Trabajo Fin de Grado

Estrategias de posicionamiento y orientación de piezas en fabricación aditiva considerando el modelo cinemático de la máquina

Autor

Juan Alfonso González Josa

Directora

Mª José Oliveros Colay

Codirector

Jorge Santolaria Mazo

Escuela de Ingeniería y Arquitectura

2015

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

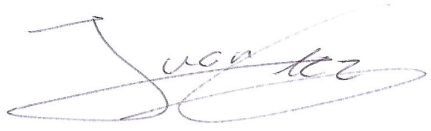
D./Dª. Juan Alfonso González Josa,

con nº de DNI 73131579C en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Estrategias de posicionamiento y orientación de piezas en fabricación aditiva considerando el modelo cinemático de la máquina,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 7 de septiembre del 2015



Fdo: Juan Alfonso González Josa

Estrategias de posicionamiento y orientación de piezas en fabricación aditiva considerando el modelo cinemático de la máquina

RESUMEN

El objetivo principal de este trabajo de fin de grado consiste en determinar las mejores estrategias de posicionamiento y orientación de una pieza en fabricación aditiva considerando el modelo cinemático de error de la máquina, para optimizar el valor de ciertas tolerancias dimensionales y de forma. Por ello, resulta necesario desarrollar un algoritmo de optimización con el que se puedan encontrar estos valores óptimos. De esta manera, será posible en un futuro, incluir un módulo que permita al fabricante tomar una decisión a la hora de fabricar una pieza en fabricación aditiva teniendo en cuenta las tolerancias exigidas por el diseñador.

En este tipo de fabricación, debido a los errores producidos por la propia tecnología y los errores debidos tanto al movimiento del cabezal como a la bandeja de la impresora, es fundamental definir cuál es la posición y orientación óptima de la pieza para que ésta cumpla las restricciones exigidas obteniendo en mínimo error en la pieza y cumpliendo con las tolerancias.

Debido a la enorme cantidad de tolerancias tanto geométricas como de forma, se ha definido un algoritmo en el cual se puede variar la tolerancia a optimizar, en función de las necesidades de la pieza. En esta primera aproximación, el algoritmo de optimización sólo tiene en cuenta errores debidos a la estructura cinemática de la máquina.

El primer paso a la hora de desarrollar el algoritmo de optimización teniendo en cuenta el modelo cinemático de error de la máquina es definir este modelo para poder introducirlo posteriormente en el algoritmo. Para definir el modelo cinemático se fabrican una misma pieza en diferentes posiciones del volumen de trabajo de la máquina y posteriormente se realizan una serie de mediciones que nos permiten aproximar el movimiento de la impresora mediante unos polinomios de Legendre. Debido a la complejidad y tiempo necesario para realizar este proceso se ha empleado un modelo ya definido de la impresora Objet Eden 350. Este modelo cinemático ha sido definido en un rango del volumen de trabajo de la impresora, no en todo su volumen.

Tras realizar el modelo cinemático de la máquina, es turno del algoritmo de optimización. El algoritmo empleado es un algoritmo de hormigas, el cual parte de 6 variables independientes (3 de posición y 3 giros) y mediante iteraciones va obteniendo diferentes estrategias de posición y orientación de la pieza que optimizan el objetivo fijado. En el caso que nos ocupa el objetivo es variable en función de la tolerancia con que vayamos a trabajar.

Una vez determinado el modelo cinemático y realizada la puesta a punto del algoritmo se procede a la experimentación. En este trabajo se ha trabajado con diversos objetivos, tolerancias, por lo que se han diseñado únicamente dos experimentos para cada una de las tolerancias debido a al tiempo computacional necesario para realizar cada uno de ellos. Tras realizar la experimentación se obtienen los resultados que nos permitirán comparar cada experimento y obtener las conclusiones generales del trabajo.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ECUACIONES

1 OBJETO Y ALCANCE DEL PROYECTO

1.1 ANTECEDENTES

En la actualidad la Fabricación Aditiva supone una revolución industrial gracias a la facilidad de producir geometrías complejas y personalizables sin aumentar el coste de fabricación.

Estudios sobre este proceso de fabricación trabajan en como orientar la pieza dentro del volumen de trabajo de la máquina para minimizar errores dimensionales. En este proyecto, y como novedad frente a otros trabajos, se tienen en cuenta los errores del equipo así como la posición que ocupa la pieza dentro del volumen de trabajo de la impresora.

Este proyecto surge debido al gran desarrollo que están teniendo hoy día las máquinas de fabricación aditiva así como de la necesidad de fabricar piezas con mejores tolerancias tanto dimensionales como geométricas.

1.2 OBJETO

El objeto de este proyecto es desarrollar un software que permita al fabricante conocer cuál es el posicionamiento y la orientación óptima de la pieza, en el volumen de trabajo de la máquina, para conseguir fabricarla cumpliendo las restricciones exigidas en cada una de las piezas.

El software que se va a desarrollar está formado por diversos módulos que permiten adaptarlo a las características de la pieza y de la máquina, es decir, en función de las tolerancias que deba cumplir la pieza se emplean unos módulos u otros. Del mismo modo, es posible adaptar el software a cada tipo de impresora en función de su modelo cinemático de error. Los módulos que van a constituir el software son: el algoritmo de optimización, el modelo cinemático de la máquina y la tolerancia.

1.3 ALCANCE

El proyecto se centra en la obtención de diversas estrategias de posicionamiento y orientación de piezas en fabricación aditiva teniendo en cuenta los errores inherentes de la máquina. Para ello se han llevado a cabo las siguientes fases:

En primer lugar, se ha realizado una documentación a partir de la bibliografía actual sobre optimización en máquinas de fabricación aditiva observándose que en ningún artículo se introducen los errores producidos por la propia máquina.

Posteriormente, se ha revisado y ajustado el modelo cinemático de la máquina, el cual está definido mediante polinomios de Legendre en un volumen determinado de ésta.

A continuación, se ha programado y validado el algoritmo de optimización habiendo definido previamente los objetivos y las variables independientes. Para validar el algoritmo se ha empleado una pieza sencilla y se ha realizado una simplificación del modelo de error que permite poder predecir el resultado que se debe obtener.

Tras la validación del algoritmo, se han introducido restricciones dimensionales y tecnológicas y se ha llevado a cabo la experimentación y el análisis de resultados.

1.4 RESUMEN DE LA MEMORIA

En este primer capítulo se ha explicado el porqué de este trabajo, así como los objetivos planteados y las diferentes fases en las que éste se desarrolla.

A continuación, en el capítulo 2, se va realizar una pequeña introducción sobre la Fabricación Aditiva y se va a presentar la documentación empleada explicando las novedades de este proyecto frente a trabajos previos.

En el capítulo 3 se van a exponer las diferentes etapas del proceso y se explicará en detalle en funcionamiento del algoritmo de optimización empleado.

Acto seguido, se detallan los componentes implicados en el proyecto, es decir, la pieza empleada para la experimentación y el modelo cinemático de la impresora 3D.

Previo a la experimentación y los resultados, se explica el proceso de verificación del algoritmo, así como los parámetros y las modificaciones que lo han adaptado al proyecto en cuestión.

Por último se van a presentar os resultados obtenidos durante la experimentación de cada una de las tolerancias y se finalizará con las conclusiones generales y la líneas futuras del proyecto.

2 INTRODUCCIÓN: OPTIMIZACIÓN EN FABRICACIÓN ADITIVA

La Fabricación Aditiva es un proceso que consiste en la sucesiva superposición de capas micrométricas de material, ya sea en forma de polvo o líquido, hasta obtener el objeto deseado. En este proceso se realiza una deposición controlada de material, aportando exclusivamente donde sea necesario, en lugar de arrancar o conformar la geometría con ayuda de moldes o utillajes.

El proceso más común de fabricación mediante este sistema comienza con la pieza diseñada en CAD y posteriormente convertida en archivo STL, el cual es capaz de entender la máquina. Tras enviar la pieza a fabricar, la impresora "corta" la pieza en capas de un espesor determinado. De cada una de esas capas obtiene un contorno interior y una exterior, lo que le permitirá fabricar la pieza capa a capa.

La consolidación del material de cada una de las capas se consigue de manera diferente en función de la tecnología empleada. Las tecnologías más exitosas son: el sinterizado por laser, tecnología que mediante un laser sinteriza el material en polvo de la pieza; la fotopolimerización, que realiza un curado del material líquido al ser expuesto a una luz ultravioleta; y por extrusión, en la cual se calienta el material para poder inyectarlo por el cabezal de la impresora. Koç y Özel [8].

Este método de fabricación supone una nueva revolución industrial. Las principales características que lo distinguen de otros procesos convencionales y le aportan grandes ventajas son:

- La capacidad de conseguir fabricar geometrías complejas con gran facilidad.
- La facilidad de adaptación del producto al cliente, creando pieza personalizadas sin aumentar el coste de fabricación.
- La competitividad de fabricación en series cortas de producción, siendo más rentable que una línea de producción.

Gracias a la revolución industrial que supone este proceso de fabricación aditiva, se han realizado numerosos estudios en el campo de la optimización, procurando minimizar los errores geométricos ocasionados por la fabricación por capas.

Como trabajo previo, se ha realizado una documentación sobre la optimización en fabricación aditiva. En este capítulo se va a recoger y comentar la bibliografía empleada como documentación.

Phatak y Pande, 2012 [17] presentan un estudio cuyo objetivo es minimizar un conjunto de objetivos ponderados. Para ello, la única variable independiente que se modifica mediante un algoritmo genético es la orientación en los tres ejes, es decir, tres ángulos. Los objetivos que tiene en cuenta son la altura de fabricación, la rugosidad, el material empleado, la superficie en contacto con material de soporte y el volumen del material de soporte.

Paul y Anand [15] presentan un estudio más específico sobre la minimización del error cilíndrico, es decir, minimizan el error de escalera que aparece en las formas cilíndricas de una pieza. Para ello, varían la orientación de la pieza en un solo eje, obteniendo diversos ángulos de orientación para el cilindro y analizando en cada uno de ellos cual era el error obtenido. De esta manera se puede observar que el mínimo error se obtiene cuando el cilindro está completamente vertical y el máximo error es obtenido cuando el cilindro se coloca de manera horizontal. Para la obtención de las diversas orientaciones emplean un algoritmo basado en los mínimos cuadrados.

Feng, Wei y Yongnian [5], al igual que en el estudio anterior, pretenden minimizar el error de escalera, debido a la fabricación por capas, con la diferencia de que no se centran en características cilíndricas sino en características generales de la pieza. Otra diferencia es que emplean un algoritmo con coordenadas esféricas, mientras que Paul y Anand [15] emplean mínimos cuadrados.

En esta ocasión Paul y Anand [16] tienen como objetivo minimizar el error de planitud, el error cilíndrico y el material de soporte necesario para fabricar una pieza. En este caso, el objetivo vuelve a ser minimizar un conjunto de objetivos ponderados. Una vez más, la variable independiente es la orientación, en este caso en el eje X e Y.

Por último, destacamos el artículo de Thrimurthulu, Pandey y Reddy [22] por dos razones: la primera de ellas es que uno de sus objetivos es minimizar el tiempo de fabricación, un objetivo diferente entre el resto de artículos; y por otro lado, además de emplear como variable independiente la orientación también trabajan con el espesor de capa, ya que la mayoría de las máquinas de fabricación aditiva tiene un rango entre el cual puede variar el espesor de capa. En este artículo se calcula cual es el espesor de capa mayor que se puede emplear en función de las rugosidades permitidas en la pieza.

Además de estos artículos más destacados también se ha leído documentación de optimización tanto de multicriterio basado en Pareto de Li y Zhang [12], optimización del perfil de corte del STL, Haipeng y Tianrui [7] u otros estudios de orientación de piezas, Kumar Mishra y Thirumavalavan [11].

Por otro lado, otra parte de la documentación recogida trataba temas enfocados hacia las restricciones de la tecnología, como pueden ser restricción de un error, Szu-Shen Chen y Feng [21] o el diseño del material de soporte para zonas en voladizo, Calignano [4].

Otro tema sobre el que también se ha realizado una documentación, son las características mecánicas que adquiere una pieza de aleación, en concreto porosidad, Bo Kim et al. [2] y métodos de fractura, Simonelli et al. [19].

Leyendo estos artículos se observa que la variable independiente fundamental es la orientación pero sin tener en cuenta el posicionamiento que pueda tener la pieza dentro del volumen de trabajo de la máquina. Por otro lado los objetivos que se quieren minimizar son errores de planitud, cilíndricos o de rugosidad, es decir, todos ellos dependen únicamente del espesor de capa y de la orientación. Estos artículos proponen métodos para minimizar errores que son propios de la fabricación aditiva pero en ninguno de ellos se tienen en cuenta los errores cometidos por la propia máquina, los errores de precisión.

Es por ello que el proyecto desarrollado tiene en cuenta otros factores. En primer lugar las variables independientes con las que vamos a trabajar son tanto posición (x, y ,z) como orientación (α, β, γ) y además vamos a introducir el modelo cinemático de la máquina de manera que en el algoritmo se tendrá en cuenta el error producido al fabricar la pieza en una determinada posición y orientación. Este error dependerá única y exclusivamente del modelo cinemático de la máquina.

3 ETAPAS DEL PROCESO

En este apartado se realiza una explicación general sobre cada uno de las fases que se realizan en el proceso, desde el diseño de la pieza en CAD hasta la fabricación de la misma.

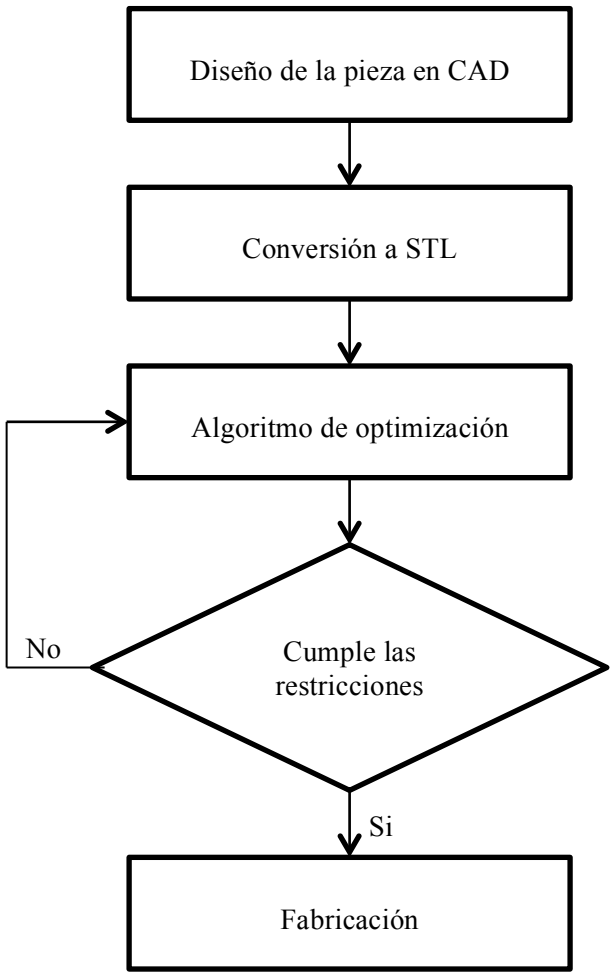


Fig. 1. Esquema de las etapas del proceso

3.1 DISEÑO DE LA PIEZA Y CONVERSIÓN A STL

Como se puede observar en el esquema del proceso, en primer lugar se diseña la pieza en CAD y, posteriormente se convierte a archivo STL. En el siguiente apartado explicaremos que pieza hemos empleado para realizar la experimentación y el porqué de esa selección.

El archivo STL es un formato de archivo informático que define geometrías de objetos 3D. Realiza una aproximación de la superficie del objeto mediante triángulos, creando las superficies del objeto diseñado con tres vértices y la normal del triángulo, de esta manera se define que cara del triángulo es exterior o interior. Para lograr una mejor aproximación de la pieza en archivo STL serán necesarios un mayor número de triángulos, lo cual implicará un mayor tiempo de computación. La selección y reducción del número de triángulos empleados para aproximar la pieza será fundamental a la hora de realizar la experimentación como se verá en capítulos posteriores.

3.2 ALGORITMO DE OPTIMIZACIÓN

3.2.1 ESQUEMA GENERAL

En este apartado se explicará el funcionamiento completo del algoritmo de optimización.

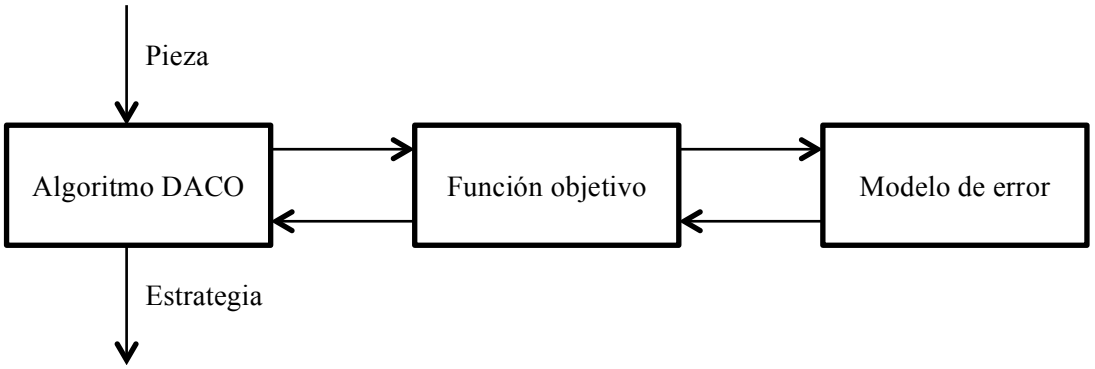


Fig. 2. Esquema del algoritmo de optimización.

En la Fig. 2 se muestra de forma esquemática cual es el funcionamiento del algoritmo de optimización empleado, el cual se ha desarrollado en Matlab, apoyándose en un manual [14].

Al iniciar el algoritmo se debe seleccionar mediante un cuadro de texto cuales son los parámetros que definen el modelo cinemático de la máquina. A su vez, es necesario seleccionar la pieza en archivo STL con la que se va a trabajar, pero debido al elevado tiempo computacional el algoritmo carga directamente los puntos de la pieza de un archivo Excel. Al realizar el paso de STL a Excel los puntos con los que trabaja el algoritmo se reducen a una tercera parte (de 394710 a 131570 puntos). Además de esta reducción ha sido necesario reducir aún más los puntos de la pieza hasta trabajar con un total de 29719 puntos. Tras cargar los puntos de la pieza es necesario cargar los puntos específicos de la geometría con la que se va a trabajar en función del objetivo.

Así pues, una vez cargados los puntos de la pieza y el modelo cinemático, el algoritmo de optimización propiamente dicho empieza a trabajar. En este caso, este algoritmo diseñado se basa en la metaheurístca de colonia de hormigas, ACO. Es importante destacar que las variables

del problema pertenecen al dominio continuo. La descripción de este algoritmo es objeto del siguiente apartado.

Para calcular el objetivo de cada una de las soluciones generadas por el algoritmo, se hace una llamada a Función_Objetivo. En esta función, lo primero que se hace construir una matriz de rototraslación con las 6 variables independientes (3 de posición y 3 de orientación) generadas, y aplicar esta matriz reorientar la nube de puntos de la geometría. La geometría resultante recibe el nombre de nominal.

El siguiente paso en la función es aplicar el modelo de error de la máquina a todos estos puntos reorientados o nominales, obteniendo una segunda geometría llamada real o con error. Como último paso en esta función, se calcula sobre la geometría real el valor del objetivo, es decir, de la tolerancia que se quiera minimizar. Para ello se emplean unas funciones que aproximan mediante mínimos cuadrados los puntos reales a una geometría perfecta. Comparando esta geometría "perfecta" con la geometría real se obtiene el valor de la función objetivo.

Como se puede prever, el tiempo computacional es muy elevado ya que para cada solución generada hay que rehacer dos veces la nube de puntos. Nótese que si sólo se trabaja con una geometría/tolerancia concreta, este conjunto de puntos no es muy grande, pero si en el objetivo se incluye el error máximo o medio de toda la pieza, la nube de puntos es completa. En el caso que nos ocupa, y como se verá más adelante, la pieza patrón utilizada en este trabajo está definida por un total de 29719 puntos, número ya reducido como se ha comentado anteriormente.

Una vez finalizado el algoritmo, con la solución final es posible emplear el STL de la pieza nominal para reconstruirla en la nueva posición y orientación. Este proceso se debe realizar una vez terminado el algoritmo de optimización.

3.2.2 ALGORITMO DACO

El algoritmo de hormigas desarrollado se basa en el desarrollado por Kong y Tian [9], con modificaciones ya establecidas en [1]. Este algoritmo había sido previamente programado en trabajos anteriores, pero una parte de él ha sido incluida en este trabajo.

Como documentación adicional sobre el algoritmo en el dominio continuo se ha empleado: Fernández-Vargas et al.[6], Kong y Tian [10], Liao et al.[13], Salama et al.[18], Socha y Dorigo [20] y Ulusam seçkiner et al.[23].

El software de optimización desarrollado está basado en el algoritmo de colonia de hormigas ACO (Ant Colony Optimization) con variables de decisión en el dominio continuo, llamado DACO. Para iniciar el algoritmo se definen dichas variables independientes a partir de las cuales se obtendrán las diversas estrategias de posicionamiento y orientación de la pieza. En este caso, estas variables independientes son 6: tres variables de posición y tres variables de orientación.

A su vez se han de definir los límites de estas variables, es decir, el rango en el cual están definidas. Para ello, resulta necesario tener en cuenta la máquina de fabricación aditiva seleccionada, ya que el rango de las tres variables de posición queda limitado por el volumen de trabajo de ésta, conocido como bounding box. Así, los límites de estas tres variables son de 0 a 300 mm para los ejes X e Y, y de 0 a 150 mm para el eje Z. En el caso de las tres variables de orientación el rango es de 0 a 180 grados.

