asup = msup 'addendum
Bsup = 1.25 * msup 'dedenddum
Csup = 0.25 * msup 'db
risup = 0.166 * msup 'radio de pie/union
hsup = Asup + Bsup


'''radios y diametros
dbsup = d1sup * Math.Cos(alpha * pi / 180)
dfsup = d1sup - 2 * Bsup
rbsup = dbsup / 2
rfsup = dfsup / 2

'''''''''''' inferior

pi = 4 * Math.Atn(1)
'betha = (pi / 2) / Z '' Z=numero de dientes
pinf = pi * d1inf / Z ''paso circunferencial
minf = pinf / pi
einf = pi * minf / 2 '''' Espesor de un diente curvilineo
rbinf = d1inf / 2
'rc = rb * Math.Cos(betha)
Ainf = minf 'addendum
```

```
Binf = 1.25 * minf 'dedenddum
Cinf = 0.25 * minf 'db
riinf = 0.166 * minf 'radio de pie/union
hinf = Ainf + Binf


'''radios y diametros
dbinf = d1inf * Math.Cos(alpha * pi / 180)
dfinf = d1inf - 2 * Binf
rbinf = dbinf / 2
rfinf = dfinf / 2



radiosup = rfsup + hsup
radioinf = rfinf + hinf
'ang_primitivo = Math.Tan(alpha)
'R = d1 / (2 * Math.Sin(ang_primitivo))



''''' sketch1 para el shaft

Dim point2D1 As Point2D
Set point2D1 = factory2d1.CreatePoint(0, 0)

Dim point2D2 As Point2D
Set point2D2 = factory2d1.CreatePoint(0, altura)

Dim point2D3 As Point2D
Set point2D3 = factory2d1.CreatePoint(radiosup, altura)

Dim point2D4 As Point2D
Set point2D4 = factory2d1.CreatePoint(radioinf, 0)

Dim line2D3 As line2D
Set line2D3 = factory2d1.CreateLine(0, 0, 0, altura)
line2D3.StartPoint = point2D1
line2D3.EndPoint = point2D2

Dim line2D4 As line2D
Set line2D4 = factory2d1.CreateLine(0, altura, radiosup, altura)
line2D4.StartPoint = point2D2
line2D4.EndPoint = point2D3

Dim line2D5 As line2D
Set line2D5 = factory2d1.CreateLine(radiosup, altura, radioinf, 0)
line2D5.StartPoint = point2D3
line2D5.EndPoint = point2D4

Dim line2D6 As line2D
Set line2D6 = factory2d1.CreateLine(radioinf, 0, 0, 0)
```

```
line2D6.StartPoint = point2D4
line2D6.EndPoint = point2D1

sketch1.CloseEdition
part1.InWorkObject = sketch1
part1.Update


Dim shapeFactory1 As ShapeFactory
Set shapeFactory1 = part1.ShapeFactory

Dim shaft1 As Shaft
Set shaft1 = shapeFactory1.AddNewShaft(sketch1)
' Se define el ángulo de revolución

Dim angle1 As Angle
Set angle1 = shaft1.FirstAngle
angle1.Value = 360 ' Valor del ángulo introducido en el menú

Dim parameters1 As Parameters
Set parameters1 = part1.Parameters


' Se establece el eje de revolución

Dim reference12 As Reference
Set reference12 = part1.CreateReferenceFromObject(line2D2)

shaft1.RevoluteAxis = reference12
part1.Update

''''''''PLANO

Dim hybridShapeFactory1 As HybridShapeFactory
Set hybridShapeFactory1 = part1.HybridShapeFactory


Dim hybridShapePlaneExplicit1 As HybridShapePlaneExplicit
Set hybridShapePlaneExplicit1 = originElements1.PlaneXY

Dim reference10 As Reference
Set reference10 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit1)

Dim hybridShapePlaneOffset1 As HybridShapePlaneOffset
Set hybridShapePlaneOffset1 = hybridShapeFactory1.AddNewPlaneOffset(reference10,
-altura, False)


body1.InsertHybridShape hybridShapePlaneOffset1

part1.InWorkObject = hybridShapePlaneOffset1
```

```
part1.Update



Dim hybridShapes1 As HybridShapes
Set hybridShapes1 = body1.HybridShapes

Dim sketch2 As Sketch
Dim reference20 As Reference
Set reference20 = hybridShapes1.Item("Plane.1")

Set sketch2 = sketches1.Add(reference20)