Una vez definidas las variables independientes y sus límites, el algoritmo genera un vector de medias y otro de desviaciones estándar para cada una de estas variables. Con estos vectores cada hormiga genera una solución inicial y la mejor solución inicial obtenida será la mejor solución global inicial (Sgb_0).

Previo al inicio de las iteraciones se realiza una actualización de la feromona, es decir teniendo en cuenta la mejor solución global inicial, la media y la desviación estándar de cada variable se recalculan estas dos últimas. Esta actualización de la feromona se realiza después de cada iteración pero de manera distinta dependiendo de si el algoritmo considera que ha convergido en una solución o no. Esta actualización se realiza al final del proceso.

Con estas nuevos vectores de medias y desviación estándar, las hormigas generan nuevas soluciones, las cuales son ordenadas seleccionando la mejor como mejor solución de la iteración (Sitb). A esta solución se le realiza un proceso de mejora local utilizando un muestreo aleatorio con la desviación estándar de la variable correspondiente. Este proceso se realiza con cada una de las variables de manera que si se mejora la mejor solución de la iteración (Sitb) esta nueva solución es denominada como solución de la iteración con mejora local (SitbML1). Este proceso de mejora local se vuelve a realizar para esta nueva solución SitbML1 obteniendo en este caso la solución SitbML2.

Al igual que se ha realizado un doble proceso de mejora local a la solución Sitb se realiza el mismo proceso de mejora local a la mejor solución global inicial (Sgb_0) obteniendo de cada uno de los procesos la solución SgbML1 y SgbML2.

Con SgbML2, SitbML2 y las soluciones generadas en la iteración que aun no han sido empleadas se genera un nuevo conjunto de soluciones del cual se obtendrá la nueva mejor solución global (Sgb).

A partir de este momento se realiza la actualización de la feromona antes de comenzar con la siguiente iteración. En el caso de que el algoritmo no haya convergido la actualización de los nuevos valores de la media y la desviación estándar se obtienen de la siguiente manera:

$$\mu_1|_i = (1-\rho) * \mu_0|_i + \rho * Sgb_{0_i}$$

Ecuación 1. Actualización del vector de medias en caso de no convergencia.

$$\sigma_1|_i = (1-\rho) * \sigma_0|_i + \rho * \left|Sgb_{0_i} - \mu_0|_i\right|$$

Ecuación 2. Actualización del vector de desviaciones estándar en caso de no convergencia.

siendo ρ un parámetro propio del algoritmo llamado factor de evaporación de la feromona.

En el caso de que el algoritmo haya convergido, hay que reinicializar el algoritmo pero sin perder la información. Para ello se almacenan los resultados en una matriz de soluciones reinicializadas (MR). Cada vez que converja el algoritmo se clasifican las soluciones (vrSgbr) y se crea un vector de pesos (vpSgbr) en función de su calidad. Teniendo en cuenta este vector de pesos el cálculo de los nuevos valores de la media y la desviación estándar quedan así:

$$\mu_1|_i = \mu_0|_i + \rho * \sum_{i=1}^{n} vpSgbr_i * vrSgbr_i$$

Ecuación 3. Actualización del vector de medias en caso de convergencia.

$$\sigma_1|_i = \sigma_0|_i + \rho * \sum_{i=1}^{n} vpSgbr_i * |vrSgbr_i - Sgb_i|$$

Ecuación 4. Actualización del vector de desviaciones estándar en caso de convergencia.

Para comprobar si el algoritmo ha convergido o no, se emplea un factor de convergencia, el cual al converger hacia una solución tiende a cero. El factor de convergencia se define como:

$$cf = \frac{\sum_{i=1}^{n} \frac{2 * \sigma_i}{ub - lb}}{n}$$

Ecuación 5. Factor de convergencia.

donde lb es el límite inferior de la variable, ub el límite superior y n el número de variables. Esta última parte de convergencia o no del algoritmo es la que ha sido objeto particular de este trabajo.

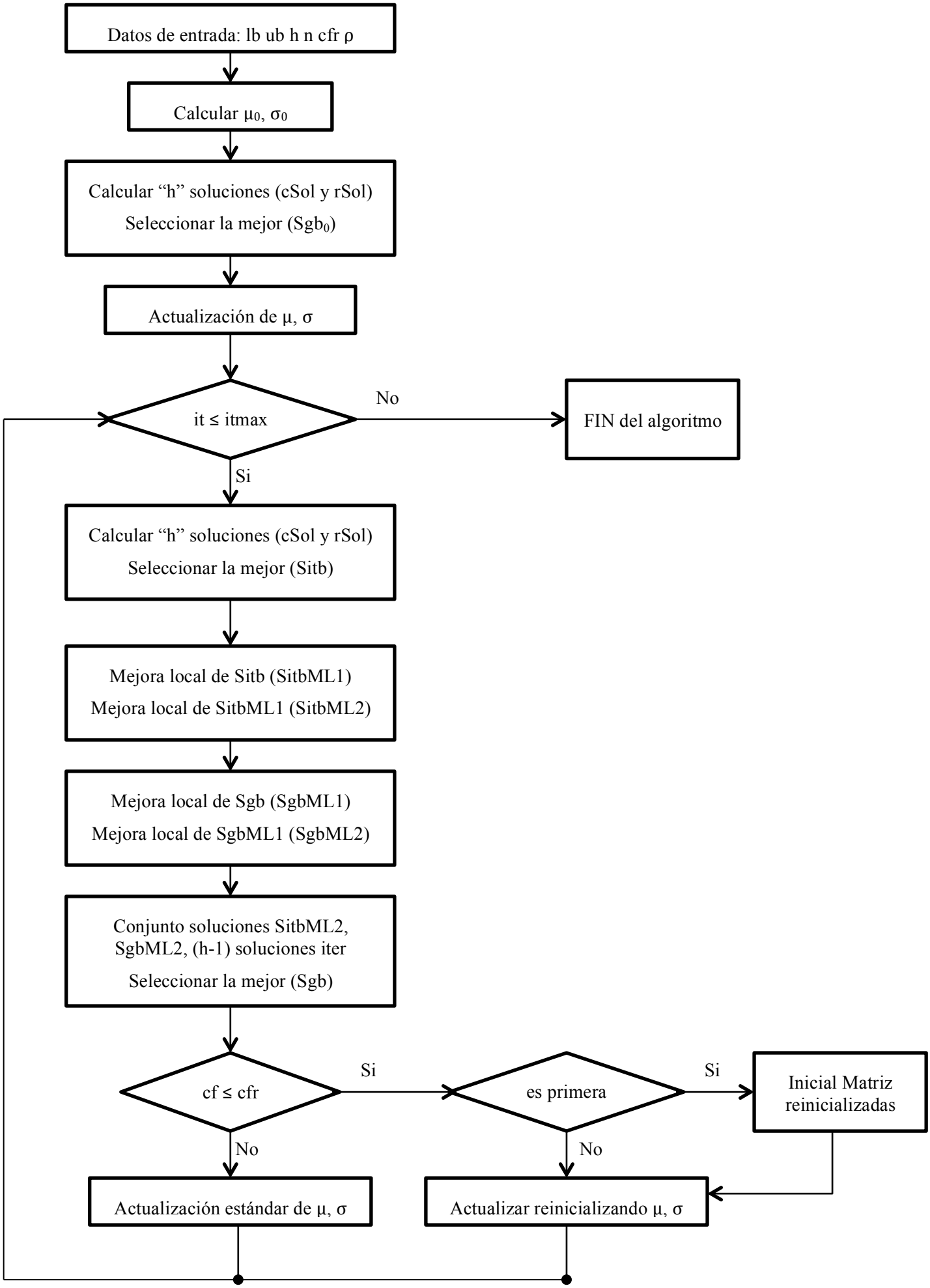


Fig. 3. Esquema DACO

La programación de este algoritmo se encuentra explicada con detalle en el anexo A.

3.2.3 FUNCIONES OBJETIVO

Según el objetivo fijado, la función que va a minimizar el algoritmo variará. Así, se han programado un total de siete funciones objetivo, las cuales son: cilindricidad, concentricidad, planitud, esfericidad, perpendicularidad, error medio y error máximo.

Las siete funciones trabajan de manera muy similar. En primer lugar a partir de las seis variables independientes establecidas por el algoritmo se obtiene la nueva posición y orientación de la pieza. Para obtener esta nueva posición y orientación se emplea una función en Matlab que crea a partir de las variables independientes una matriz de rototraslación.

En una matriz de rototraslación el orden en el que se realizan los giros en cada uno de los ejes no es indiferente. En particular esta función aplica los 3 giros sobre la pieza con el siguiente orden: giro en Z, giro en Y y giro en X. Por otro lado conforme se aplica un giro, los ejes de coordenadas de la pieza van variando pero los tres giros se van a aplicar siempre sobre los ejes de coordenadas fijos del sistema.

Una vez colocada la pieza en la nueva posición se aplica el modelo de error de la máquina. Conocidos los puntos teóricos y los puntos con el error aplicado es cuando se calcula el valor de la función.

Para el cálculo del error medio y el error máximo, se calcula el error que hay entre el punto teórico y el punto con error:

$$error_{punto} = \sqrt{(x_{error} - x_{nom})^2 + (y_{error} - y_{nom})^2 + (z_{error} - z_{nom})^2}$$

Ecuación 6. Error de un punto.

Así, el error medio de un conjunto de puntos de define como:

$$error_{total} = \frac{\sum_{i=1}^{n} error_{punto}}{n}$$

Ecuación 7. Error medio.

El cálculo de las otras cinco tolerancias se realiza a partir de una función de aproximación. En estos casos no se trabaja con todos los puntos de la pieza sino que al iniciar el algoritmo se introducen los puntos de la geometría con la que vamos a trabajar. Por ejemplo, para la tolerancia de cilindricidad se trabaja solo con uno de los cilindros.

Esta selección de la geometría es necesaria debido a la función de aproximación empleada, ya que a esta función de aproximación se introducen los puntos de la geometría con el error aplicado y, siguiendo con el mismo ejemplo, se aproximan a un cilindro, devolviendo un punto del eje, el vector del eje, el radio del cilindro, la distancia de cada punto al cilindro aproximado y la desviación típica de éstos.

Para mejorar esta primera aproximación, teniendo en cuenta la desviación típica, se seleccionan todos los puntos del cilindro cuya distancia el cilindro aproximado estén en un intervalo de $\pm 2\sigma$ y se vuelve a aplicar la función de aproximación a estos puntos.

Una vez conocida la nueva geometría, ya se puede calcular la tolerancia y será este valor el que minimizará el algoritmo.

En el capítulo 6 se realiza una explicación más detallada de cada una de las tolerancias programadas así como de las pruebas y los resultados obtenidos.

Por otro lado en el anexo A se encuentra el código de todas y cada una de las funciones con las explicaciones pertinentes del código.

4 DESCRIPCIÓN DE LOS COMPONENTES IMPLICADOS

En este apartado se va a explicar la pieza con la que se ha trabajado y la máquina de fabricación aditiva que se ha modelizado y empleado en el algoritmo.

4.1 LA PIEZA

La pieza que se ha seleccionado para realizar la experimentación es la que se muestra en Fig. 4.

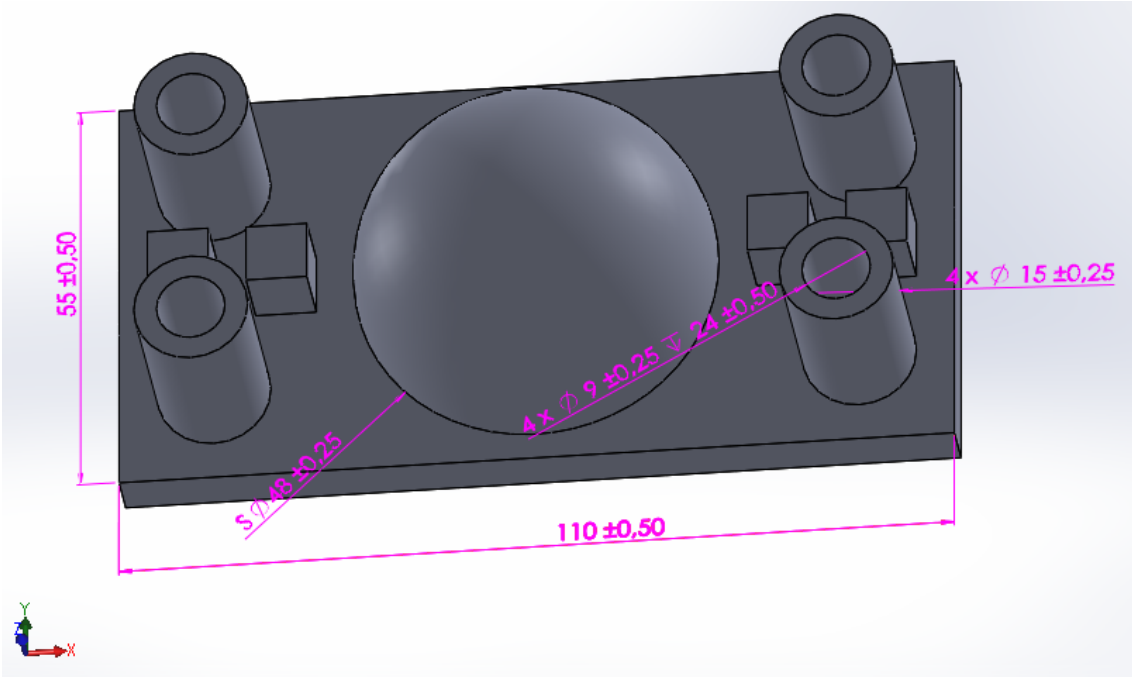


Fig. 4. Pieza Test.

En la Fig. 4 se aprecian las cotas de las geometrías que interesan para el posterior cálculo de las tolerancias. A su vez el sistema de referencia de la pieza queda colocado en la esquina superior izquierda del plano inferior, lo que hace que estando el sistema de referencia de la bandeja y el de la pieza alineados, la pieza queda colocada fuera de la bandeja.

La pieza seleccionada es una de las pieza empleadas para verificación del modelo cinemático de error de la máquina en la tesis doctoras de Cajal Hernando [3]. Al disponer de geometrías habituales en las piezas, la medición de la pieza tras la fabricación permite detectar errores de forma y determinar el correcto funcionamiento del modelo.

Una pieza test de verificación debe cubrir el volumen de trabajo en el que se ha definido el modelo, debe disponer de geometrías de tamaño variado además de tener huecos y protuberancias y tener una gran cantidad de geometrías reales.

La Fig. 4 muestra la pieza de trabajo con sus diferentes geometrías, los cuales van a permitir estudiar numerosas tolerancias.

Las tolerancias que se van a estudiar son las siguientes:

- Cilindricidad, definida en uno de los cilindros exteriores de la pieza, en concreto en señalado en Fig. 4.
- La concentricidad entre dos cilindros, definido por la distancia entre los ejes del cilindro exterior y el cilindro interior señalados.
- La planitud del plano en el cual se apoyan las diversas geometrías de la pieza.
- La esfericidad de la semiesfera.
- La perpendicularidad entre el cilindro exterior y el plano señalado.

4.2 LA MÁQUINA DE FABRICACIÓN ADITIVA

La impresora 3D con la que se ha trabajado es la Objec Eden 350. Esta impresora es la que se encuentra en el Área de Ingeniería de los Procesos de Fabricación de la Universidad de Zaragoza.

Fig. 5. Impresora 3D Objet Eden 350.

Esta impresora esta formada por dos cadenas cinemáticas. Una cadena cinemática que permite el movimiento del cabezal de la impresión en los ejes X e Y y una segunda cadena que permite el desplazamiento en el eje Z, quedando definida como una máquina de tres ejes lineales del tipo ZFXY. Estas dos cadenas cinemáticas se representan en la Fig. 6.

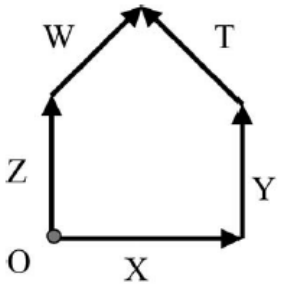


Fig. 6. Diagrama vectorial de las cadenas cinemáticas.

Mediante la manipulación matemática del diagrama vectorial se pueden expresar cada una de las tres coordenadas del vector W en función de los errores de la impresora.

A partir de estas ecuaciones vectoriales se pueden calcular e identificar los errores mediante dos métodos, el cálculo geométrico y la identificación volumétrica.

En este caso la identificación empleada en el modelo de error es la identificación volumétrica. Es un proceso de identificación de parámetros mediante la medición de los errores de la máquina, a partir de un modelo no lineal. Para poder aproximar los errores es necesario emplear una función de regresión. Debido a sus características, los polinomios de Legendre son los empleados para definir cada función de error. Para definir cada función de error han sido necesarios 60 coeficientes, 54 errores (18 por cada eje), 3 componentes offset del cabezal y 3 componentes aleatorios.

El cálculo del modelo cinemático de error esta detallado en la tesis doctoral de Cajal [3], ya que este modelo no ha sido ámbito de este trabajo.

A partir de los resultados obtenidos durante la experimentación del modelo cinemático, se determinó que en el intervalo en el que se define el modelo cinemático de error, el error medio de la pieza de trabajo, Fig. 4, debe estar entre 300 y 400 μm

5 VERIFICACIÓN Y VALIDACIÓN DEL ALGORITMO

En este apartado se van a exponer los diferentes experimentos que se han realizado así como los resultados obtenidos en cada uno de ellos. Las condiciones de experimentación son las mismas en cada uno de ellos, y son establecidas en el análisis preliminar:

- Las 6 variables independientes, 3 de posición y 3 de orientación, tienen los mismos límites inferiores y superiores.
- Los parámetros del algoritmo de optimización DACO, así como el número de iteraciones, hormigas y réplicas de dicho algoritmo.
- La diferencia entre estos experimentos radica en la función objetivo o lo que es lo mismo la tolerancia que se intenta optimizar. En cualquiera de estos casos la meta del algoritmo es intentar obtener como resultado el mínimo posible, el cero, no trabajando en ningún caso contra una meta diferente.

5.1 VALIDACIÓN DEL ALGORITMO

Para validar el correcto funcionamiento del algoritmo se ha empleado una pieza muy sencilla, un listón, con unas dimensiones muy particulares. Al poseer una dimensión mucho mayor que las otras dos es sencillo conocer cuál debe ser la orientación de la pieza al aplicarle un error en uno de los ejes ya que esta dimensión nunca debe estar en la dirección de dicho error. Este listón se ha representado mediante 12 puntos: 8 de ellos en los vértices y los 4 restantes son el punto medio de los lados más largos.

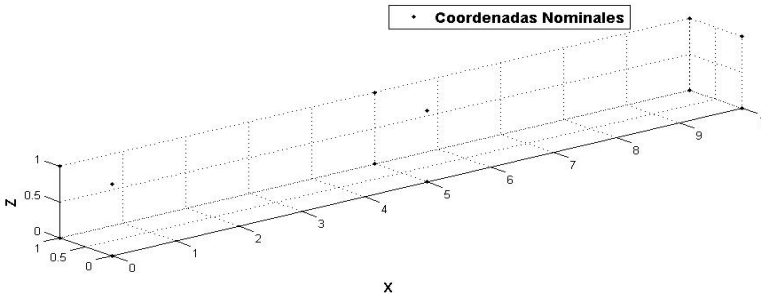


Fig. 7. Orientación inicial del listón.

A su vez el modelo cinemático de error de la impresora ha sido simplificado, poniendo a cero todos los parámetros que lo caracterizan excepto uno o dos para poder controlar fácilmente la estrategia de posicionamiento y orientación.

El objetivo de estas simplificaciones es comprobar que el algoritmo obtiene una estrategia de orientación y posicionamiento que se corresponde con la lógica y que se puede entender fácilmente e incluso comprobar numéricamente. Así, se obtendrá a partir de los puntos que definen el listón teórico o nominal, y aplicando el modelo de error cinemático de la máquina, una nueva nube de puntos que definen el listón real.

Durante la verificación del algoritmo no se ha trabajado con ninguna tolerancia sino que el objetivo es minimizar el error medio del listón. Se define el error medio como la suma para cada punto de la raíz cuadrada de la suma de los cuadrados de las diferencias entre las coordenadas de cada punto nominal y su punto real correspondiente y todo ello dividido por el número de puntos.

5.1.1 ERROR DIMENSIONAL EJE X

En la primera prueba realizada se ha simplificado el modelo de la máquina para que solo tenga error dimensional en el eje X. De esta manera y con el listón como pieza de verificación, el algoritmo debería proporcionar como mejor estrategia la pieza situada en el plano Y-Z. En la Fig. 7 se puede ver el resultado obtenido.

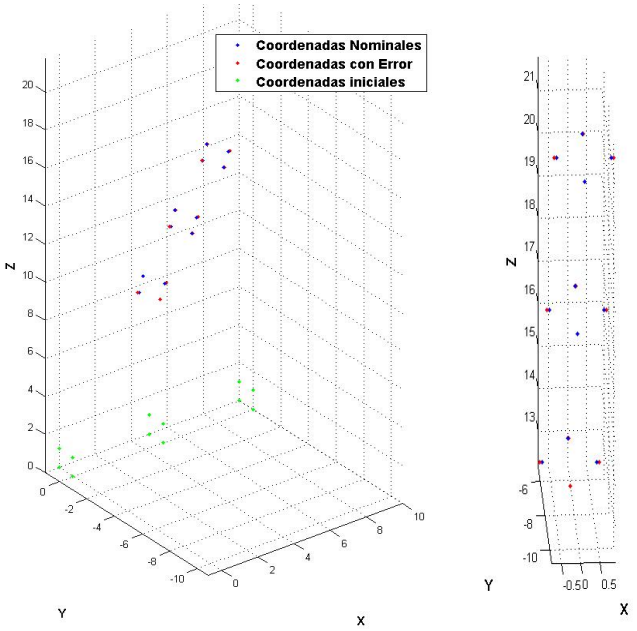


Fig. 8. Estrategia con error dimensional en X.

Pos.X (mm)	Pos.Y (mm)	Pos.Z (mm)	Or.X (º)	Or.Y (º)	Or.Z (º)	Error (mm)
0.0196	-5.0667	12.8279	117.1104	44.1421	89.2073	0.0376

Tabla 1. Resultados con error dimensional en el eje X.

Respecto a la posición, la posición en el plano Y-Z no afecta al error, ya que para este caso, la máquina sólo tiene error en el eje X. Es por ello, que la posición en X es muy cercana a 0 ya que este error aumenta proporcionalmente con la distancia al origen de coordenadas en el eje X. Este valor, Pos.X, no es 0 ya que ha habido un giro de casi 45º en el eje Y.

Según los resultados de las variables de orientación, el listón acaba colocado en el plazo Y-Z (Or.Z=89.2073) y con un giro de casi 45º en el eje Y, lo cual minimiza el error en la pieza ya que solo los 6 puntos que no están situados en la coordenada 0 en X tienen error. La distancia de estos puntos al origen de coordenadas en el eje X en esta posición es menor (0,707 mm) que si no estuviera rotada 45º. En tal caso la distancia sería de 1 mm lo cual implica un mayor error.

Por último analizando el error se puede confirmar que no solo se obtiene una estrategia adecuada sino que los valores medios de desviación de cada uno de los puntos están dentro de lo esperado, ya que como se ha indicado en el apartado 4.2 el error medio de una pieza como la que se empleará en la experimentación se encuentra en torno a los 400 μm.

5.1.2 ERROR DIMENSIONAL EJE X E Y

Continuando con la verificación se ha simplificado el modelo de error simulando en este caso un error de la máquina solo en el eje X e Y. En este caso la mejor estrategia proporcionada debería ser con la pieza vertical al plano X-Y. El resultado obtenido se muestra a continuación.

Pos.X (mm)	Pos.Y (mm)	Pos.Z (mm)	Or.X (º)	Or.Y (º)	Or.Z (º)	Error (mm)
-0.5794	0.5202	13.1851	0.1832	90.4696	180.0	0.0708

Tabla 2. Resultados con error dimensional en el eje X e Y.

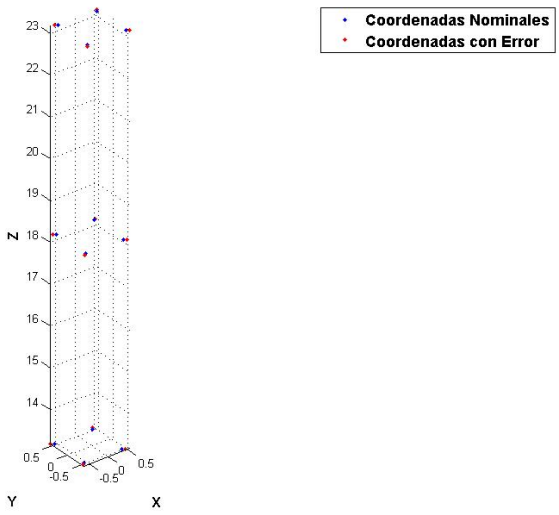


Fig. 9. Estrategia con error dimensional en X e Y.

En este caso, Pos.Z es indiferente y las otras dos son muy significativas. Como se puede ver Pos.X y la Pos.Y hacen que el centro de la cara del listón quede situado en el origen del plano X-Y reduciendo de este modo el error de todos los puntos.

Con respecto a la orientación, el giro Or.Z y Or.Y hacen que el listón quede perpendicular al plano X-Y y en el eje positivo de Z aunque como se ha comentado la posición en este eje es indiferente. Una vez en esta orientación, es fundamental que el giro Or.X sea lo más próximo a 0º para mantener en vertical el listón.

Se puede observar como ha aumentado el valor del error medio con respecto al caso anterior. Esto es debido a que en esta ocasión todos los puntos tienen error.

Tras verificar el correcto funcionamiento del algoritmo se trabajará siempre con el modelo cinemático de error de la máquina sin simplificación alguna.

5.2 PARÁMETROS DEL ALGORITMO DE OPTIMIZACIÓN

El algoritmo de optimización tiene unos parámetros de entrada que deben ser fijados antes de iniciar la experimentación. Entre estos parámetros están los límites inferiores y superiores de cada variable, el número de hormigas, el número de iteraciones y otros parámetros propios del DACO.

Ya se ha comentado en el apartado 3.2.1 como se ha determinado el número de variables independientes así como los límites que se han fijado. A continuación se explica cómo se han determinado el número de hormigas y el número de iteraciones que va a tener cada réplica. En todos estos casos, el factor de evaporación ρ toma un valor de 0.85 y el coeficiente de reinicialización de 0.05.

Para la determinación de estos valores se han realizado una serie de pruebas variando en primer lugar el número de iteraciones, y una vez fijadas las iteraciones variando el número de hormigas. En esta ocasión la pieza empleada ha sido una esfera. La razón de esta elección en lugar del listón es debida a la simplicidad de esta última. La esfera tiene un mayor número de puntos y requiere mayor trabajo encontrar una solución óptima, lo que aumenta en gran manera el tiempo de computación. Es por ello que la solución es más representativa y permite seleccionar de manera más adecuada estos dos parámetros.

En primer lugar si se fija un bajo número de iteraciones el algoritmo no tiene suficiente tiempo para encontrar la mejor solución pero por el contrario si se aumenta considerablemente no solo no mejora la solución sino que aumenta drásticamente el tiempo computacional.

Hay que tener en cuenta que la experimentación se ha realizado en diferentes ordenadores pero de iguales características, lo cual puede influir en el tiempo computacional. En cada una de las pruebas se programó el algoritmo para realizar tres réplicas. La Tabla 3 recoge el mejor resultado de las tres réplicas.

Número de hormigas	Número de iteraciones	Tiempo computacional	Mejor solución obtenida
10	50	2 h 5 min	3,623
10	100	3 h 5 min	3,57
10	500	Más de 20 h	3,62

Tabla 3. Comparación en función de las iteraciones.

Debido a los resultados obtenidos en estas tres pruebas se han fijado el número de iteraciones en 100 ya que aunque para el tiempo de computación el algoritmo es capaz de encontrar la mejor de las soluciones en 3 réplicas.

Número de hormigas	Número de iteraciones	Tiempo computacional	Mejor solución obtenida
10	100	3 h 5 min	3,57
20	100	4 h 25 min	3,564
50	100	23 h 23 min	3,58

Tabla 4. Comparación en función de las hormigas.

La Tabla 4 recoge la experimentación variando el número de hormigas, en ella se puede apreciar que con 50 hormigas el tiempo computacional es demasiado elevado, pero entre 10 y 20 hormigas el tiempo aumenta pero a su vez mejora la solución. Al tener un mayor número de hormigas el algoritmo calcula más soluciones iniciales por lo que es más probable obtener mejores soluciones. Es por ello que se han fijado para todos los casos 20 hormigas y 100 iteraciones por réplica.

5.3 MODIFICACIONES EN EL ALGORITMO

Este algoritmo inicial ha sufrido una serie de modificación y restricciones hasta convertirse en el final con el que se ha trabajado. La primera restricción que se añadió al algoritmo fue la bounding box ya que una pieza situada fuera del volumen de trabajo de la máquina no puede ser fabricada.

Por ello, se ha programado dos versiones del algoritmo. En la primera de ellas, todos los puntos de la pieza nominal resultante deben encontrarse dentro de la bounding box o volumen de trabajo de la máquina. En la segunda versión, se permiten hasta 8 soluciones no válidas, es decir

que no estén dentro de la bounding box de la máquina. A estas soluciones se les suma un valor para impedir que puedan llegar a ser las mejores soluciones pero con el objetivo de que el algoritmo busque soluciones por otras zonas.

Además se han añadido otras modificaciones:

- Debido al coste computacional necesario para trabajar con el STL de la pieza, el algoritmo ha sido ligeramente modificado para trabajar con nubes de puntos y una vez obtenida la solución poder crear el STL de la pieza.
- Se han limitado las soluciones aceptadas en función de un valor de error.
- Se han penalizado de forma exponencial aquellas soluciones que estén por encima de un error mínimo pero que hayan sido aceptadas.

Revisando las soluciones que se obtienen, se puede dar el caso de que al minimizar según una tolerancia determinada, el valor del error tanto máximo como medio sea inaceptable (en el caso de la tolerancia de planitud de la pieza patrón, el error máximo alcanzaba un valor de 20mm).

Por ello, ha sido necesario establecer límites de para este error y penalizar aquellas soluciones que lo superen aunque el valor de la función objetivo sea muy bueno. Así, se han calculado dos límites, uno inferior y otro superior. El límite inferior es el mínimo error máximo obtenido por el algoritmo cuando éste es la función objetivo.

Respecto al límite superior, el valor será diferente según el objetivo de optimización. Para calcularlo se minimiza según la tolerancia en cuestión, y con la posición y orientación de la pieza para obtener dicho valor, se calcula el error máximo con estas variables. Este es el límite superior para esta tolerancia.

La Tabla 5 recoge los valores de los límites fijados.

Tolerancia	Valor mínimo de la tolerancia	Lim_INF de Emax (mm)	Lim_SUP de Emax (mm)
Cilindricidad	37,4 µm	0,543	1,543
Planitud	128,3 µm	0,543	20,6718
Esfericidad	284,2 µm	0,543	0,698
Concentricidad	0,022 µm	0,543	7,6692
Perpendicularidad	0,69º	0,543	4,055

Tabla 5. Límites fijados en el algoritmo.

El objetivo buscado aplicando estos dos límites al algoritmo es lograr que el algoritmo tenga en cuenta dos factores, minimizar la tolerancia pero sin perder de vista el error máximo que tiene la pieza ya que si solo minimizamos la tolerancia el error máximo de la pieza tenderá hacia el límite superior y en la mayoría de los casos ese límite superior está muy lejos del mínimo error máximo alcanzado por el algoritmo (0,543 mm).

Así pues, a la tolerancia calculada se le aplica una penalización proporcional al error máximo que tenga la pieza con dicha estrategia de posicionamiento y orientación. Esta penalización es exponencial de manera que a errores más próximos al límite inferior la penalización sea menor que para errores más próximos al límite superior.

$$tolerancia = tolerancia * K^{\frac{error - \lim\ inferior}{\lim superior - \lim\ inferior}}$$

Ecuación 8. Aplicación de la penalización a la tolerancia.

En este caso y tras realizar numerosas pruebas, la base K toma una valor de 1,75, ya que de esta manera se evita que en la solución se le dé una prioridad excesiva al error máximo de la pieza. No se debe olvidar que el objetivo sigue siendo minimizar la tolerancia con la que estemos trabajando.

6 EXPERIMENTACIÓN Y RESULTADOS

A continuación se expondrán los resultados obtenidos con cada una de las tolerancias. Para mostrar mejor los resultados se van a adjuntar dos imágenes con cada uno de ellos. La primera de ellas se ha obtenido directamente de Matlab y muestra la pieza nominal y la pieza con error mediante una nube de puntos. La segunda imagen es un gráfico con escala cromática en el cual se aprecia directamente en la pieza donde se encuentran los errores. En algunos casos, los valores de error máximo obtenidos por el algoritmo no coinciden con los mostrados en el gráfico cromático ya que para realizar la experimentación se ha realizado una reducción de los puntos de la pieza ya que el tiempo computacional se disparaba.

6.1 TOLERANCIA DE CILINDRICIDAD

En esta tolerancia se trabaja con uno de los 4 cilindros exteriores que tiene la pieza. Como se puede ver en el código adjunto en el anexo A, una vez calculados los puntos con el error de la máquina se introducen en una función de aproximación a un cilindro. Calculada una primera aproximación se vuelven a aproximar aquellos puntos que estén en un intervalo de ±2σ, siendo esta segunda aproximación la que se toma como válida.

Esta función devuelve el valor del radio del cilindro, llamado radio de aproximación, así como la distancia de cada punto a la superficie cilindro aproximado. Con estos valores y el radio téorico del cilindro (que en este caso vale 7,5 mm) se puede calcular la distancia de cada punto a la superficie teórica del cilindro. La tolerancia de cilindricidad es la mayor de todas estas distancias.

A continuación se van a presentar los resultados de los dos algoritmo programados para el cálculo de las tolerancias. En primer lugar se muestra el resultado obtenido del algoritmo más restrictivo, en el cual todas las soluciones tienen que estar dentro de la bounding box.

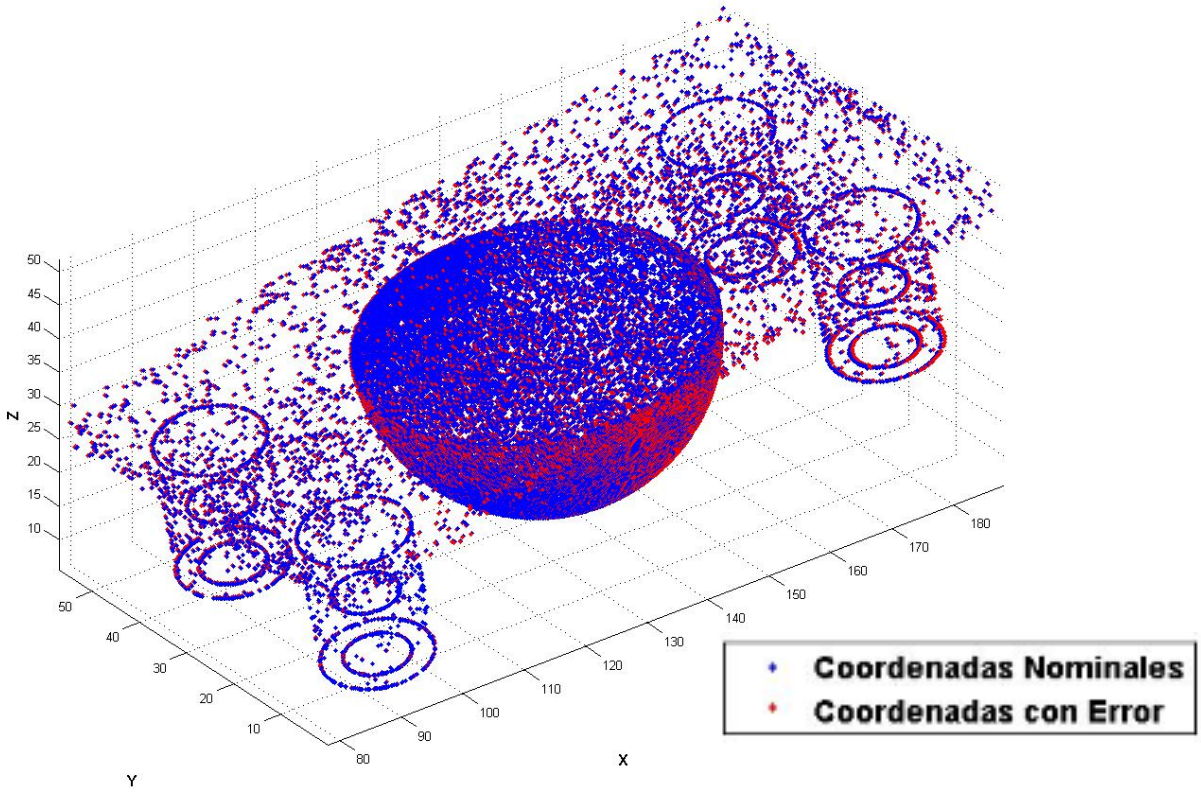


Fig. 10. Estrategia para la tolerancia de cilindricidad aceptando solo soluciones factibles.

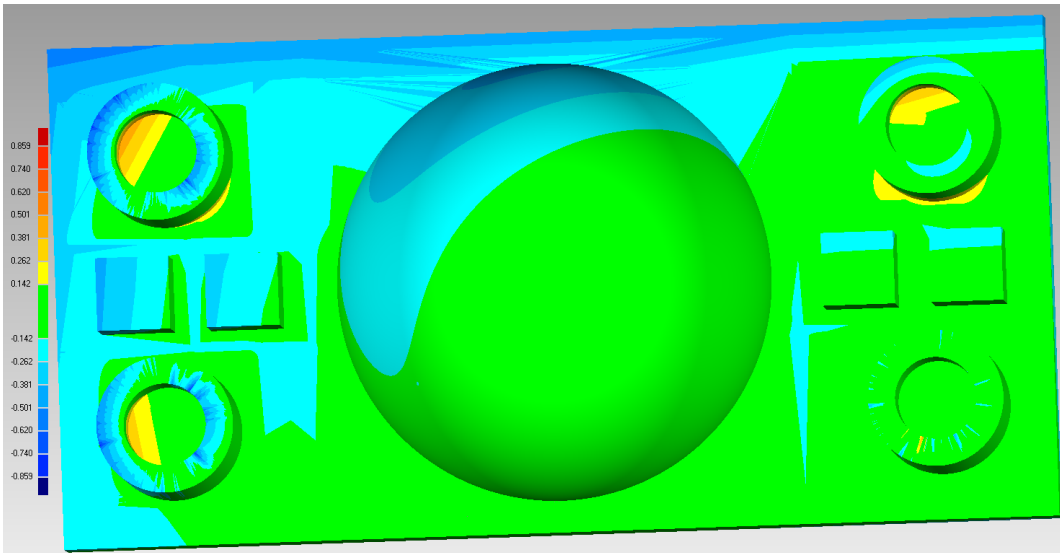


Fig. 11. Error entre la pieza real y la nominal.

Pos. (mm)			Objetivo	Tolerancia Cilindricidad (mm)	Desviación típica	Radio (mm)	Error medio (mm)	Error máximo (mm)	Tiempo computacional
	X	185							
	Y	56,64							
	Z	51,7							
Or.	X	180	0,0428	±0,0408	0,018	7,517	0,244	0,63	85 h. 3 min.
	Y	10,8							
	Z	179,17							

Tabla 6. Resultados de la estrategia con todas las soluciones factibles para la tolerancia de cilindricidad.

Respecto a las imágenes, se puede ver que no hay una gran diferencia entre la pieza nominal y la pieza real. Continuando con los valores numéricos, el radio real en esta posición es 17 μm más grande que el teórico y la tolerancia una vez aplicada la penalización es de ±42,8 μm. Como se puede ver el error máximo está muy próximo al límite inferior (0,543 mm) por lo que se puede concluir que la tolerancia sin penalización será similar a la obtenida. En este caso el valor de dicha tolerancia es de ±40,8 μm.

El resultado obtenido cuando se permiten que 8 soluciones iniciales no estén dentro de la bounding box es el siguiente.

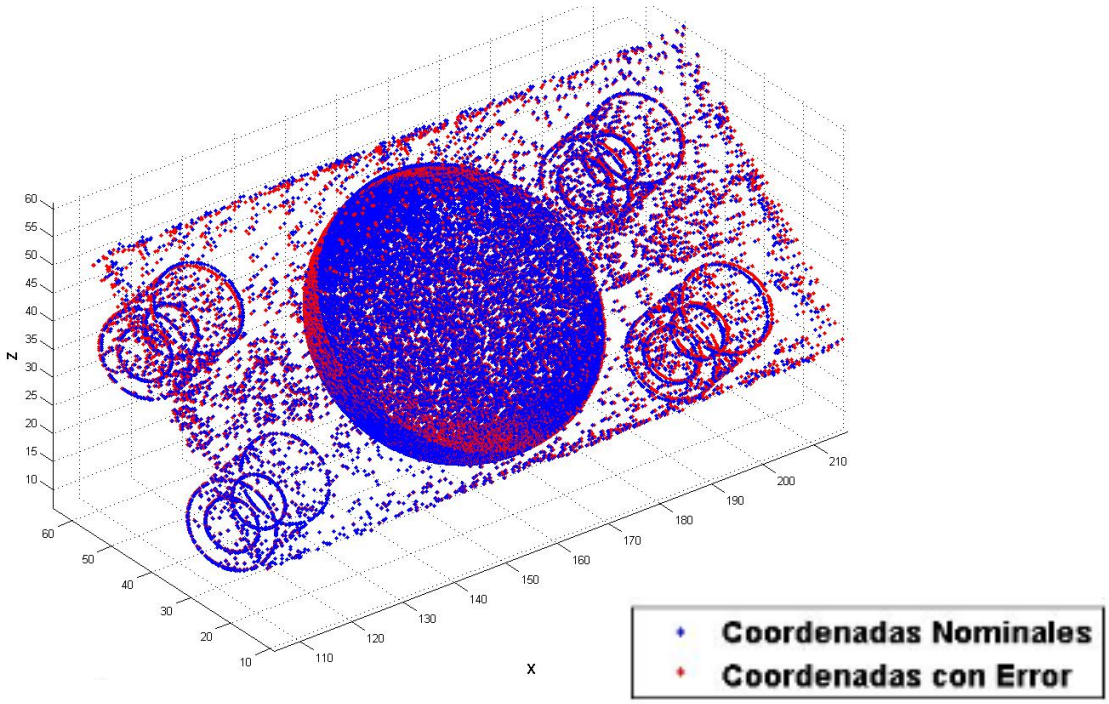


Fig. 12. Estrategia para la tolerancia de cilindricidad aceptando soluciones no factibles.