Dim arrayOfVariantOfDouble2(8)
arrayOfVariantOfDouble2(0) = 0#
arrayOfVariantOfDouble2(1) = 0#
arrayOfVariantOfDouble2(2) = 0#
arrayOfVariantOfDouble2(3) = 1#
arrayOfVariantOfDouble2(4) = 0#
arrayOfVariantOfDouble2(5) = 0#
arrayOfVariantOfDouble2(6) = 0#
arrayOfVariantOfDouble2(7) = 1#
arrayOfVariantOfDouble2(8) = 0#
Set sketch2Variant = sketch2
sketch2Variant.SetAbsoluteAxisData arrayOfVariantOfDouble2

part1.InWorkObject = sketch2


''''SKETCH2

Dim factory2d2 As Factory2D
Set factory2d2 = sketch2.OpenEdition()
Dim geometricelements2 As GeometricElements
Set geometricelements2 = sketch2.GeometricElements
Dim axis2d2 As Axis2D
Set axis2d2 = geometricelements2.Item("AbsoluteAxis")
Dim line7 As line2D
Set line7 = axis2d2.GetItem("HDirection")
line7.ReportName = 1
Dim line8 As line2D
Set line8 = axis2d2.GetItem("VDirection")
line8.ReportName = 2



Call dienteh.dienteh(d1sup, psup, rfsup, pi, Z, dbsup, hsup, geometricelements2,
axis2d2, partDocument1, part1, sketch2, originElements1, factory2d2)
sketch2.CloseEdition
part1.InWorkObject = sketch2
part1.Update
```

```
''''''' segundo diente para el multisection



Dim hybridShapePlaneExplicit2 As HybridShapePlaneExplicit
Set hybridShapePlaneExplicit2 = originElements1.PlaneXY

Dim reference11 As Reference
Set reference11 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit2)

Dim hybridShapePlaneOffset2 As HybridShapePlaneOffset
Set hybridShapePlaneOffset2 = hybridShapeFactory1.AddNewPlaneOffset(reference11,
0, False)


body1.InsertHybridShape hybridShapePlaneOffset2

part1.InWorkObject = hybridShapePlaneOffset2

part1.Update




Dim sketch3 As Sketch
Dim reference17 As Reference
Set reference17 = hybridShapes1.Item("Plane.2")
Set sketch3 = sketches1.Add(reference17)

Dim arrayOfVariantOfDouble3(8)
arrayOfVariantOfDouble3(0) = 0#
arrayOfVariantOfDouble3(1) = 0#
arrayOfVariantOfDouble3(2) = 0#
arrayOfVariantOfDouble3(3) = 1#
arrayOfVariantOfDouble3(4) = 0#
arrayOfVariantOfDouble3(5) = 0#
arrayOfVariantOfDouble3(6) = 0#
arrayOfVariantOfDouble3(7) = 1#
arrayOfVariantOfDouble3(8) = 0#
Set sketch3Variant = sketch3
sketch3Variant.SetAbsoluteAxisData arrayOfVariantOfDouble3

part1.InWorkObject = sketch3

Dim factory2D3 As Factory2D
Set factory2D3 = sketch3.OpenEdition()
Dim geometricElements3 As GeometricElements
Set geometricElements3 = sketch3.GeometricElements
Dim axis2D3 As Axis2D
```

```
Set axis2D3 = geometricElements3.Item("AbsoluteAxis")
Dim line2D7 As line2D
Set line2D7 = axis2D3.GetItem("HDirection")
line2D7.ReportName = 1
Dim line2D8 As line2D
Set line2D8 = axis2D3.GetItem("VDirection")
line2D8.ReportName = 2



Call dienteh.dienteh(d1inf, pinf, rfinf, pi, Z, dbinf, hinf, geometricElements3, axis2D3,
partDocument1, part1, sketch3, originElements1, factory2D3)
sketch3.CloseEdition
part1.InWorkObject = sketch3
part1.Update

''''''multiseccion remove

Dim loft1 As Loft
Set loft1 = shapeFactory1.AddNewRemovedLoft()

Dim hybridShapeLoft1 As HybridShapeLoft
Set hybridShapeLoft1 = loft1.HybridShape

hybridShapeLoft1.SectionCoupling = 3

hybridShapeLoft1.Relimitation = 1

hybridShapeLoft1.CanonicalDetection = 2


'Dim reference1000 As Reference
'Set reference1000 = part1.CreateReferenceFromObject(hybridShapeSpline1)

'hybridShapeLoft1.AddGuide reference1000


Dim reference3000 As Reference
Set reference3000 = part1.CreateReferenceFromObject(sketch3)

Dim reference4000 As Reference
'                                    reference4                                    =
part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(GSMRot
ate.1;(Brp:(Sketch.1;6)));None:();Cf11:());Face:(Brp:(GSMRotate.1;(Brp:(Sketch.1;1)));None:();Cf
11:()));Cf11:());WithPermanentBody;WithoutBuildError;WithInitialFeatureSupport;MFBRepVer
sion_CXR15)", hybridShapeRotate1)

hybridShapeLoft1.AddSectionToLoft reference3000, 1, Nothing


Dim reference5000 As Reference
Set reference5000 = part1.CreateReferenceFromObject(sketch2)
```