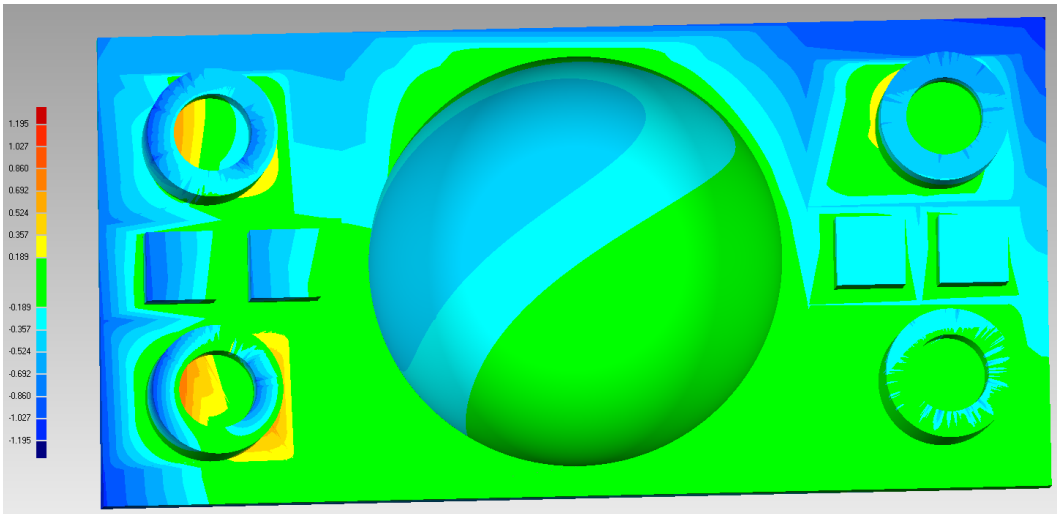


Fig. 13. Error entre la pieza real y la nominal.

Pos. (mm)			Objetivo	Tolerancia Cilindricidad (mm))	Desviación típica	Radio (mm)	Error medio (mm)	Error máximo (mm)	Tiempo computacional
	X	215,41							
	Y	49							
	Z	61,05							
Or.	X	43	0,0407	±0,0245	0,014	7,514	0,459	1,45	88 h. 45 min.
	Y	179,56							
	Z	0							

Tabla 7. Resultados de la estrategia con soluciones no factibles para la tolerancia de cilindricidad.

Observando las imágenes ya se puede apreciar que el error que hay en esta pieza es mayor que el obtenido en el caso anterior, ya que se observa una mayor diferencia en la escala de colores, particularmente en los cilindros con los que no trabaja el algoritmo. Si se aprecian los datos numéricos se ve que el error máximo es de 1,45 mm mucho mayor que antes pero aun así el valor de la tolerancia es menor que el obtenido anteriormente. Esto es debido a que la tolerancia real obtenida es ±24,5 μm. Por lo tanto en esta estrategia se ha obtenido una valor de tolerancia muy reducido a pesar del error máximo de la pieza.

Con estas dos estrategias se pueden observar dos resultados muy diferentes. Por un lado el primer algoritmo ha obtenido una solución más próxima al límite inferior fijado, es decir, un error máximo menor mientras que la otra versión del algoritmo ha obtenido una solución con un valor de tolerancia muy inferior pero sacrificando el error máximo de la pieza. El error medio obtenido en ambas soluciones es aceptable a pesar de que en esta segunda estrategia supera las 400 μm. El hecho de que supere este valor se debe a que como se ha explicado el modelo cinemático está definido en una parte del volumen de trabajo del equipo y si nos salimos de ese volumen los polinomios ya no se aproximan tan bien a la realidad.

Otras conclusiones que se pueden observar en las dos estrategias es que el cilindro con el que se ha calculado la tolerancia en ambos casos se encuentra en la zona más cercana al origen de coordenadas y el error en esa zona es menor que en el resto de la pieza. Ambas estrategias colocan la pieza boca abajo pero la inclinación con respecto al eje Z de la segunda estrategia se

aprecia que provoca un error mayor en el cilindro más próximo al cilindro con el que ha trabajado el algoritmo.

Si se piensa en el tipo de fabricación con la que se está trabajando es importante destacar que la mejor solución obtenida coloque el cilindro de la manera más vertical posible ya que esto reduciría el efecto escalera producido por las diversas capas.

Si se observan los datos obtenidos por el mismo algoritmo sin tener en cuenta el error total de la pieza se podrá apreciar en determinados experimentos la gran importancia de la penalización. A continuación se muestran los resultados de ambos algoritmos sin tener en cuenta el error total.

Pos. (mm)	**X**	201,67	**Tolerancia Cilindricidad (mm)**	**Desviación típica**	**Radio (mm)**	**Error medio (mm)**	**Error máximo (mm)**
	Y	55,04					
	Z	41,8					
Or.	**X**	0	±0,0416	0,0141	7,523	0,29	0,845
	Y	175,28					
	Z	0					

Tabla 8. Resultados de la estrategia con todas las soluciones factibles sin tener en cuenta el error máximo.

Pos. (mm)	**X**	216,39	**Tolerancia Cilindricidad (mm)**	**Desviación típica**	**Radio (mm)**	**Error medio (mm)**	**Error máximo (mm)**
	Y	48,52					
	Z	61,71					
Or.	**X**	43,08	±0,0377	0,0147	7,514	0,501	1,543
	Y	179,03					
	Z	0,5					

Tabla 9. Resultados de la estrategia con soluciones no factibles sin tener en cuenta el error máximo.

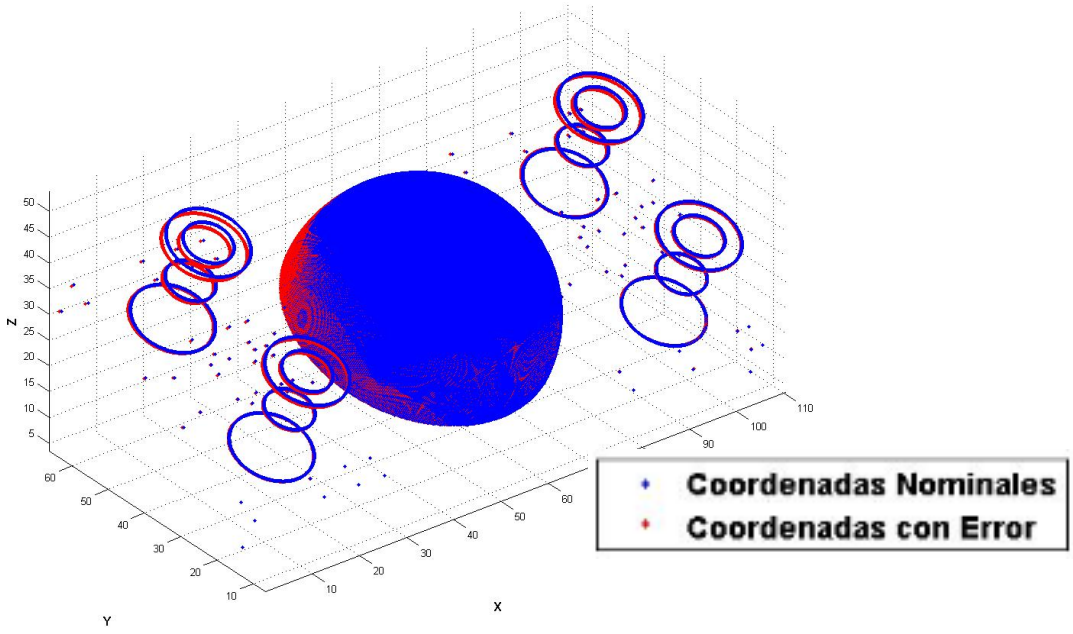


Fig. 14. Estrategia aceptando soluciones no factibles sin tener en cuenta el error máximo.

Comparando estos resultados con los anteriores se aprecia que las tolerancias son muy similares pero en ambos casos el error tanto medio como máximo es mayor. En la tolerancia que se expone ahora se puede apreciar de manera visual la importancia de esta mejora en el algoritmo. En esta tolerancia parece que mejora la solución si se coloca la pieza invertida como se observa en las anteriores estrategias.

6.2 TOLERANCIA DE CONCENTRICIDAD

Para calcular la tolerancia de concentricidad se empleará el mismo cilindro exterior que en la tolerancia de cilindricidad y el cilindro interior de éste. En este caso una vez aplicado el modelo de error a los puntos se calculará por un lado la aproximación de los puntos exteriores a un cilindro y la aproximación de los puntos interiores a otro cilindro. Conocido un punto y el vector director de cada uno de los ejes es posible calcular la tolerancia de concentricidad, distancia entre ambos ejes.

Los resultados obtenidos para esta tolerancia se muestran a continuación.

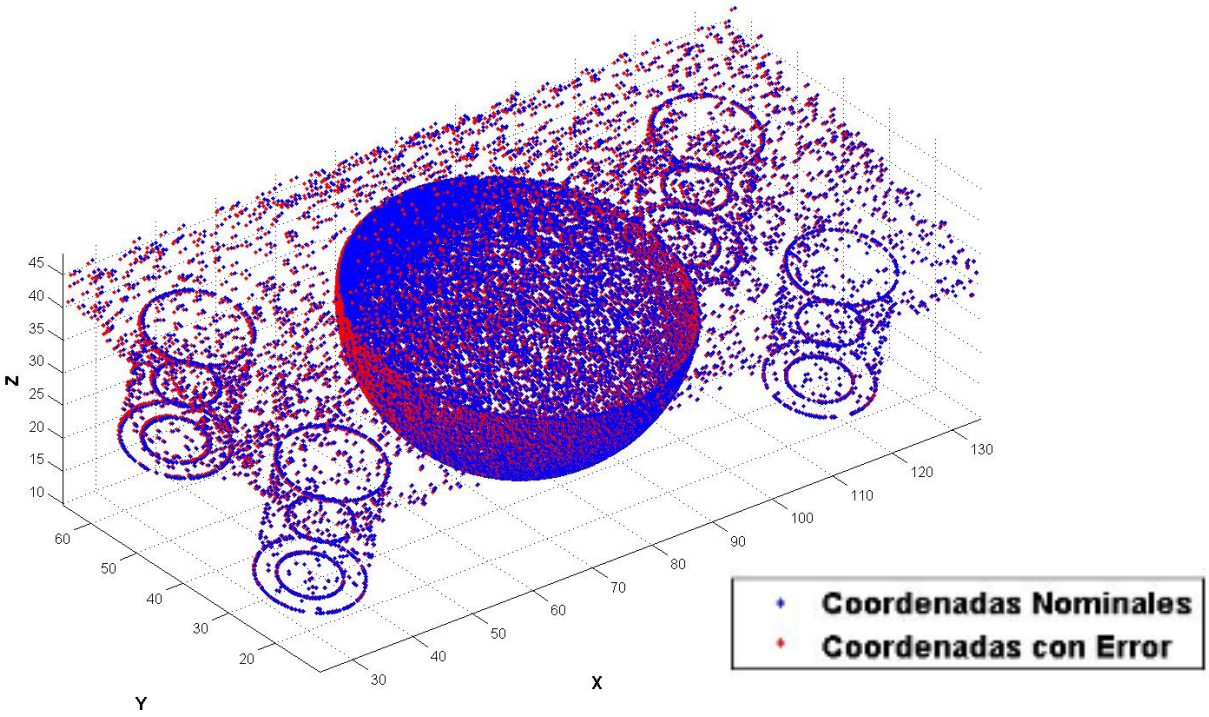


Fig. 15. Estrategia para la tolerancia de concentricidad aceptando solo soluciones factibles.

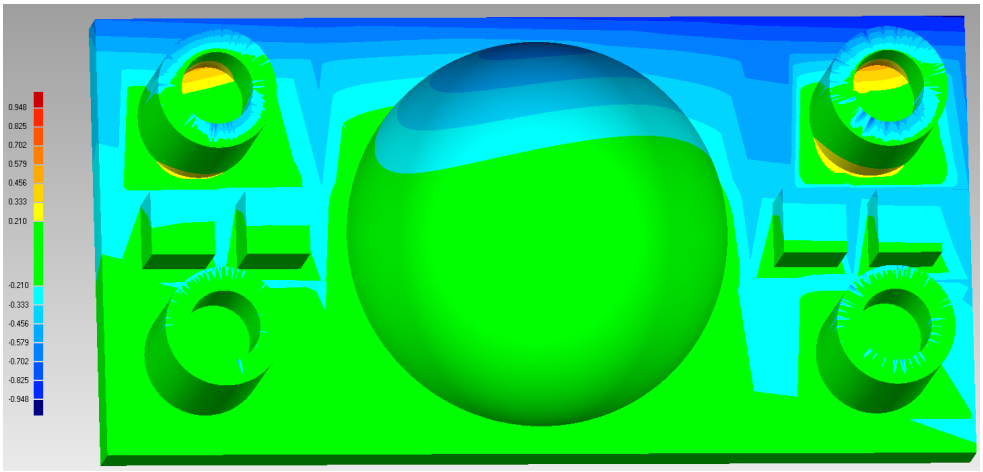


Fig. 16. Error entre la pieza real y la nominal.

Pos. (mm)	**X**	24,72	**Objetivo**	**Tolerancia Concentricidad (mm)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	10,22					
	Z	34,79					
Or.	**X**	14,02	5,45E-06	±5,19E-06	0,383	1,163	274 h. 30 min.
	Y	180					
	Z	180					

Tabla 10. Resultados de la estrategia con todas las soluciones factibles para la tolerancia de concentricidad.

Analizando los resultados numéricos obtenidos y comparándolos con la Tabla 5 en la que aparecen los límites fijados para el algoritmo, así como los valores mínimos que habían sido obtenidos sin tener en cuenta el error, se aprecia que la tolerancia de 0,005 μm obtenida en esta última estrategia es considerablemente menor que la obtenida sin tener en cuenta los límites del error máximo que era de 0,022 μm. Esto es debido a que el algoritmo se ha quedado en un mínimo local.

Si calculamos la misma tolerancia con la otra versión del algoritmo los resultados obtenidos son:

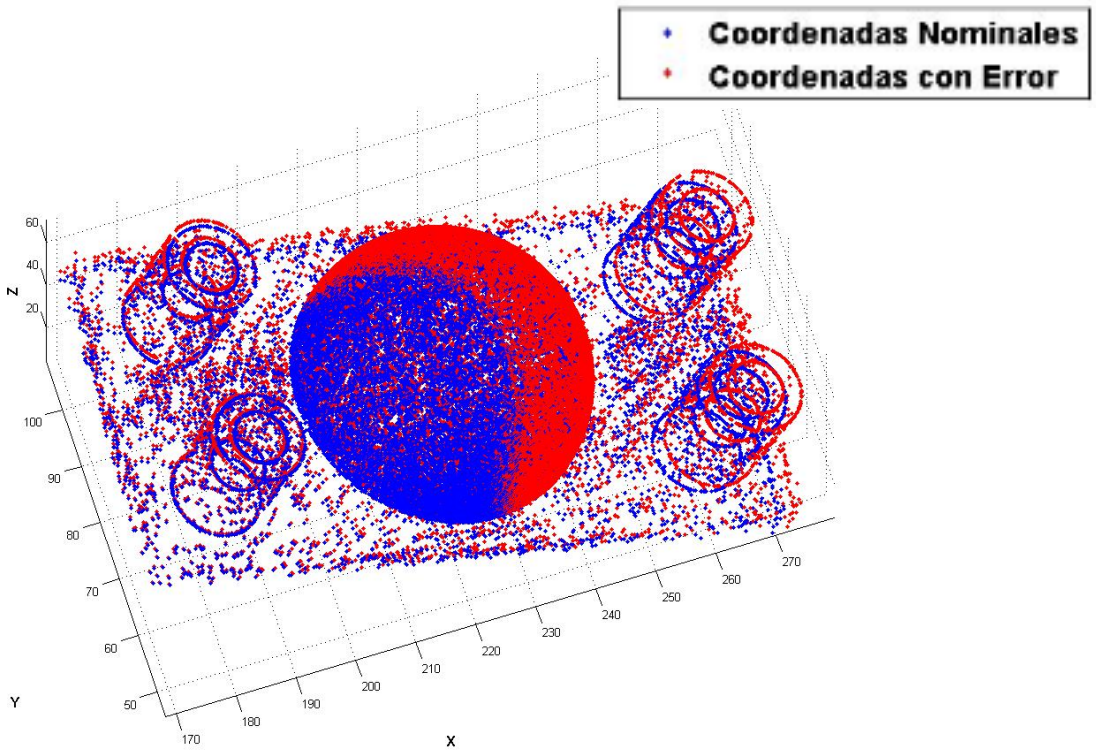


Fig. 17. Estrategia para la tolerancia de concentricidad aceptando soluciones no factibles.

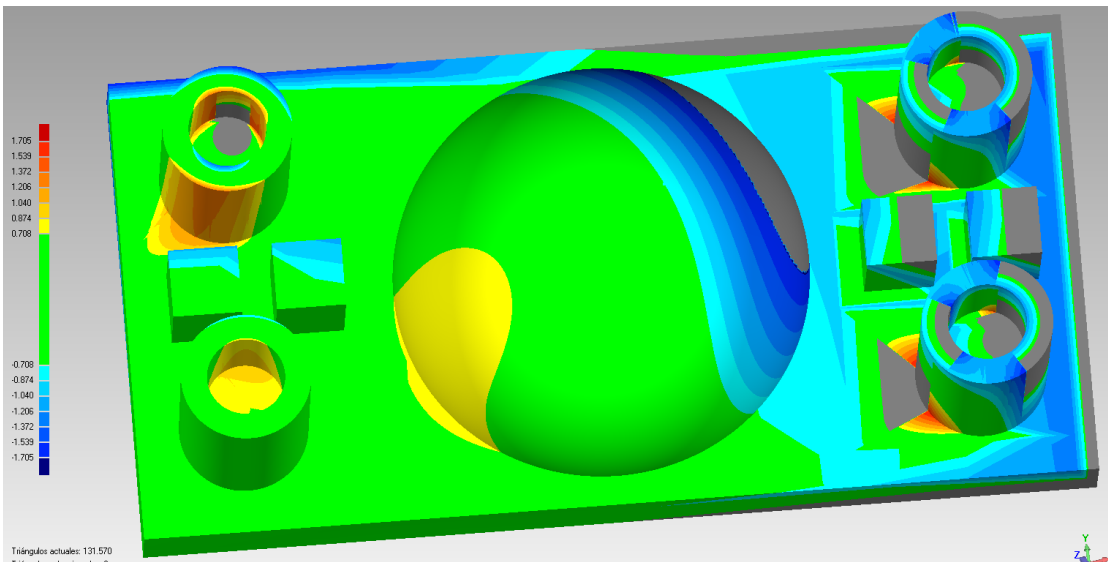


Fig. 18. Error entre la pieza real y la nominal.

Pos. (mm)	**X**	170,02	**Objetivo**	**Tolerancia Concentricidad (mm)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	106,72					
	Z	45,37					
Or.	**X**	178,59	4,662E-05	±3,77E-05	1,542	3,251	60 h. 15 min.
	Y	157,7					
	Z	177,35					

Tabla 11. Resultados de la estrategia con soluciones no factibles para la tolerancia de concentricidad.

Es sencillo llegar a la conclusión de que la estrategia obtenida permitiendo solo soluciones factibles es mejor que esta estrategia ya que tanto el valor de la tolerancia como los valores de los dos errores son menores. Por el contrario el tiempo computacional es significativamente menor que el anterior.

En esta tolerancia llama la atención la enorme diferencia que hay entre ambas estrategias. Para empezar, en la primera de ellas la pieza acaba invertida y mucho más próxima al origen de coordenadas que la segunda que por el contrario está derecha. Por otro lado si se analiza en la posición del cilindro a partir del cual se ha calculado la tolerancia se puede observar que en la primera estrategia ese cilindro se encuentra más alejado del origen que en la segunda estrategia y de todas formas la tolerancia calculada es menor.

Por último también se puede apreciar la diferencia en la variación del error. En la primera estrategia el error en la pieza aumenta con forme la pieza se aleja del eje Y mientras que en la segunda estrategia aumenta más si se desplaza en el eje X. Es cierto que en esta segunda estrategia la pieza está colocada fuera del volumen en el cual se ha definido el modelo cinemático aunque sí dentro del volumen de trabajo, y eso hace que los errores aumenten.

Ya que la primera estrategia es mucho mejor que la segunda vamos a comparar esos resultados con los obtenidos sin tener en cuenta el error de la pieza. Empleando el mismo algoritmo pero sin tener en cuenta ese error, el resultado es:

Pos. (mm)	**X**	130,44	**Tolerancia Concentricidad (mm)**	**Error medio (mm)**	**Error máximo (mm)**
	Y	22,09			
	Z	40,65			
Or.	**X**	151,23	±3,44 E-05	0,921	2,163
	Y	177,74			
	Z	0			

Tabla 12. Resultados de la estrategia con todas las soluciones factibles sin tener en cuenta el error máximo.

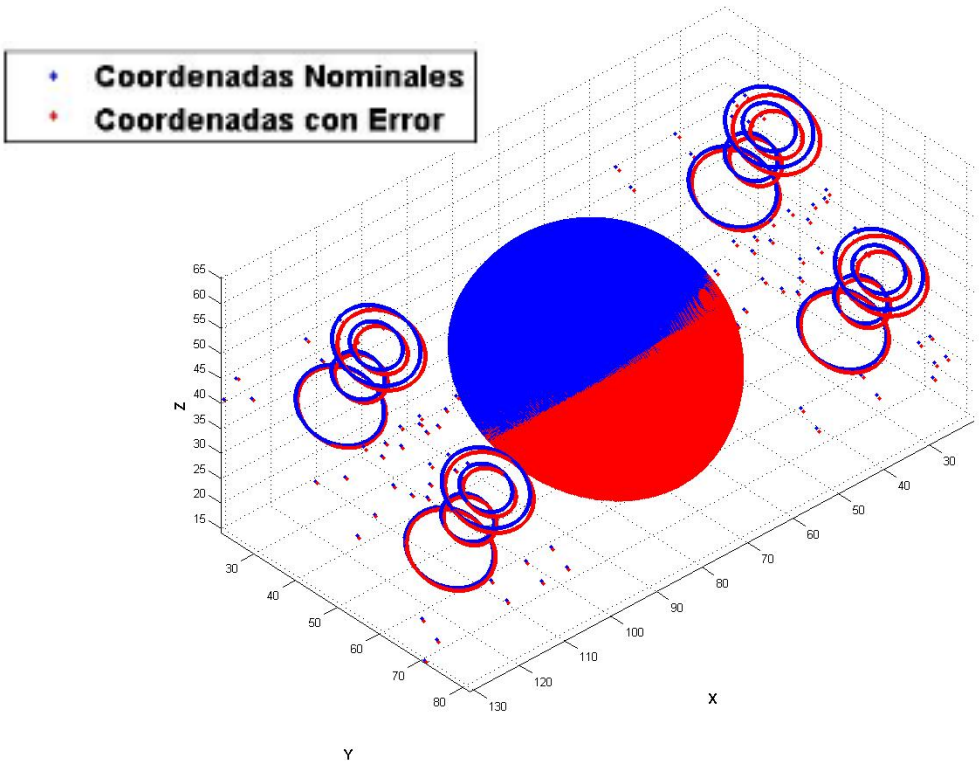


Fig. 19. Estrategia con todas las soluciones factibles sin tener en cuenta el error máximo.

En este caso se puede apreciar mucho mejor como influye considerar o no considerar el error máximo de la pieza ya que estamos hablando de una tolerancia mayor además de un error medio y máximo del doble o mayor. Se puede observar que esta estrategia es muy similar a la obtenida con el segundo algoritmo con la diferencia de que tiene otra inclinación.

De este modo se podría concluir que es más adecuado para esta tolerancia colocar la pieza invertida al igual que en la tolerancia de cilindricidad.

6.3 TOLERANCIA DE ESFERICIDAD

La tolerancia de esfericidad, según la tabla 3, es la que más ajustados tiene los límites del error. Mostrando los resultados se explicará a que se deben esos límites y porque son tan ajustados. Además, se verá que el hecho de que sean tan ajustados también afecta directamente al tiempo computacional. En este apartado, es importante notar que no toda la experimentación se ha realizado en los mismos ordenadores pero este caso en concreto se realizó en uno de los ordenadores más rápidos.

Los resultados obtenidos con ambos algoritmos se muestran a continuación.

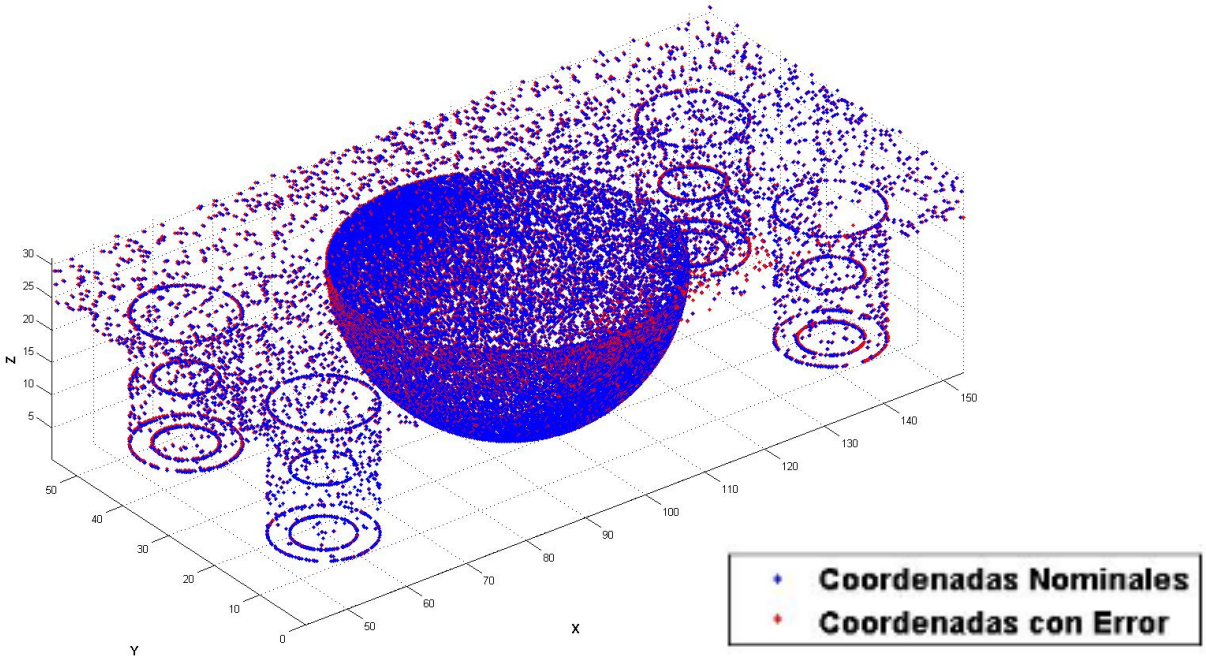


Fig. 20. Estrategia para la tolerancia de esfericidad aceptando solo soluciones factibles.

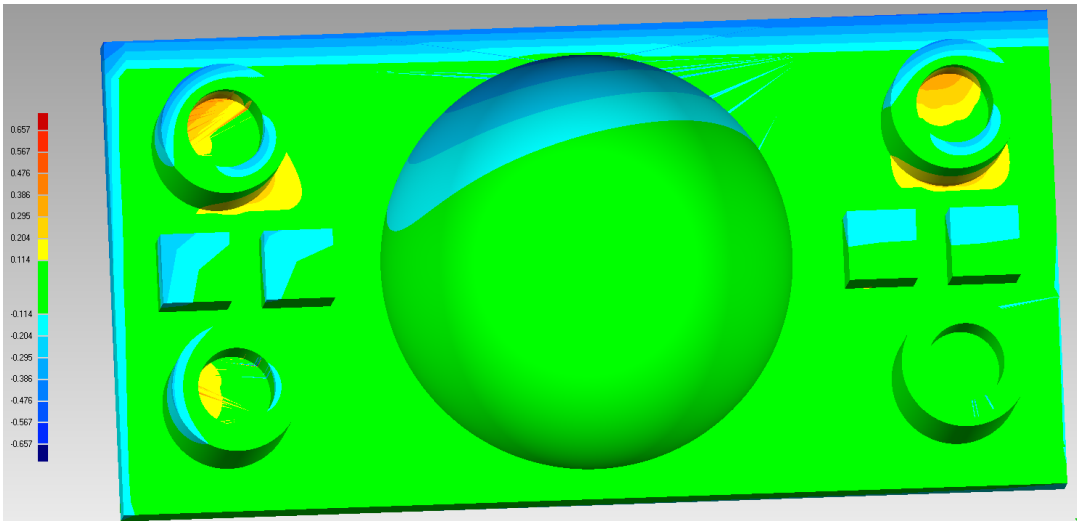


Fig. 21. Error entre la pieza real y la nominal.

Pos. (mm)	**X**	43,12	**Objetivo**	**Tolerancia Esfericidad (mm)**	**Desviación típica**	**Radio (mm)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	0							
	Z	31,05							
Or.	**X**	0	0,2561	±0,2561	0,05	24,182	0,203	0,543	144 h. 20 min.
	Y	180							
	Z	180							

Tabla 13. Resultados de la estrategia con todas las soluciones factibles para la tolerancia de esfericidad.

En este primer resultado se observa al igual que en el caso anterior que el valor de tolerancia obtenido es menor al mínimo calculado previamente (0,284 mm) y que el error máximo coincide con el límite inferior fijado por lo que este valor de tolerancia, por lo que no ha sufrido penalización alguna en este sentido.

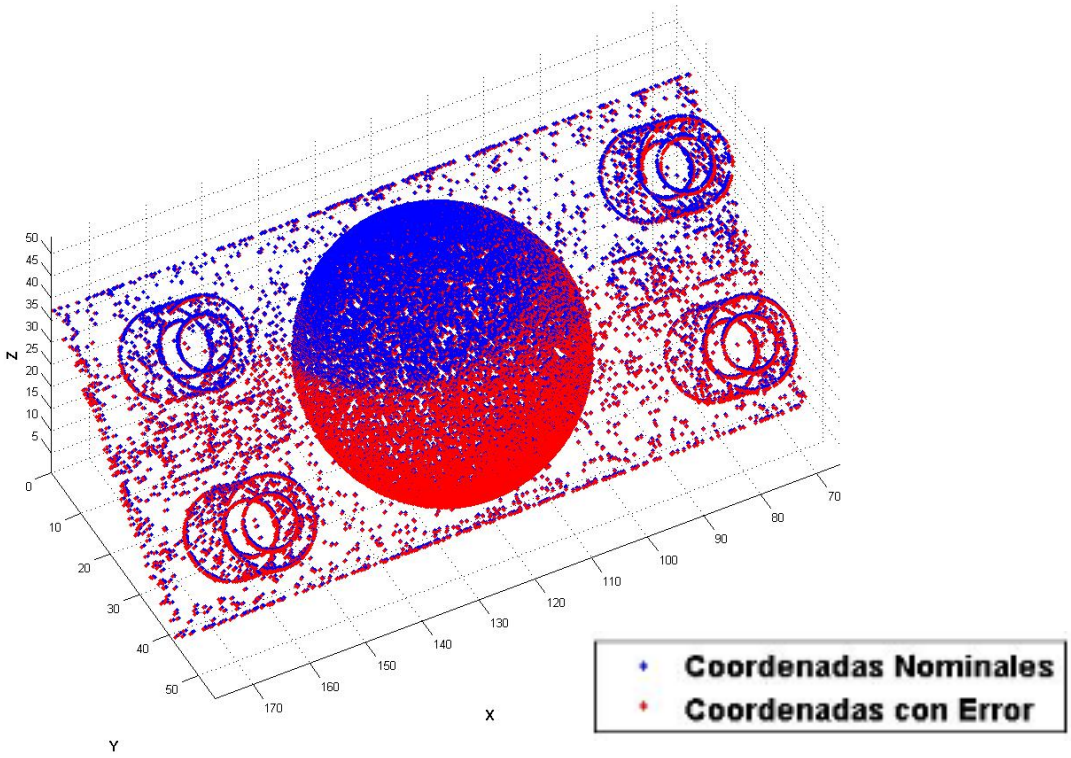


Fig. 22. Estrategia para la tolerancia de esfericidad aceptando soluciones no factibles.

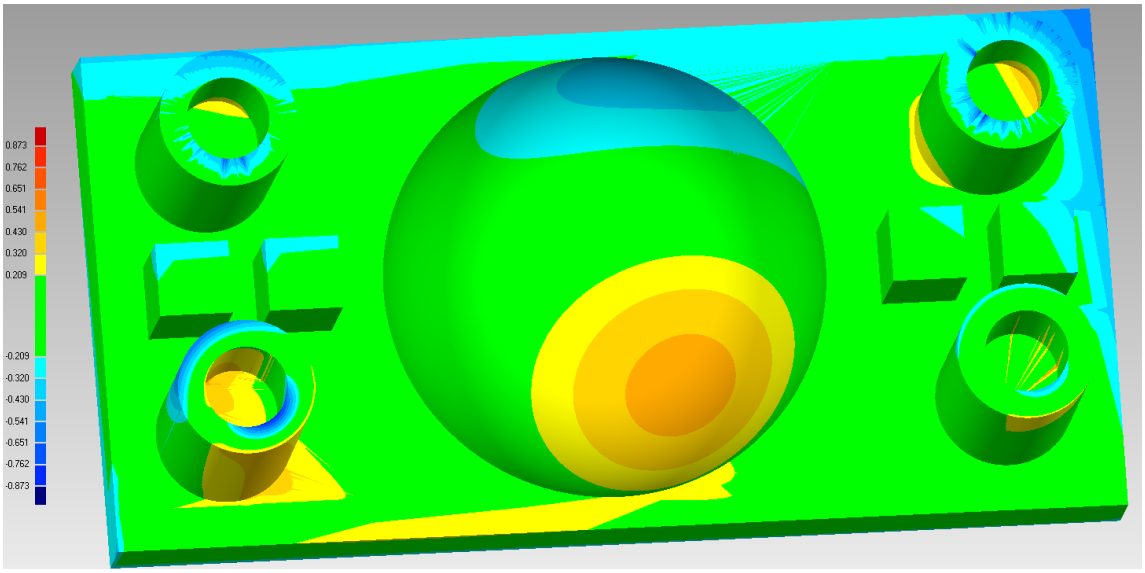


Fig. 23. Error entre la pieza real y la nominal.

Pos. (mm)			Objetivo	Tolerancia Esfericidad (mm)	Desviación típica	Radio (mm)	Error medio (mm)	Error máximo (mm)	Tiempo computacional
	X	176,27							
	Y	0							
	Z	37,13							
Or.	X	138,32	0,2281	±0,2164	0,108	24,041	0,3492	1,024	136 h. 23 min.
	Y	180							
	Z	0							

Tabla 14. Resultados de la estrategia con soluciones no factibles para la tolerancia de esfericidad.

Los resultados de esta estrategia pueden ser un poco confusos ya que el error máximo está por encima del límite superior fijado. En este caso, a pesar de la penalización sufrida por el error, el valor de la tolerancia es inferior que el caso anterior en el cual no había penalización. El valor real de la tolerancia en esta estrategia es de ±0,2164 mm y el error medio no es muy superior al anterior, lo que resulta aceptable.

El hecho de que el error máximo sea mayor del fijado no es un error en la programación. Si se miran los tiempos de computación y se analiza rápidamente el código del algoritmo del anexo A se observa que para obtener las primeras soluciones no solo es necesario que la pieza esté dentro del volumen de trabajo de la máquina sino que también el error máximo de la pieza debe ser inferior al límite superior. Esto hace que sea demasiado costoso encontrar de forma aleatoria 20 soluciones iniciales (porque hay 20 hormigas) que cumplan ambos requisitos. Es por ello que el límite superior fijado ha sido superior al que según los cálculos iniciales correspondía.

A continuación se presentan los resultados del algoritmo sin tener en cuenta el error máximo de la pieza.

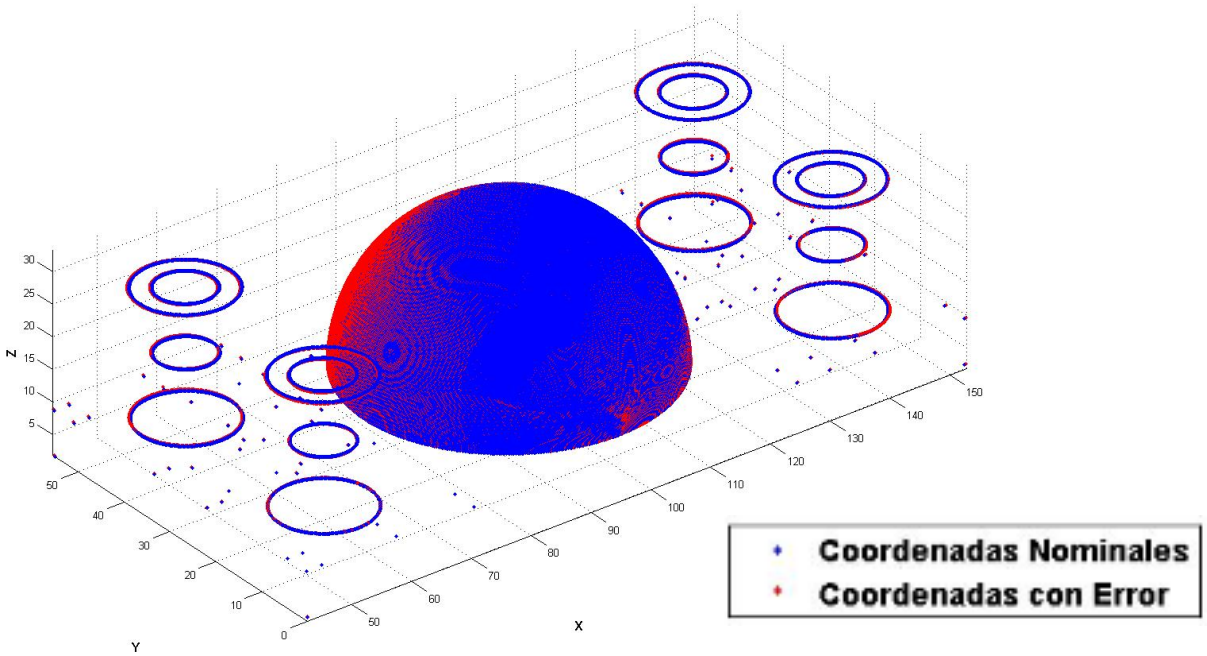


Fig. 24. Estrategia con soluciones no factibles sin tener en cuenta el error máximo.

Pos. (mm)			Tolerancia Esfericidad (mm)	Desviación típica	Radio (mm)	Error medio (mm)	Error máximo (mm)
	X	176,33					
	Y	0					
	Z	0,08					
Or.	**X**	180	±0,2842	0,045	24,209	0,224	0,698
	Y	180					
	Z	0					

Tabla 15. Resultados de la estrategia con soluciones no factibles sin tener en cuenta el error máximo.

Si se analizan los resultados sin tener en cuenta el error máximo de la pieza, éstos son muy similares a los obtenidos en las otras dos estrategias. La conclusión que se obtiene de esto es que en esta tolerancia la relación de puntos empleados con respecto a los puntos totales es muy alta. Al trabajar con más puntos, la pieza con error es mucho más precisa que si solo se emplean puntos de un cilindro. Como inconveniente tiene que el tiempo computacional para trabajar con un mayor número de puntos aumenta ya que tiene que realizar más cálculos.

A pesar de la similitud en los resultados de la tolerancia de cada estrategia, cada una de ellas tiene una orientación completamente distinta. Este hecho demuestra que es más importante el número de puntos con el que trabaja el algoritmo ya que ha obtenido tres buenas estrategias muy diferentes entre ellas.

6.4 TOLERANCIA DE PERPENDICULARIDAD ENTRE CILINDRO Y PLANO

En esta tolerancia al contrario que en la de cilindricidad y en la de esfericidad, en las cuales había una relación entre el radio teórico y el radio real y por lo tanto había otra relación además del error máximo, no existe más relación entre la pieza nominal y la pieza con error que el error máximo. Esto hace que los errores sean mayores ya que el objetivo de la tolerancia es lograr que cilindro y plano sean lo más perpendiculares posibles.

Para el cálculo de esta tolerancia se tiene en cuenta el ángulo que hay entre la normal al plano y el vector director del cilindro.

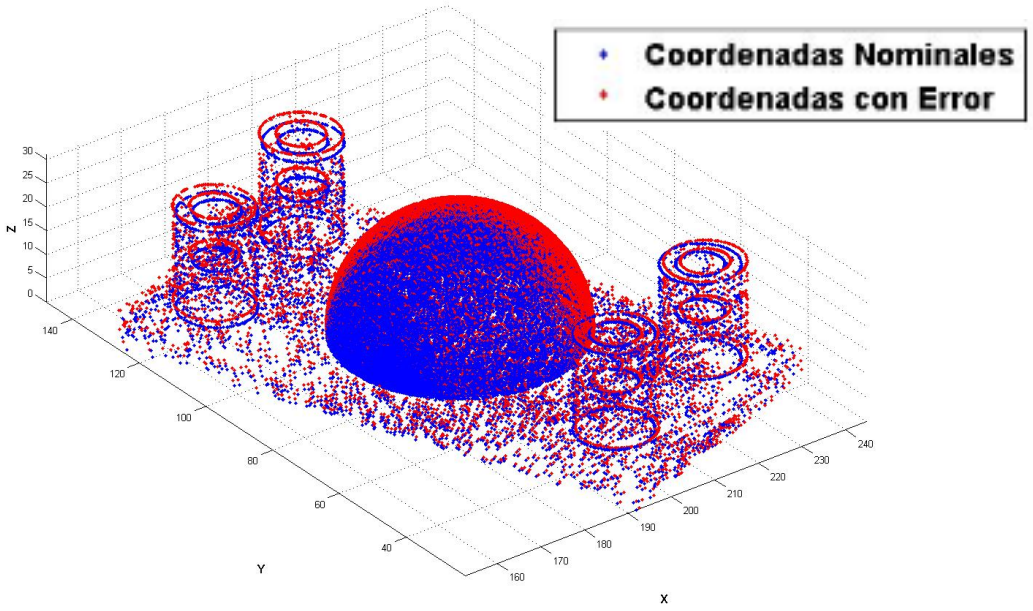


Fig. 25. Estrategia para la tolerancia de perpendicularidad aceptando solo soluciones factibles.

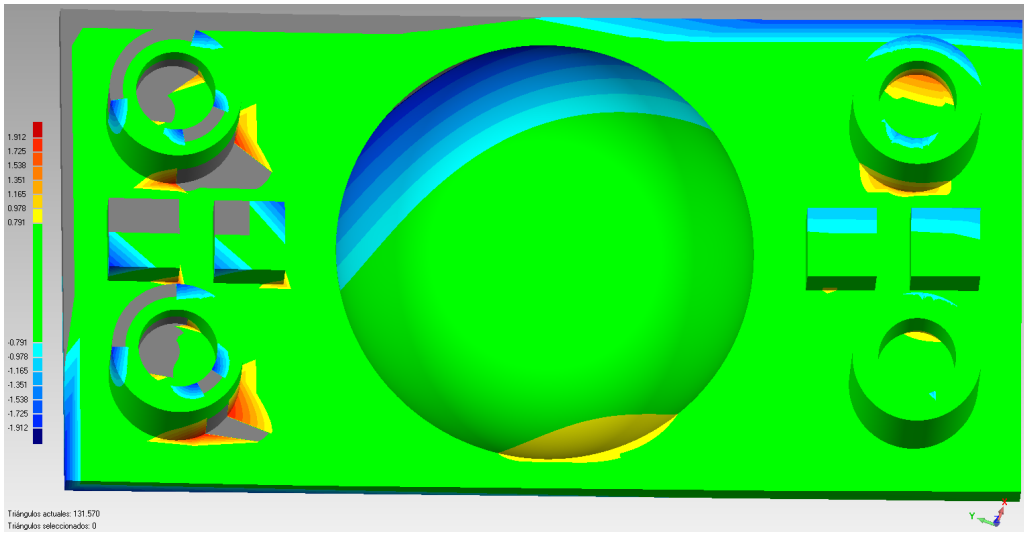


Fig. 26. Error entre la pieza real y la nominal.