```
        Dim reference6000 As Reference
        'Set                                    reference6                                    =
part1.CreateReferenceFromBRepName("WireFVertex:(Vertex:(Neighbours:(Face:(Brp:(Sketch.
3;6);None:();Cf11:());Face:(Brp:(Sketch.3;1);None:();Cf11:()));Cf11:());WithPermanentBody;Wit
houtBuildError;WithInitialFeatureSupport;MFBRepVersion_CXR15)", sketch1)

        hybridShapeLoft1.AddSectionToLoft reference5000, 1, Nothing

        part1.InWorkObject = hybridShapeLoft1

        part1.Update

        ''''''' Vamos a hacer el circular pattern

        Dim ref5 As Reference
        Set ref5 = part1.CreateReferenceFromName("")

        Dim ref6 As Reference
        Set ref6 = part1.CreateReferenceFromName("")

        esp = 2 * psup / d1sup * 180 / pi

        Dim circPattern1 As CircPattern
        Set circPattern1 = shapeFactory1.AddNewCircPattern(Nothing, 1, 2, 20#, esp, 1, 1,
ref5, ref6, True, 0#, True)

        'parametro 5=angulo del diente
        'parametro4=numero de dientes

        circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

        circPattern1.CircularPatternParameters = catInstancesandAngularSpacing

        Dim angularRepartition1 As AngularRepartition
        Set angularRepartition1 = circPattern1.AngularRepartition

        Dim intParam1 As IntParam
        Set intParam1 = angularRepartition1.InstancesCount
        intParam1.Value = Z

        'Dim intParam2 As IntParam
        'Set intParam2 = angularRepartition1.AngularSpacing

        'intParam2.Value = 360 / N


        Dim shapes1 As Shapes
        Set shapes1 = body1.Shapes


        'Set pad2 = shapes1.Item("Pad.1")
```

```
'Set sketch2 = pad2.Sketch

circPattern1.ItemToCopy = loft1
Dim ref7 As Reference
Set ref7 = part1.CreateReferenceFromObject(axis2D3)

circPattern1.SetRotationAxis ref7

part1.UpdateObject circPattern1


''''' hueco interno

Dim sketch4 As Sketch
Dim reference18 As Reference
Set reference18 = originElements1.PlaneXY
Set sketch4 = sketches1.Add(reference18)

Dim arrayOfVariantOfDouble4(8)
arrayOfVariantOfDouble4(0) = 0#
arrayOfVariantOfDouble4(1) = 0#
arrayOfVariantOfDouble4(2) = 0#
arrayOfVariantOfDouble4(3) = 1#
arrayOfVariantOfDouble4(4) = 0#
arrayOfVariantOfDouble4(5) = 0#
arrayOfVariantOfDouble4(6) = 0#
arrayOfVariantOfDouble4(7) = 1#
arrayOfVariantOfDouble4(8) = 0#
Set sketch4Variant = sketch4
sketch4Variant.SetAbsoluteAxisData arrayOfVariantOfDouble4

part1.InWorkObject = sketch4

Dim factory2D4 As Factory2D
Set factory2D4 = sketch4.OpenEdition()
Dim geometricElements4 As GeometricElements
Set geometricElements4 = sketch4.GeometricElements
Dim axis2D4 As Axis2D
Set axis2D4 = geometricElements4.Item("AbsoluteAxis")
Dim line2D10 As line2D
Set line2D10 = axis2D4.GetItem("HDirection")
line2D10.ReportName = 1
Dim line2D11 As line2D
Set line2D11 = axis2D4.GetItem("VDirection")
line2D11.ReportName = 2

If rfinf > rfsup Then
rfp2 = rfsup
Else
rfp2 = rfinf
End If
```

```
Dim circlep2 As Circle2D
Set circlep2 = factory2D4.CreateClosedCircle(0, 0, rfp2 / 3)

Dim pocket2 As Pocket
Set pocket2 = shapeFactory1.AddNewPocket(sketch4, altura)
Dim limitp2 As Limit
Set limitp2 = pocket2.FirstLimit
Dim lengthp2 As Length
Set lengthp2 = limitp2.Dimension
lengthp2.Value = altura

part1.Update




End Sub
```

9 Referencias

[1] Varios autores. Engranaje [En línea]. Wikipedia la enciclopedia Libre.

[2] Justo Albarrán Ligero. Fundamentos del KBE (Knowledge Based Engineering). Aplicación al diseño de engranajes de ejes paralelos con Catia v5. Proyecto fin de carrera.Sevilla. Escuela Superior de Ingenieros,2008

[3]Joseph E.Shigley. Charles R.Mischke. Diseño en ingeniería mecánica. Sexta edición.Editorial Mc Graw Hill.

[4] Cristina Torrecillas. Introducción a la programación Visual Basic en CATIA V5 y V6 [No publicado].

[5] Emmett Ross. VB Scripting for CATIA V5.Segunda edición.

[7] Cecil Jensen. Jay D.Helsel. Dennis R.Short. Dibujo y diseño en ingeniería. Sexta edición. Editorial Mc Graw Hill.