Pos. (mm)	**X**	192,02	**Objetivo**	**Tolerancia Perpendicularidad (º)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	22,35					
	Z	0					
Or.	**X**	0	0,05797	±0,0353	1,475	3,653	8 h.
	Y	0					
	Z	110,9					

Tabla 16. Resultados de la estrategia con todas las soluciones factibles para la tolerancia de perpendicularidad.

Con el otro algoritmo los resultados se muestran a continuación.

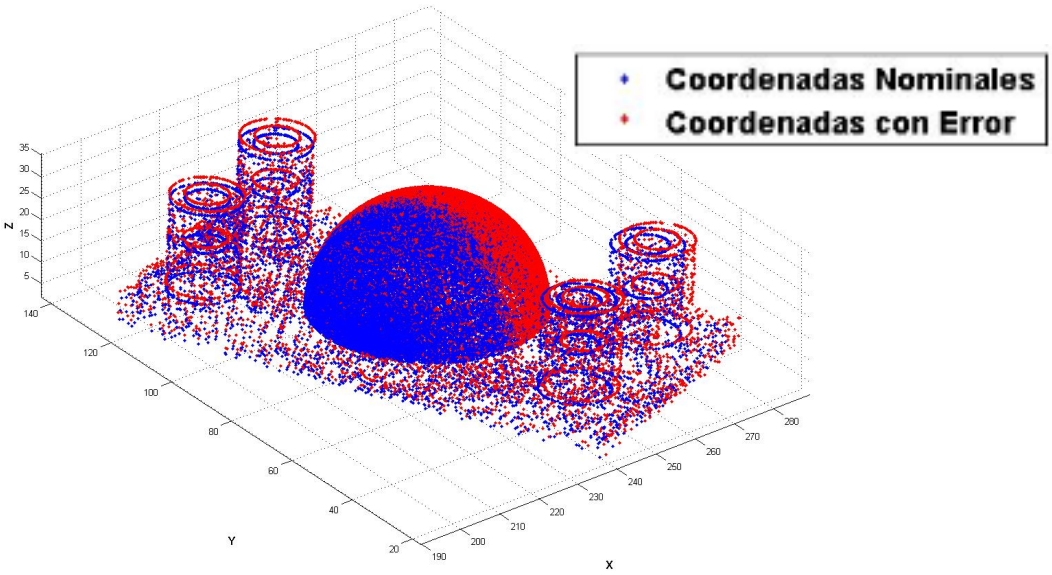


Fig. 27. Estrategia para la tolerancia de perpendicularidad aceptando soluciones no factibles.

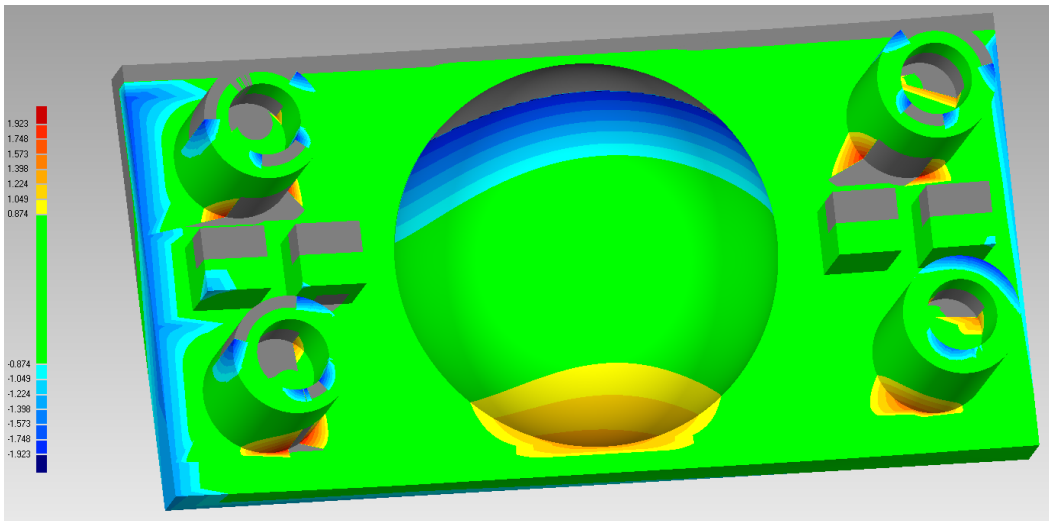


Fig. 28. Error entre la pieza real y la nominal.

Pos. (mm)	**X**	139,24	**Objetivo**	**Tolerancia Perpendicularidad (º)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	140,29					
	Z	2,66					
Or.	**X**	178,54	0,000768	±0,000459	2,033	3,769	139 h.
	Y	175,79					
	Z	114,88					

Tabla 17. Resultados de la estrategia con soluciones no factibles para la tolerancia de perpendicularidad.

Como se puede observar ambas soluciones tienen una tolerancia muy reducida pero ninguna de las dos llega a ser una buena solución ya que el error medio de la pieza es realmente grande.

Ambas estrategias son realmente similares y ninguna de ellas se encuentra dentro del volumen en el que se ha definido el modelo de la máquina, lo cual hace que aumenten los errores. Estos resultados podrían mostrar que en la zona donde está definido el modelo cinemático de error es muy complejo conseguir una perpendicularidad total entre dos geometrías. Para poder demostrar estas conclusiones serían necesarias realizar un mayor número de pruebas.

En este caso no se adjuntan los resultados obtenidos sin tener en cuenta el error máximo de la pieza ya que son muy similares a los mostrados aunque las tolerancias obtenidas con las dos estrategias son mucho menores que la mínima obtenida por el algoritmo sin tener en cuenta el error máximo. Pero como se ha comentado las estrategias que obtiene el algoritmo no siempre son mínimos absolutos, se podría decir que en pocas ocasiones la estrategia final coincidirá con otras obtenidas anteriormente.

6.5 TOLERANCIA DE PLANITUD

Para ver la evolución del algoritmo, en este caso se muestra primero los resultados obtenidos sin tener en cuenta otro tipo de errores y/o relaciones. Los resultados fueron los siguientes:

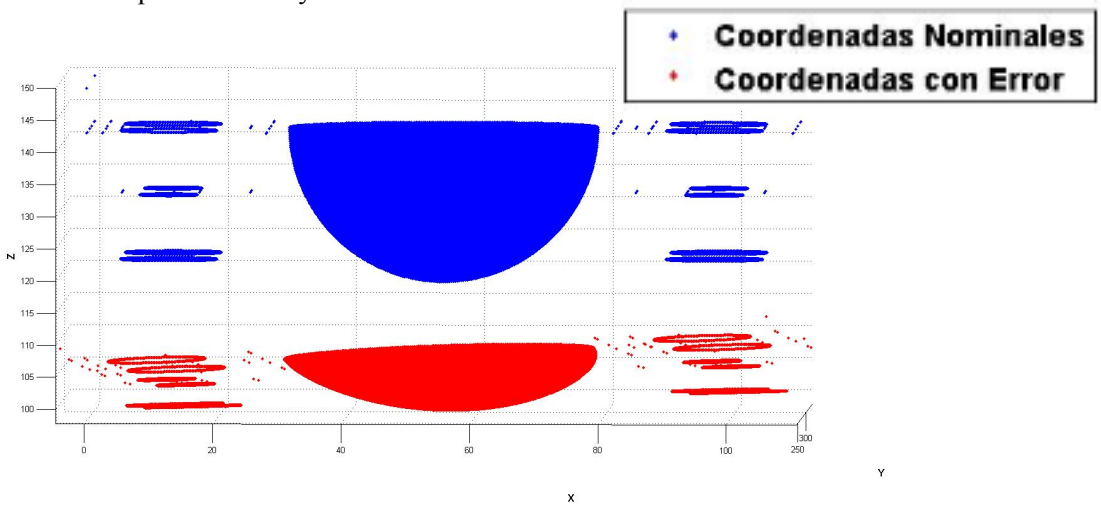


Fig. 29. Estrategia para la tolerancia de planitud sin tener en cuenta la pieza nominal.

Pos. (mm)	**X**	110,2	**Tolerancia Planitud (mm)**
	Y	300	
	Z	150	
Or.	**X**	180	±0,0656
	Y	0	
	Z	179,84	

Tabla 18. Resultados.

Como se puede apreciar en la figura, la diferencia entre pieza nominal y pieza aplicando el error cinemático de la máquina es muy grande. Esto es debido a que no se ha tenido en cuenta el plano nominal y el algoritmo busca una solución en la que los puntos del plano con el error se aproximen lo mejor posible a un plano. Esto corrobora la idea de tener que conjugar la tolerancia de optimización con otros parámetros.

La segunda razón de este resultado tan anómalo se debe al modelo cinemático de error de la máquina. Como se ha explicado en el capítulo 4, el modelo ha sido ajustado en un volumen más pequeño que el volumen total de la máquina por lo que la aproximación de los polinomios es buena en dicho volumen. Fuera de ese volumen, los polinomios ya no hacen una buena aproximación y el resultado de esos polinomios fuera del rango es el que se observa. Aunque no se aprecia con claridad en Fig. 29, la pieza se posiciona entre las coordenadas 250-300 mm en el eje Y, intervalo que está fuera del volumen para el que se ha realizado la aproximación con polinomios de Legendre.

Si se tienen en cuenta los puntos nominales del plano pero sin añadir el error máximo, los cambios son claros como se puede ver en Fig. 30.

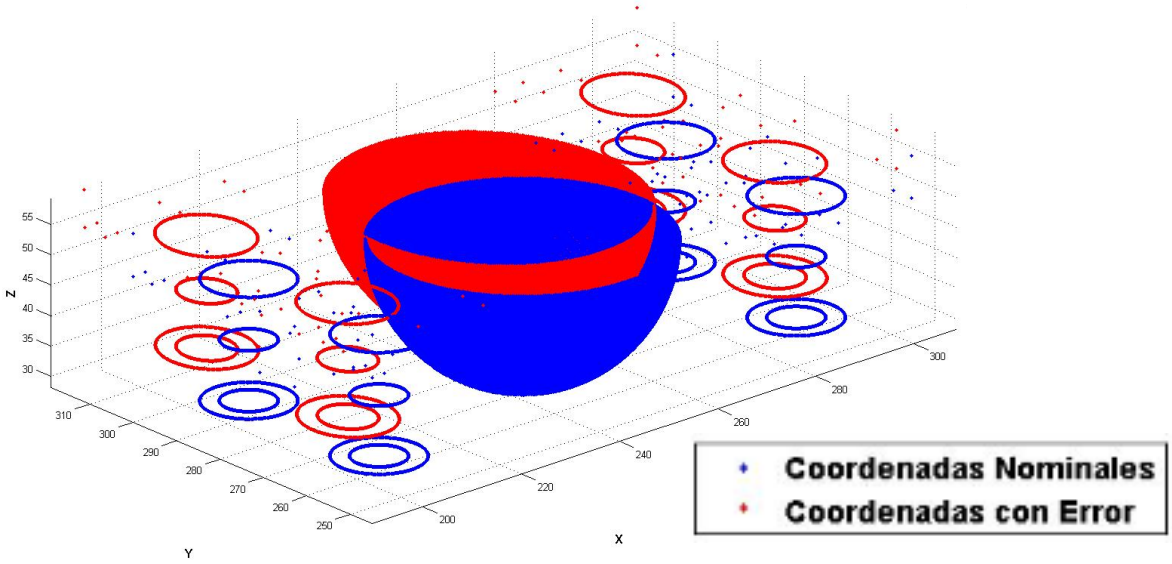


Fig. 30. Estrategia con soluciones no factibles sin tener en cuenta el error máximo.

Pos. (mm)	**X**	299,6	**Tolerancia Planitud (mm)**	**Error medio (mm)**	**Error máximo (mm)**
	Y	300			
	Z	59,21			
Or.	**X**	180	±0,1283	16,1293	20,6718
	Y	0			
	Z	180			

Tabla 19. Resultados de la estrategia con soluciones factibles sin tener en cuenta el error máximo.

A pesar de la clara mejoría en el error que no en el valor de la tolerancia de planitud, esta solución sigue siendo inviable ya que tiene unos errores demasiado elevados. Se puede observar otra vez que la pieza está colocada en el límite del volumen de trabajo de la máquina (posición en X y en Y). En esa posición el resultado obtenido no sería fiable debido al ajuste del modelo de error.

Por último se van a mostrar los resultados finales teniendo en cuenta el error máximo de la pieza. En primer lugar se mostrarán los resultados del algoritmo que solo acepta soluciones factibles y después los resultados del otro algoritmo.

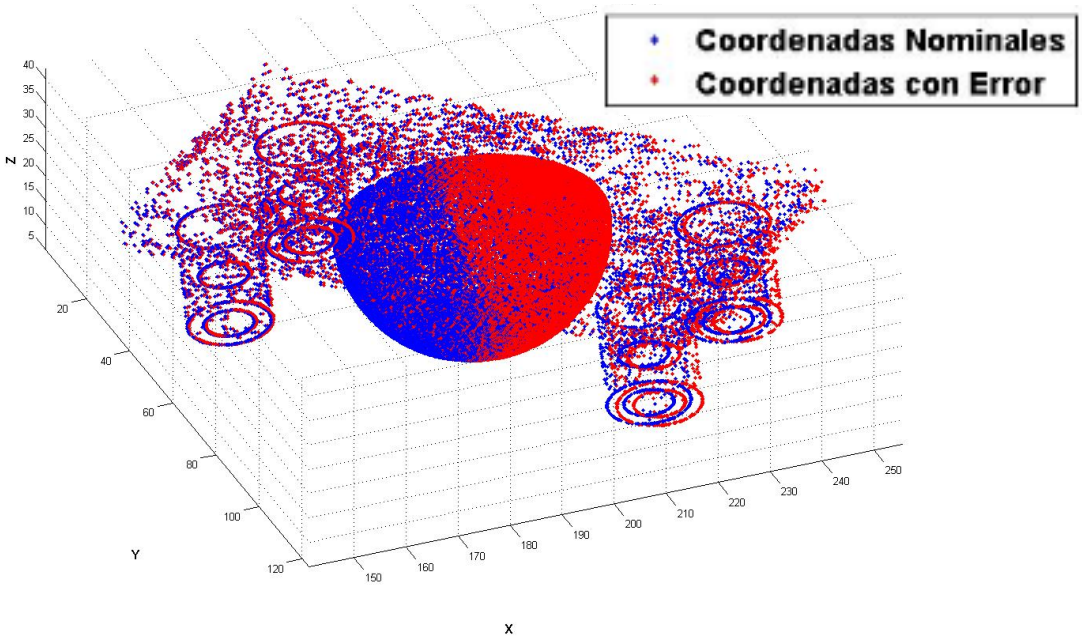


Fig. 31. Estrategia para la tolerancia de planitud aceptando solo soluciones factibles.

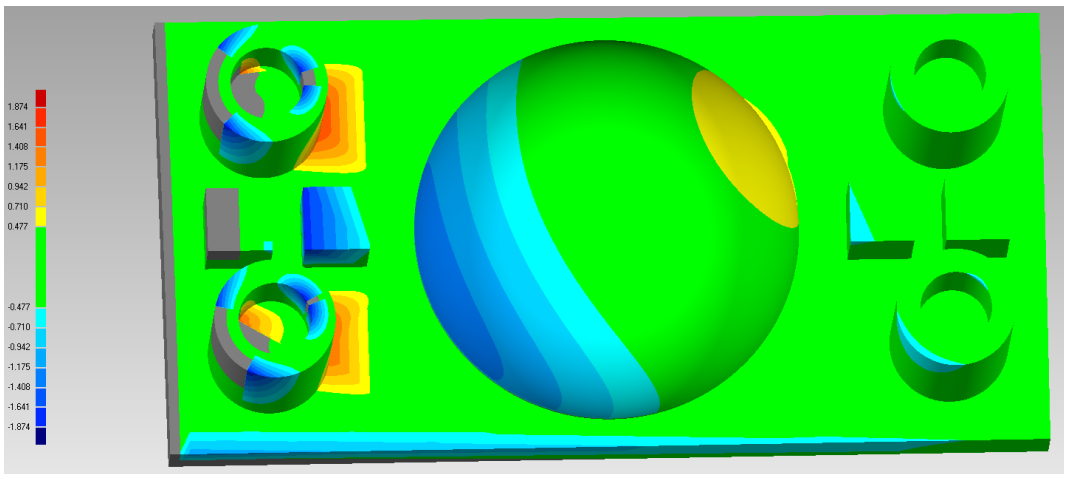


Fig. 32. Error entre la pieza real y la nominal.

Pos. (mm)	**X**	183,28	**Objetivo**	**Tolerancia Planitud (mm)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	1,32					
	Z	33,68					
Or.	**X**	1,72	0,1417	±0,1343	1,047	2,466	21 h. 7 min.
	Y	174,78					
	Z	129,32					

Tabla 20. Resultados de la estrategia con todas las soluciones factibles para la tolerancia de planitud.

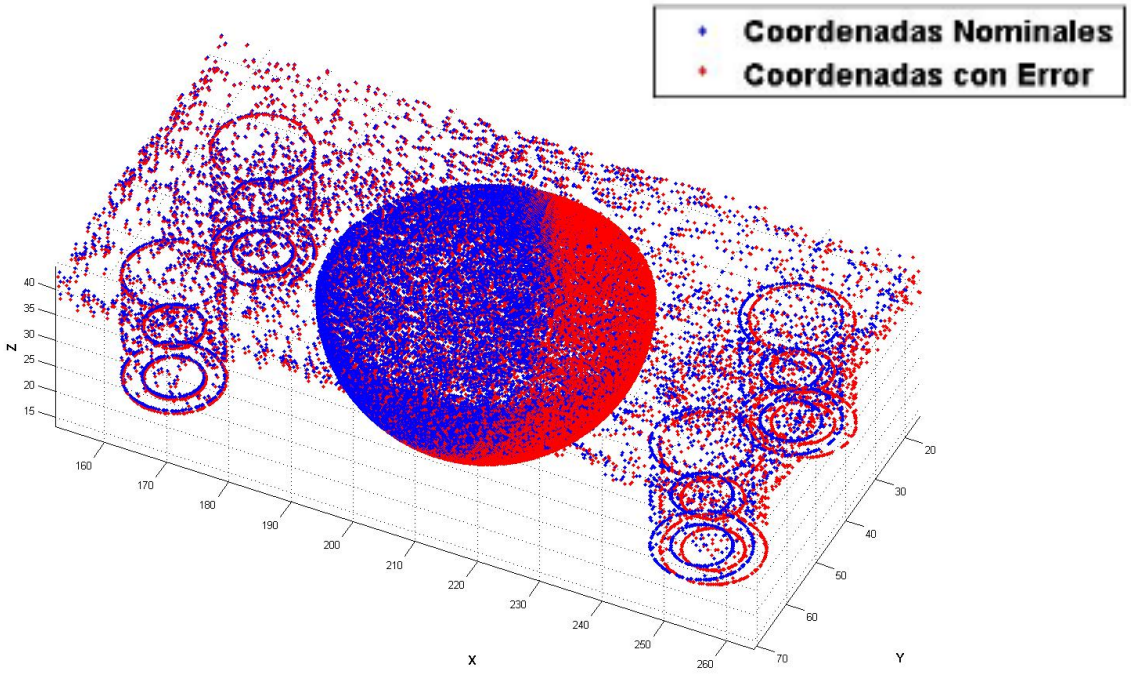


Fig. 33. Estrategia para la tolerancia de planitud aceptando soluciones no factibles.

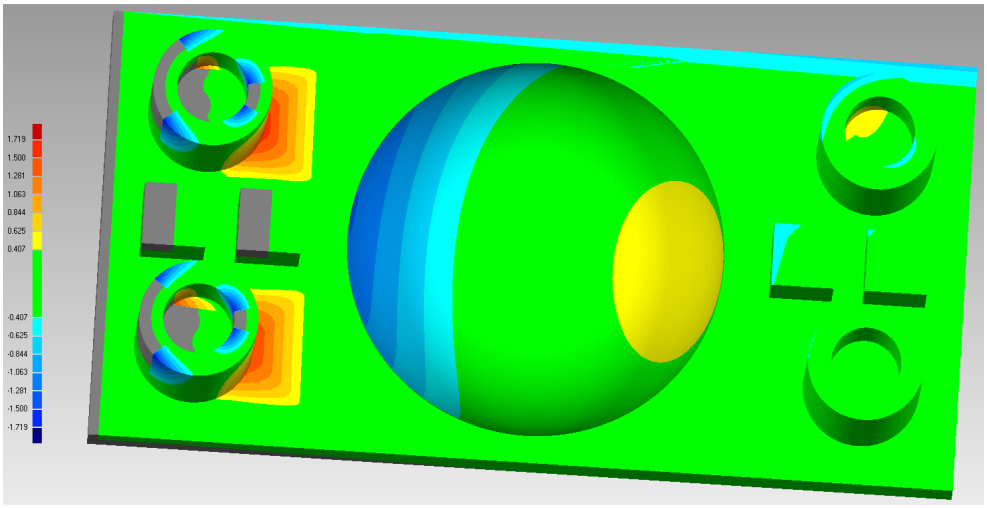


Fig. 34.Error entre la pieza real y la nominal.

Pos. (mm)	**X**	152,29	**Objetivo**	**Tolerancia Planitud (mm)**	**Error medio (mm)**	**Error máximo (mm)**	**Tiempo computacional**
	Y	15,24					
	Z	44,23					
Or.	**X**	180	0,1778	±0,1696	0,954	2,236	24 h. 54 min.
	Y	0					
	Z	0					

Tabla 21. Resultados de la estrategia con soluciones no factibles para la tolerancia de planitud.

Como se puede apreciar en ambos casos la solución obtenida es mejor que la anterior pero se vuelve a ver un error medio que no es muy aceptable. Comparando las dos imágenes se puede observar que conforme el cabezal de la máquina se aleja en el eje X el error aumenta, de ahí que haya predominancia en la parte derecha de la esfera de puntos rojos.

Analizando las cuatro estrategias aquí mostradas, se puede concluir que en la tolerancia de planitud la mejor estrategia es colocar la pieza invertida aunque al igual que en otras estrategias la pieza vuelve a estar situada fuera del volumen donde se ha definido el modelo de error.

6.6 COMPARACIÓN DE RESULTADOS

En este apartado se recopilan todos los resultados en la Tabla 21 y la Tabla 22 y se sacarán las conclusiones generales.

En la Tabla 21 se puede apreciar que en todas las tolerancias experimentadas excepto en la de planitud, el error máximo y el error medio son mayores cuando se emplea el algoritmo que acepta soluciones no factibles pero el valor de tolerancia obtenido con esta versión del algoritmo es menor (excepto en la tolerancia de concentricidad). Por esta razón es importante llegar a un equilibrio entre la tolerancia y el error máximo. Si se tiene en cuenta este criterio se podría concluir que el algoritmo que solo acepta soluciones factibles obtiene mejores estrategias sacrificando un poco la tolerancia pero obteniendo errores menores.

				Objetivo	Tolerancia (mm)	Error medio (mm)	Error máximo (mm)
CILINDRICIDAD	**Todas factibles**	**Pos.**	185	0,0428	±0,0408	0,244	0,63
			56,64				
			51,7				
		Or.	180				
			10,8				
			179,17				
	No todas factibles	**Pos.**	215,41	0,0407	±0,0245	0,459	1,45
			49				
			61,05				
		Or.	43				
			179,56				
			0				
CONCENTRICIDAD	**Todas factibles**	**Pos.**	24,72	5,45E-06	±5,19E-06	0,383	1,163
			10,22				
			34,79				
		Or.	14,02				
			180				
			180				
	No todas factibles	**Pos.**	170,02	4,662E-05	±3,77E-05	1,542	3,251
			106,72				
			45,37				
		Or.	178,59				
			157,7				
			177,35				
ESFERICIDAD	**Todas factibles**	**Pos.**	43,12	0,2561	±0,2561	0,203	0,543
			0				
			31,05				
		Or.	0				
			180				
			180				
	No todas factibles	**Pos.**	176,27	0,2281	±0,2164	0,3492	1,024
			0				
			37,13				
		Or.	138,32				
			180				
			0				
PERPENDICULARIDAD	**Todas factibles**	**Pos.**	192,02	0,05797	±0,0353 °	1,475	3,653
			22,35				
			0				
		Or.	0				
			0				
			110,9				
	No todas factibles	**Pos.**	139,24	7,68E-04	±4,59E-04 °	2,033	3,769
			140,29				
			2,66				
		Or.	178,54				
			175,79				
			114,88				
PLANITUD	**Todas factibles**	**Pos.**	183,28	0,1417	±0,1343	1,047	2,466
			1,32				
			33,68				
		Or.	1,72				
			174,78				
			129,32				
	No todas factibles	**Pos.**	152,29	0,1778	±0,1696	0,954	2,236
			15,24				
			44,23				
		Or.	180				
			0				
			0				

Tabla 22. Comparación de resultados.

			TOLERANCIAS				
			CILINDRICIDAD (mm)	**CONCENTRICIDAD (mm)**	**ESFERICIDAD (mm)**	**PERPENDICULARIDAD (°)**	**PLANITUD (mm)**
OBJETIVOS	**CILINDRICIDAD**	**Todas factibles**	±0,0408	±0,0043	±0,2679	±0,4569	±0,2633
		No todas factibles	±0,0245	±0,002	±0,3021	±1,3066	±0,3781
	CONCENTRICIDAD	**Todas factibles**	±0,1169	±5,19E-06	±0,378	±1,0134	±0,3963
		No todas factibles	±0,2468	±3,77E-05	±0,8475	±1,3187	±0,7042
	ESFERICIDAD	**Todas factibles**	±0,0933	±0,0021	±0,2561	±0,1813	±0,157
		No todas factibles	±0,0921	±0,0045	±0,2164	±1,0796	±0,2402
	PERPENDICULARIDAD	**Todas factibles**	±0,3364	±0,0225	±0,7511	±0,0353	±0,438
		No todas factibles	±0,2356	±0,0161	±0,8251	±4,59E-04	±0,6505
	PLANITUD	**Todas factibles**	±7,68	±5,2183	±0,5194	±31,4592	±0,1343
		No todas factibles	±0,2412	±0,089	±2,47E+12	±0,5109	±0,1696

Tabla 23. Todas las tolerancias con cada estrategia.

Observando los datos recogidos en la Tabla 22 se puede concluir que cuanto más se reduce el valor del objetivo, el efecto que esto provoca en el resto de tolerancias es inverso. Si se

observan los resultados de el objetivo con una y otra versión del algoritmo se aprecia que en aquella versión que más ha reducido el objetivo, el resto de tolerancias tienen un valor más elevado. Este hecho tiene mucho que ver con las conclusiones obtenidas de la Tabla 21, en las que se observaba como al reducir el valor del objetivo el error de la pieza aumentaba.

Si se compara el valor de cada tolerancias en cada objetivo, es posible ver que minimizando la tolerancia de esfericidad se obtienen resultados mejores y más próximos a los mínimos obtenidos en cada objetivo en todas las tolerancias. Como se ha comentado esto se debe a que el error cometido en esta estrategia es menor, al emplear una mayor cantidad de puntos, por lo tanto es más preciso en toda la pieza.

Otra conclusión importante está en el algoritmo. En la Tabla 21 si se observan los valores de posición y orientación de las diferentes estrategias, es posible apreciar tanto valores exactos (0 ó 180) y valores con decimales. Estos resultados del algoritmo son importantes porque demuestran que es capaz de obtener resultados en extremos de los intervalos de las variables independientes pero también obtener valores con más decimales y precisión.

7 CONCLUSIONES Y LINEAS FUTURAS

Tras realizar toda la experimentación y obtener los resultados mostrados anteriormente en este capítulo se van a recoger las conclusiones que se pueden sustraer de éstos.

7.1 CONCLUSIONES

Una de las observaciones más destacadas de los resultados radica en el número de puntos con los que trabaja el algoritmo. Como se ha observado en los resultados de la tolerancia de esfericidad, que sin duda son los mejores resultados obtenidos, a un mayor número de puntos el algoritmo coloca mucho mejor los puntos y los errores cometidos a la hora de fabricar la pieza se reducen de forma considerable si los comparamos con otras tolerancias.

El principal inconveniente de trabajar con un elevado número de puntos es el tiempo computacional.

TIEMPO COMPUTACIONAL		
Tolerancia	**Sin tener en cuenta el error máximo**	**Teniendo en cuenta el error máximo**
Cilindricidad	9 horas 46 minutos	86 horas 54 minutos
Concentricidad	4 horas	167 horas 23 minutos
Esfericidad	100 horas	140 horas 21 minutos
Perpendicularidad	13 horas 45 minutos	73 horas 30 minutos
Planitud	112 horas 41 minutos	21 horas 30 minutos

Tabla 24 Tiempos computacionales

Como se puede ver en la Tabla 22, en la que se adjuntan la media de los tiempos computacionales de ambos algoritmos , si se tiene en cuenta el error máximo de la pieza el tiempo se dispara. Esto se debe a que para calcular el error máximo hay que calcular el error de cada uno de los puntos de la pieza, si no se tiene en cuenta el error máximo el algoritmo solo trabaja con los puntos de la geometría de la que se quiere calcular la tolerancia lo cual reduce drásticamente los puntos.

El hecho de tener tiempos computacionales tan elevados ha impedido la realización de un diseño de experimentos.

Otra conclusión que se puede sacar con los resultados obtenidos es que de los dos algoritmos empleados (aceptando soluciones no factibles y aceptando exclusivamente soluciones factibles) no hay uno mejor que otro, ya que en algunos casos las mejores soluciones las ha obtenido uno y en otras ocasiones las ha obtenido el otro. Como se ha comentado a lo largo del apartado anterior la solución obtenida en un caso no tiene porque ser la mejor, no solo porque se tiene en cuenta también el error máximo de la pieza sino que el valor de tolerancia mínimo que ha podido encontrar en una réplica puede que sea un mínimo relativo y haya otra estrategia en la cual esa tolerancia sea menor. También hay que recalcar que en toda la experimentación se ha trabajado para minimizar al máximo esa tolerancia pero es posible trabajar contra una meta. No siempre es beneficioso reducir al mínimo un valor siendo que con un valor mayor de tolerancia ya se pueden cumplir las exigencias del cliente.

Ya que como se acaba de decir una solución no tiene porque ser la mejor, sino que es la mejor que ha encontrado en esa réplica el algoritmo, los límites fijados en el algoritmo para cada una de las tolerancias pueden ser variables. Con esto lo que se quiere decir es que es posible que en otra réplica el algoritmo encuentre una solución mejor por lo tanto habría que recalcular los límites teniendo en cuenta esa solución.

Por último si apreciamos las imágenes de cada estrategia se puede llegar a una conclusión sobre el modelo cinemático de la máquina. A pesar de que el modelo esta ajustado para solo una parte

del volumen de trabajo de la impresora se observa en la mayoría de las imágenes que cuanto más nos alejamos del origen de referencia de la bandeja de la impresora mayor es el error cometido por ésta. Por lo tanto lo más aconsejable a la hora de fabricar las diversas piezas sería buscar una estrategia que situara la pieza cerca del origen aunque también se ha comprobado que no siempre estar cerca del origen nos da un mínimo de la tolerancia con la que estamos trabajando.

7.2 LÍNEAS FUTURAS

Debido a la inmensidad de ramas que se pueden abordar en este tema en este proyecto nos hemos centrado en la creación de un algoritmo de optimización que tuviera en cuenta el modelo cinemático de error de la máquina. La ventaja de este algoritmo es que funciona con cualquier máquina, simplemente hay que sustituir el modelo de error.

En este proyecto solo te han tenido en cuenta los errores producidos por la propia máquina, una pequeña parte de todo lo que se podría realizar en un futuro.

En primer lugar una de las labores que quedan pendientes es la revisión exhaustiva del modelo cinemático de error de la máquina con el objetivo de poder ajustarlo a todo el volumen de trabajo de la impresora siendo de esta manera más fiable el resultado del algoritmo.

Por otro lado siguiendo con la misma línea de trabajo realizar un diseño de experimentos que nos permita averiguar si algún factor influye en el cálculo del error y en tal caso cuantificarlo.

Por último pero no por ello menos importante otra línea sobre la que se ha empezado a trabajar levemente es la introducción de errores propios de la tecnología. Como se ha comentado en la introducción todos los artículos leídos tratan de optimizar errores de forma de una determinada pieza pero teniendo en cuenta que en este tipo de tecnología la fabricación es mediante adición de capas y esto también produce un error. Por lo tanto el objetivo es introducir el error producido por el espesor de capa dando lugar a zonas escalonadas y rugosas que afectan directamente a la tolerancia de la geometría. No solo se esta trabajando en este error de la tecnología sino que también se esta intentando implantar el error de altura producido por la misma razón, las capas.

Además de estos dos errores típicos de la fabricación aditiva también es muy importante tener en cuenta el material base, es decir, ese material de soporte necesario para fabricar partes de la pieza. En toda la experimentación se presupone que la máquina puede fabricar la pieza en cualquier zona del volumen de trabajo pero hay que tener en cuenta que todo el volumen que hay entre la pieza y la bandeja de la impresora debe ser rellenado con material base. Este es un factor importante que se deberá tener en cuenta en líneas futuras.

8 BIBLIOGRAFIA

[1] Bernad Miana, M., & Oliveros Colay, M. (Diciembre de 2011). Optimización del proceso de embutidción de chapa mediante un algoritmo de colonia de hormigas. *Trabajo Fin de Master* .

[2] Bo Kim, T., Yue, S., Zhang, Z., Jones, E., Jones, J., & Lee, P. (2014). Additive manufatured porous titanium structures: Through-process quantification of pore and strut networks. *Journal of Materials Processing Technology* , 2706-2715.

[3] Cajal Hernando, C. (4 de Abril de 2014). Modelado cinemático, verificación y compensación de error de sistemas de fabricación aditiva para prototipado rápido. *Tesis Doctoral* . Zaragoza, Zaragoza, España.

[4] Calignano, F. (2014). Design optimization of supports overhanging structures in aluminum and titanium alloys by selective laser melting. *Materials and Design* , 203-213.

[5] Feng, L., Wei, S., & Yongnian, Y. (2001). Optimization with minimum process error for layerd manufacturing fabrication. *Rapid Prototyping Journal* , 73-81.

[6] Fernández-Vargas, J., & Bonilla-Petriciolet, A. (2014). Desarrollo de un algoritmo de optimización global en colonias de hormigas con seleccion de región factible para espacios continuos. *Resvista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingenieria* , 178-187.

[7] Haipeng, P., & Tianrui, Z. (2007). Generation and optimization of slice profile data in rapid prototyping and manufaturing. *Journal of Materual Processing Technology* , 623-626.

[8] Koç, M., & Özel, T. (2011). *MICRO-MANUFACTURING: Design and Manufacturing of Micro-Products.* New Jersey: Wiley.

[9] Kong, M., & Tian, P. (2006). A Direct Application of Ant Colony Optimization to Function Optimization Problem in Continuous Domain. 324-331.

[10] Kong, M., & Tian, P. (2006). Application of ACO in Continuous Domain. 126-135.

[11] Kumar Mushra, A., & Thirumavalavan, S. (2014). A Study of Part Orientation in Rapid Prototyping. *Middle-East Journal of Scientific Research* , 1197-1201.

[12] Li, Y., & Zhang, J. (2013). Multi-criteria GA-based Pareto optimization of building direction for rapid prototyping. *Internation Journal Advanced Manufaturing Technology* , 1819-1831.

[13] Liao, T., Stützle, T., Mondes de Oca, M., & Dorigo, M. (2014). A unified ant colony optimization algorithm for continuous optimization. *European Journal of Operational Research* , 597-609.

[14] Moore, H. (2007). *MATLAB para ingenieros.* México: Pearson educación.

[15] Paul, R., & Anand, S. (2011). Optimal part orientation in Rapid Manufacturing process for achieving geometric tolerances. *Journal of Manufacturing Systems* , 214-222.

[16] Paul, R., & Anand, S. (2014). Optimization of layered manufacturing process for reducing form error with minimal support structures. *Journal of Manufacturing Systems* .

[17] Phatak, A., & Pande, S. (2012). Optimum part orientation in Rapid Prototyping using genetic algorithm. *Journal of Manufacturing Systems* , 395-402.

[18] Salama, K., Abdelbar, A., Otero, F., & Freitas, A. (2013). Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery. *Applied Soft Computing* , 667-675.

[19] Simonelli, M., Tse, Y., & Tuck, C. (2014). Effect of built orientation on the mechanical properties and fracture modes of SLM Ti-6Al-4V. *Material Science & Engineering A* , 1-11.

[20] Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research* , 1155-1173.

[21] Szu-Shen Chen, J., & Feng, H.-Y. (2011). Optimal layer setup generation in layered manufacturing with a given error constraint. *Journal of Manufacturing Systems* , 165-174.

[22] Thrimurthulu, K., Pandey, P., & Reddy, N. (2004). Optimum part deposition orientation in fused deposition modeling. *International Jounal of Machine Tools & Manufacture* , 585-594.

[23] Ulusam Seçkiner, S., Eroglu, Y., Emrullah, M., & Dereli, T. (2013). Ant colony optization for continuous function by using novel pheromone updating. *Applied Mathematics and Computation* , 4163-4175.

9 ANEXOS

9.1 ANEXO A

En este anexo se va a mostrar el código del algoritmo y de las funciones objetivo con las explicaciones pertinentes.

9.1.1 CÓDIGO DEL ALGORITMO DACO

Como se ha ido comentando se han desarrollado dos códigos de optimización. La diferencia entre ellos radica en las restricciones que tiene cada uno frente a cada solución.

En ambas versiones del código la lectura de los datos es la misma. En un primer momento partíamos de un stl para leer los datos. Debido al elevado número de puntos necesarios para definir la pieza mediante un stl se decidió trabajar con menos puntos y por lo tanto fue necesario trabajar con un Excel que tuviera los puntos.

```
%syms lb ub
t0CPU=cputime;
n_run=1; % Número de réplicas de DACO a realizar
nsolugen=0;
% importamos desde una excel los puntos de la pieza
Pexcel1 = xlsread('PiezaTest1.xlsx');
n2 = size(Pexcel1,1); % número de puntos que hay en la excel
Pinicial(:,1:3)=Pexcel1(:,1:3);
Pinicial(:,4)=1;
box = boundingBox(Pinicial);
% importamos desde una excel los puntos
Pexcel = xlsread('CilindroExtTest.xlsx');
n2_geo = size(Pexcel,1); % número de puntos que hay en la excel
P_geo(:,1:3)=Pexcel(:,1:3);
P_geo(:,4)=1;
% parámetros de la geometria
x0(1,1)=mean(P_geo(:,1));
x0(2,1)=mean(P_geo(:,2));
x0(3,1)=mean(P_geo(:,3));
x0(4,1)=1;
a0=[0; 0; 1];
r0=7.5;
tolp=0.05;
tolg=0.05;
```

En esta primera parte del código se leen todos los puntos de la pieza y posteriormente se leen los puntos de la geometría con la que vamos a trabajar, en este caso los puntos del cilindro.

Además la variable “box” recoge la bounding box de la pieza lo que nos permitirá comprobar en cada estrategia que la pieza esta dentro del volumen de trabajo de la máquina.

```
% importamos el modelo de error desde un txt
Merror = uigetfile('*.txt','Seleccione parametros del modelo de error');
parametros3 = importdata(Merror);
n3 = size(parametros3);
```

```
X = parametros3(n3(1),:);
```

Con estas líneas del código introducimos el modelo cinemático de error de la máquina.

Estas primeras líneas del código son simplemente lectura de datos que para ambas versiones son iguales. A continuación se adjunta el resto del código de una de las versiones y posteriormente se adjuntarán las partes que difieren de la otra versión.

9.1.1.1 Algoritmo aceptando solo soluciones factibles

En esta versión del algoritmo las primeras 20 soluciones que obtiene cada una de las hormigas tienen que ser todas soluciones factibles, es decir todas ellas deben estar dentro del volumen de trabajo de la máquina.

```
for k=1:n_run
   replica=k;

   clear MR MRp MRrabs Sgb cSgb cf cfr fid h iter itermax lb mu mu_0 ...
   mu_0 n nc nf rho sigma sigma_0 status sumrSgbr ub v vv v_cf vpSgbr ...
   vpSgbr vrSgbr ans

   %PARAMETROS DE ENTRADA AL PROGRAMA
   n=6; % Número de variables
   lb=[0 0 0 0 0 0]; % vector limite inferior
   ub=[300 300 150 180 180 180]; % vector limite superior
   EMAX=1.543; % Error maximo permitido para aceptar la solución
   LIM=0.543; % Error maximo permitido a partir del cual se penaliza la solución
   h=20; % Número de hormigas
   rho=0.85; % Factor de evaporación
   itermax=100; % Iteraciones máximas
   cfr=0.05; % Factor salida de minimo local (varía 1->0)
   MR=[]; % Inicialización de matriz de soluciones reinicializadas
   v_min_contraste=0; % Introducir valor referencia para error

   %INICIALIZACION DE VECTORES mu_0 y sigma_0
   %rand genera PSEUDOALEATORIOS con PDF uniforme [0,1]
   for k=1:n;
      mu_0(k)=lb(k)+(ub(k)-lb(k))*rand;
      sigma_0(k)=(ub(k)-lb(k))/2;
   end
   clear k

   % hacemos un primer cálculo de la bounding box para que salga fuera de
   % los límites de nuestra impresora. A partir de aquí comprobaremos en
   % cada solución que la pieza está dentro de la bounding box de la
   % máquina
   obox=OrientacionBox(mu_0,box);

   %GENERAMOS PRIMERA SOLUCION GLOBAL BEST Sgb_0
   %Primera seleccion aleatoria de variables para las hormigas.  Generamos en
   %forma de matriz. Cada fila es una solución generada por una hormiga.
   %Las columnas 1:n son componentes solucion y columna n+1 es la
   %solución asociada a esos componentes solución.
   for k=1:h; % h hormigas
```

```
        obox(2)=obox(2)+300;
        while(obox(1)<lb(1))|(obox(2)>ub(1))|(obox(3)<lb(2))|(obox(4)>ub(2))| (obox(5)<
        lb(3))|(obox(6)>ub(3)|(error>EMAX))
           for k1=1:n; % n variables
              rnd=randn; % Aleatorio distribución normal [0,1]
              aleat=mu_0(k1)+sigma_0(k1)*rnd; %Aleatorio (lb,ub)
              if aleat>=lb(k1) & aleat<=ub(k1);
                 MS_0(k,k1)=aleat;
              elseif aleat<lb(k1);
                 MS_0(k,k1)=lb(k1);
              else aleat>ub(k1);
                 MS_0(k,k1)=ub(k1);
              end
           end
           error=EMax(MS_0(k,[1:n]),X,Pinicial,n2);
           obox=OrientacionBox(MS_0(k,[1:n]),box);
        end
MS_0(k,k1+1)=CilindricidadMejorada(MS_0(k,[1:n]),X,P_geo,n2_geo,error,LIM,EMAX,x0,
a0,r0,tolp,tolg); %Calculo solución asociada y asigno a MS_0
        nsolugen = nsolugen +1;
     end
     MS_0=sortrows(MS_0,n+1); % Ordeno por solucion. La mínima la primera
     Sgb_0=MS_0(1,[1:n+1]); % Selecciono la mejor (c_solucion y solucion) como Sgb
     clear k k1 rnd aleat MS_0
```

En esta primera parte del código se calculan una solución por cada hormiga y se selecciona la mejor que se fijará como Sgb_0.

```
     %ACTUALIZACION FEROMONA. Primera antes de comenzar a iterar
     mu=(1-rho)*mu_0+rho*Sgb_0([1:n]); %iniz mu para iterar
     sigma=(1-rho)*sigma_0+rho*abs(Sgb_0([1:n])-mu_0);%iniz sigma para iterar

     %COMENZAMOS EL PROCESO DE ITERACIONES
     iter=1; Sgb=Sgb_0;
     clear Sgb_0

     while iter<=itermax;
      %generamos las matrices de componentes solución y solucion
      cont=0;
      for k=1:h; % h hormigas
        obox(2)=obox(2)+300;
        while (obox(1)<lb(1))|(obox(2)>ub(1))|(obox(3)<lb(2))|(obox(4)>ub(2))|(obox(5)<lb(3))|
(obox(6)>ub(3)|(error>EMAX))
           cont=cont+1;
           for k1=1:n; % n variables
              rnd=randn;% Aleatorio distribución normal [0,1]
              aleat=mu(k1)+sigma(k1)*rnd; %Aleatorio (lb,ub)
              if aleat>=lb(k1) & aleat<=ub(k1);
                 MSit(k,k1)=aleat;
              elseif aleat<lb(k1);
                 MSit(k,k1)=lb(k1);
              else aleat>ub(k1);
                 MSit(k,k1)=ub(k1);
              end
```

```
        end
        error=EMax(MSit(k,[1:n]),X,Pinicial,n2);
        obox=OrientacionBox(MSit(k,[1:n]),box);
      end
MSit(k,k1+1)=CilindricidadMejorada(MSit(k,[1:n]),X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r
0,tolp,tolg);
      nsolugen = nsolugen +1;
    end
    clear k k1 aleat obox
    MSit=sortrows(MSit,n+1);% Matriz ordenada de soluciones para la iteracion.
    Sitb=MSit(1,[1:n+1]);% Seleccion de mejor solucion de la matriz.
```

Con la actualización de la feromona se vuelve a calcular una solución para cada hormiga y se vuelve a seleccionar la mejor (Sitb). En este momento comienza el doble proceso de mejora local tanto para Sgb_0 como para Sitb. Solo vamos a mostrar uno de los 4 bucles ya que son iguales todos.

```
    %Aplicar mejora local ML1 a Sitb
    cSitbML1_0=Sitb([1:n]); % Definición de componentes
    rSitbML1_0=Sitb(n+1); % Definición de resultado
    cSitbML1_n=cSitbML1_0; % Inicializo cSitML1_n con las componentes de cSitML1_0

    for k=1:n;
      rnd=randn; % Aleatorio distribución normal [0,1]
      cSitbML1_n(k)=cSitbML1_0(k)+rnd*sigma(k); % UPDATE componente (k) de cSn
      error=EMax(cSitbML1_n,X,Pinicial,n2);
      obox=OrientacionBox(cSitbML1_n,box);
      if
(cSitbML1_n(k)>=lb(k)&cSitbML1_n(k)<=ub(k)&(obox(1)>=lb(1))&(obox(2)<=ub(1))&(obox
(3)>=lb(2))&(obox(4)<=ub(2))&(obox(5)>=lb(3))&(obox(6)<=ub(3))&(error<=EMAX));

rSitbML1_n=CilindricidadMejorada(cSitbML1_n,X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r0,t
olp,tolg);
        nsolugen = nsolugen +1;
        while rSitbML1_n<rSitbML1_0;
          rSitbML1_0=rSitbML1_n;
          cSitbML1_0(k)=cSitbML1_n(k);
          rnd=randn;
          cSitbML1_n(k)=cSitbML1_0(k)+rnd*sigma(k);
          error=EMax(cSitbML1_n,X,Pinicial,n2);
          obox=OrientacionBox(cSitbML1_n,box);
          if
(cSitbML1_n(k)>=lb(k)&cSitbML1_n(k)<=ub(k)&(obox(1)>=lb(1))&(obox(2)<=ub(1))&(obox
(3)>=lb(2))&(obox(4)<=ub(2))&(obox(5)>=lb(3))&(obox(6)<=ub(3))&(error<=EMAX));

rSitbML1_n=CilindricidadMejorada(cSitbML1_n,X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r0,t
olp,tolg);
            nsolugen = nsolugen +1;
          else
            cSitbML1_n(k)=cSitbML1_0(k);
          end
        end
      else
        cSitbML1_n(k)=cSitbML1_0(k);
      end
```

```
end
SitbML1=[cSitbML1_0 rSitbML1_0];
clear k Sitb obox
```

Tras las mejoras locales se selecciona entre esas dos soluciones mejoradas y el resto de soluciones creadas por las hormigas y que no han sido empleadas la mejor de todas ellas, la cual será la nueva Sgb_0 en la siguiente iteración.

```
%Formar conjunto [Sitb_mej] [Sgb_mej] [(h-1) restantes iteración]
%Ordenar y seleccionar la mejor como nueva Sgb
MSit_r=MSit([2:h],:);
MSG=[MSit_r;SgbML2;SitbML2];
MSG=sortrows(MSG,n+1);
Sgb=MSG(1,[1:n+1]);
cSgb=Sgb([1:n]);
% linea introducida para calcular el error máximo de todos los puntos
[desv maxd mind rad] = CilindroDesv(cSgb,X,P_geo,n2_geo,x0,a0,r0,tolp,tolg);
error=EMax(cSgb,X,Pinicial,n2);
clear MSit MSit_r MSG SgbML1 SgbML2 SitbML1 SitbML2
clear cSgbML1_0 cSgbML1_n cSgbML2_0 cSgbML2_n cSitbML1_0 cSitbML1_n
clear cSitbML2_0 cSitbML2_n rSgbML1_0 rSgbML1_n rSgbML2_0 rSgbML2_n
clear rSitbML1_0 rSitbML1_n rSitbML2_0 rSitbML2_n
```

Tras seleccionar la mejor solución de todas comienza el proceso de actualización de la feromona dependiendo de si el algoritmo ha convergido o no.

```
%ANALISIS DEL MINIMO LOCAL Y ACTUACIONES NECESARIAS
v_cf=(2*sigma)./(ub-lb);
cf=sum(v_cf)/n;
if cf>cfr; % Implica que la solucion NO ha convergido
   sigma=(1-rho)*sigma+rho*abs(cSgb-mu);%primero por la "mu"
   mu=(1-rho)*mu+rho*cSgb;
else % Implica que la solucion SI ha convergido
   if size(MR)==[0 0]; % Si es la primera vez que se reinicia
      MR=Sgb; % Genero la matriz de soluciones reiniciadas MR
      [nf nc]=size(MR); % Tomamos dimensiones matriz
   else % Si NO es la primera vez que se reinicia
      MR=[MR;Sgb]; % Amplio MR con la nueva solucion reiniciada
      [nf nc]=size(MR); % Tomamos dimensiones matriz
   end
   vrSgbr=MR([1:nf],nc)'; %Vector resultados de Sgb reiniciadas
   sumrSgbr=sum(vrSgbr); %Suma vector resultados
   vpSgbr=vrSgbr./sumrSgbr; %vector pesos de Sgb reiniciadas (w en DACO)
   MRp=[MR vpSgbr']; %Amplio MR con vector pesos
   for k=1:nc-1; % Genero vector dim(n) usando dot(v_peso,v_c_sol(k))
      v(k)=dot(MRp([1:nf]),MR([1:nf],k));
   end %
   clear k
   %ACTUALIZACION DE MU
   mu=mu_0+rho*dot(vpSgbr,vrSgbr);

   for k=1:nc-1 %Genero matriz v_abs MRr-cSgb
      MRrabs([1:nf],k)=abs(MRp([1:nf],k)-cSgb(k));
   end
```

```
        clear k
        for k=1:nc-1;
            vv(k)=dot(MRp([1:nf]),MRrabs([1:nf],k));
        end
        clear k
        %ACTUALIZACION DE SIGMA
        sigma=sigma_0+rho*dot(vpSgbr,abs(vrSgbr-Sgb(1,7)));
    end
```

Y para finalizar la iteración se genera un archivo que nos guarda los resultados de cada iteración en un archivo txt

```
    %GENERACION ARCHIVO HISTORICO DE LA ITERACION
    fid = fopen('historico_iter.txt', 'a');
    fprintf(fid, '%10.10f \t', [replica iter Sgb desv rad maxd mind error]);
    fprintf(fid, '\r\n');
    status = fclose(fid);

    iter=iter+1;

   end % FIN DE BUCLE DACO
end %FIN DE BUCLE PARA REPLICAS
clear k replica
```

Tras la realización de todas las iteraciones se hace un análisis de los datos obtenidos que no se adjunta. El análisis de datos nos crea otro archivo txt en el cual aparece cual ha sido la mejor solución de todas las iteraciones.

Y por último dibujamos la mejor solución obtenida por el algoritmo.

```
%dibujamos las mejores soluciones de cada réplica
Perror = DibujarStl(MConv,X,Pinicial);
```

9.1.1.2 Algoritmo aceptando soluciones no factibles

En esta versión del algoritmo se aceptan 8 soluciones no factibles, es decir que no estén dentro del volumen de trabajo de la impresora. Solo se van a mostrar aquellas partes del código que difieran de la versión anterior.

En primer lugar en los parámetros de entrada hay que añadir un error. Este error se le sumará a aquellas soluciones iniciales que no sean factibles.

```
   ENF=50; % Error añadido a soluciones no factibles
```

Las primeras 20 soluciones de las hormigas al igual que en algoritmo anterior tienen que ser todas válidas. Se aceptan soluciones no factibles cuando comenzamos a iterar e incluso en las mejoras locales.

```
while iter<=itermax;
    %generamos las matrices de componentes solución y solucion
    cont=0;
    nofactible=0;
    for k=1:h; % h hormigas
```

```
    if nofactible<8;
       for k1=1:n; % n variables
           rnd=randn;% Aleatorio distribución normal [0,1]
           aleat=mu(k1)+sigma(k1)*rnd; %Aleatorio (lb,ub)
           if aleat>=lb(k1) & aleat<=ub(k1);
             MSit(k,k1)=aleat;
           elseif aleat<lb(k1);
             MSit(k,k1)=lb(k1);
           else aleat>ub(k1);
             MSit(k,k1)=ub(k1);
           end
       end
       error=EMax(MSit(k,[1:n]),X,Pinicial,n2);
       obox=OrientacionBox(MSit(k,[1:n]),box);
       if
(obox(1)<lb(1))|(obox(2)>ub(1))|(obox(3)<lb(2))|(obox(4)>ub(2))|(obox(5)<lb(3))|(obox(6)>ub(
3)|(error>EMAX));
          nofactible=nofactible+1;
          MSit(k,k1+1)=Cilindricidad(MSit(k,[1:n]),X,P_geo,n2_geo,x0,a0,r0,tolp,tolg)+ENF;
          nsolugen = nsolugen +1;
       else
MSit(k,k1+1)=CilindricidadMejorada(MSit(k,[1:n]),X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r
0,tolp,tolg);
          nsolugen = nsolugen +1;
       end
    else obox(2)=obox(2)+300;
       while
(obox(1)<lb(1))|(obox(2)>ub(1))|(obox(3)<lb(2))|(obox(4)>ub(2))|(obox(5)<lb(3))|(obox(6)>ub(
3)|(error>EMAX))
          cont=cont+1;
          for k1=1:n; % n variables
             rnd=randn;% Aleatorio distribución normal [0,1]
             aleat=mu(k1)+sigma(k1)*rnd; %Aleatorio (lb,ub)
             if aleat>=lb(k1) & aleat<=ub(k1);
               MSit(k,k1)=aleat;
             elseif aleat<lb(k1);
               MSit(k,k1)=lb(k1);
             else aleat>ub(k1);
               MSit(k,k1)=ub(k1);
             end
          end
          error=EMax(MSit(k,[1:n]),X,Pinicial,n2);
          obox=OrientacionBox(MSit(k,[1:n]),box);
       end
MSit(k,k1+1)=CilindricidadMejorada(MSit(k,[1:n]),X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r
0,tolp,tolg);
       nsolugen = nsolugen +1;
    end
  end
  clear k k1 aleat
  MSit=sortrows(MSit,n+1);% Matriz ordenada de soluciones para la iteracion.
  Sitb=MSit(1,[1:n+1]);% Seleccion de mejor solucion de la matriz.
```

Como podemos observar la obtención de soluciones sigue el mismo proceso con la diferencia de que dispone de un contador en el que almacena el número de soluciones no factibles encontradas. Una vez llegue a 8 el código es exactamente el mismo que en la versión anterior.

En las mejoras locales el procedimiento es el mismo con la salvedad de que al encontrar una solución no factible se le aplica el error correspondiente, por lo tanto esa solución no factible nunca es mejor que la anterior durante la mejora local.

```
%Aplicar mejora local ML1 a Sitb
cSitbML1_0=Sitb([1:n]); % Definición de componentes
rSitbML1_0=Sitb(n+1); % Definición de resultado
cSitbML1_n=cSitbML1_0; % Inicializo cSitML1_n con las componentes de cSitML1_0

for k=1:n;
  rnd=randn; % Aleatorio distribución normal [0,1]
  cSitbML1_n(k)=cSitbML1_0(k)+rnd*sigma(k); % UPDATE componente (k) de cSn
  if (cSitbML1_n(k)>=lb(k)&cSitbML1_n(k)<=ub(k));
    error=EMax(cSitbML1_n,X,Pinicial,n2);
    obox=OrientacionBox(cSitbML1_n,box);
    if
(obox(1)<lb(1))|(obox(2)>ub(1))|(obox(3)<lb(2))|(obox(4)>ub(2))|(obox(5)<lb(3))|(obox(6)>ub(
3)|(error>EMAX));
      rSitbML1_n=Cilindricidad(cSitbML1_n,X,P_geo,n2_geo,x0,a0,r0,tolp,tolg)+ENF;
      nsolugen = nsolugen +1;
    else
rSitbML1_n=CilindricidadMejorada(cSitbML1_n,X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r0,t
olp,tolg);
      nsolugen = nsolugen +1;
    end
    while rSitbML1_n<rSitbML1_0;
      rSitbML1_0=rSitbML1_n;
      cSitbML1_0(k)=cSitbML1_n(k);
      rnd=randn;
      cSitbML1_n(k)=cSitbML1_0(k)+rnd*sigma(k);
      if (cSitbML1_n(k)>=lb(k)&cSitbML1_n(k)<=ub(k));
        error=EMax(cSitbML1_n,X,Pinicial,n2);
        obox=OrientacionBox(cSitbML1_n,box);
        if
(obox(1)<lb(1))|(obox(2)>ub(1))|(obox(3)<lb(2))|(obox(4)>ub(2))|(obox(5)<lb(3))|(obox(6)>ub(
3)|(error>EMAX));

rSitbML1_n=Cilindricidad(cSitbML1_n,X,P_geo,n2_geo,x0,a0,r0,tolp,tolg)+ENF;
          nsolugen = nsolugen +1;
        else
rSitbML1_n=CilindricidadMejorada(cSitbML1_n,X,P_geo,n2_geo,error,LIM,EMAX,x0,a0,r0,t
olp,tolg);
          nsolugen = nsolugen +1;
        end
      else
        cSitbML1_n(k)=cSitbML1_0(k);
      end
    end
  else
    cSitbML1_n(k)=cSitbML1_0(k);
```

```
    end
  end
  SitbML1=[cSitbML1_0 rSitbML1_0];
  clear k Sitb
```

9.1.2 CÓDIGO DE LAS FUNCIONES OBJETIVO

A continuación se va a mostrar el código de las funciones objetivo. La parte inicial del código es igual para todas ellas ya que es el cálculo de los puntos con error en la estrategia calculada.

```
% pasa sistema referencia pieza a sistema pieza patrón
RT = rpy2tr(t(4),t(5),t(6),'deg');
RT(1,4) = t(1);
RT(2,4) = t(2);
RT(3,4) = t(3);

% pasamos los puntos de la pieza de sus sistema de referencia
% al sistema de referencia de la pieza patron
tPnom=RT*transpose(Pinicial);
Pnom=transpose(tPnom);

% calcula los puntos una vez aplicado el error de la máquina
Pfinal = AplicaErrorStlJuan(X,Pnom);
```

Esta parte del código calcular la matriz de rototranslación a partir de las 6 variables independientes, cambia los puntos de la pieza a ese nuevo sistema de referencia y calcula los puntos con error.

A partir de aquí cada función objetivo es diferente aunque muchas de ellas son parecidas.

9.1.2.1 Error máximo de la pieza

```
function emax = EMax(t,X,Pinicial,n2)

% calculo del error
emax = 0;
for i=1:n2
   en=sqrt((Pfinal(i,1)-Pnom(i,1))^2+(Pfinal(i,2)-Pnom(i,2))^2+(Pfinal(i,3)-Pnom(i,3))^2);
   if en >= emax;
      emax = en;
   end
end
```

9.1.2.2 Error medio de la pieza

```
function e = ErrorTotalStl(t,X,Pinicial,n2)

% calculo del error
suma = 0;
for i=1:n2
   en=sqrt((Pfinal(i,1)-Pnom(i,1))^2+(Pfinal(i,2)-Pnom(i,2))^2+(Pfinal(i,3)-Pnom(i,3))^2);
   suma = suma + en;
end
e = suma/n2;
end
```

Como podemos observar el cálculo del error medio y el error máximo tienen el mismo código.

9.1.2.3 Tolerancia de cilindricidad

Para el cálculo de las tolerancias como se ha dicho a lo largo de la memoria se realiza una primera aproximación y posteriormente un segundo cálculo rechazando los puntos que estén fuer del intervalo ±2σ.

Acto seguido se calcula la tolerancia y se le añade la penalización correspondiente.

A continuación os mostramos una de las funciones completas y en el resto solo se mostrarán las pequeñas variaciones.

```
function e = CilindricidadMejorada(t,X,Pinicial,n2_geo,error,LIM,EMAX,x0,a0,r0,tolp,tolg)

% calculo de la tolerancia
[x0n an rn d sigmah conv Vx0n Van urn GNlog a R0 R] = lscylinder(Pfinal(:,1:3), x0(1:3,1), a0,
r0, tolp, tolg);

k=1;
Pdefinitivos(1,1:3)=0;
for i=1:n2_geo
   if abs(d(i))<=(2*sigmah)
      Pdefinitivos(k,:)=Pfinal(i,:);
      k=k+1;
   end
end

if Pdefinitivos==0
   Pdefinitivos=Pfinal;
end

[x0n an rn d sigmah conv Vx0n Van urn GNlog a R0 R] = lscylinder(Pdefinitivos(:,1:3),
x0(1:3,1), a0, r0, tolp, tolg);

pts=size(Pdefinitivos);
dif=rn-r0;
for i=1:pts(1)
   d(i)=d(i)+dif;
end

e=max(abs(d));

if error>LIM
   pen=(error-LIM)/(EMAX-LIM);
   e=e*1.75^pen;
end

end
```

9.1.2.4 Tolerancia de concentricidad

En esta tolerancia se han de realizar las aproximaciones de los dos cilindros antes de calcular la distancia entre los dos ejes.

```
function dist = ConcentricidadMejorada(t,X,P_cil1,n2_cil1,x1,r1,P_cil2,n2_cil2,x2,r2,error,LIM,EMAX,a0,tolp,tolg)

% calculo de la tolerancia
 [x1n an1 rn d sigmah conv Vx0n Van urn GNlog a R0 R] = lscylinder(Pdefinitivos1(:,1:3), x1(1:3,1), a0, r1, tolp, tolg);

% calculo de la tolerancia
 [x2n an2 rn d sigmah conv Vx0n Van urn GNlog a R0 R] = lscylinder(Pdefinitivos2(:,1:3), x2(1:3,1), a0, r2, tolp, tolg);

% calculo de la distancia entre los dos ejes
x1nx2n=[x2n(1)-x1n(1); x2n(2)-x1n(2); x2n(3)-x1n(3)];
dist=cross(x1nx2n,an1);
dist=norm(dist)/norm(an1);

end
```

9.1.2.5 Tolerancia de esfericidad

```
function e = EsfericidadMejorada(t,X,Pinicial,n2_geo,error,LIM,EMAX,x0,r0,tolp,tolg)


% calculo de la tolerancia
[x0n rn d sigmah conv Vx0n urn GNlog a R]=lssphere(Pdefinitivos(:,1:3),x0(1:3,1),r0,tolp,tolg);

pts=size(Pdefinitivos);
dif=rn-r0;
for i=1:pts(1)
   d(i)=d(i)+dif;
end

e=max(abs(d));

end
```

9.1.2.6 Tolerancia de perpendicularidad

Como en esta tolerancia también entran en juego dos geometrías de la pieza se han de calcular las aproximaciones tanto del cilindro como del plano. De este modo tenemos el vector normal del plano y el eje del cilindro para poder calcular la tolerancia.

```
function e = PerpendicularidadPlanoCilindroMejorada(t,X,P_plano,P_cil,n2_cil,error,LIM,EMAX,x0,a0,r0,tolp,tolg)

% calculo de la tolerancia
```

```
[x0n an rn d sigmah conv Vx0n Van urn GNlog a R0 R] = lscylinder(Pdefinitivos(:,1:3), x0(1:3,1), a0, r0, tolp, tolg);

% calculo de la tolerancia
[x0 a] = lsplane(Pfinal_plano(:,1:3));
a=transpose(a);

e=acos(dotprod(a,an)/(norm(a)*norm(an)))*180/pi;

end
```

9.1.2.7 Tolerancia de planitud

Por último en la tolerancia de planitud se ha de calcular tanto el plano teórico como el plano con error para tener en cuenta la distancia entre los dos planos así como la distancia de cada punto con error al plano con error.

```
function e = PlanitudMejorada(t,X,Pinicial,error,LIM,EMAX)

% calculo del plano teorico
[x0 a0] = lsplane(Pnom(:,1:3));

% calculo de la tolerancia
[x a d] = lsplane(Pfinal(:,1:3));

% distancia entre los planos
syms r;
D=solve(a(1)*x(1)+a(2)*x(2)+a(3)*x(3)+r);
dist=abs(a(1)*x0(1)+a(2)*x0(2)+a(3)*x0(3)+D)/norm(a);

pts=size(Pfinal);
for i=1:pts(1)
   d(i)=d(i)+dist;
end

e=max(abs(d));

end
```