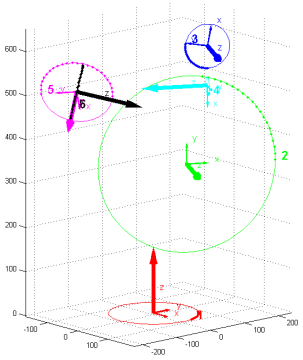
Universidad Zaragoza

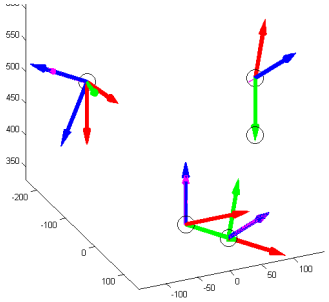
Escuela Universitaria Ingeniería Técnica Industrial ZARAGOZA

Departamento de Ingeniería de Diseño y Fabricación
UNIVERSIDAD DE ZARAGOZA

67100 Trabajo Fin de Máster

IDENTIFICACIÓN DE PARÁMETROS CINEMÁTICOS DE UN BRAZO ROBOT MEDIANTE EL MÉTODO DE CIRCLE-POINT. SIMULACIÓN Y VALIDACIÓN EXPERIMENTAL.





Curso Académico 2010-2011

Titulación: MÁSTER SISTEMAS MECÁNICOS

Programa Oficial de Posgrado: INGENIERÍA MECÁNICA Y DE MATERIALES

Autor: Manuel Ginés Buil

Director: Dr. Jorge Santolaria Mazo

Zaragoza, 12 de Noviembre de 2010

A MªMar, por su amor, comprensión y paciencia durante el Máster, y en la realización de este proyecto, sin cuyo apoyo esto no hubiera sido posible. Este trabajo es en buena parte suyo.

A Álvaro y María, sin quienes todo esto no tendría sentido.

A Pablo, por inspirarme y alentarme para afrontar este reto.

Y un agradecimiento especial a Jorge Santolaria Mazo, por su apoyo y dedicación en la elaboración de este proyecto.

Identificación de Parámetros cinemáticos de un brazo robot mediante el método "Circle Point". Simulación y validación experimental

Resumen

1 Resumen del Proyecto

Este proyecto tiene como objetivo la identificación de los parámetros cinemáticos de un brazo robot mediante el método de *Circle Point*, así como realizar un proceso de simulación que emule la adquisición de los puntos en el espacio (que definen la ubicación de cada articulación) que daría el método de *Circle Point*, y permita obtener las incertidumbres asociadas a los valores individuales de cada parámetro determinado por este procedimiento.

Para llevar a cabo este objetivo, se han seguido los siguientes pasos:

1. Implementación en MATLAB del modelo del robot utilizando los modelos de Denavit-Hartenberg y Hayati-Mirmirani.
2. Implementación en MATLAB de un generador paramétrico de puntos de medida sintéticos (Simulador puntos *Circle Point*) que reproduzca el resultado de un procedimiento de medida según el método *Circle Point*. Este generador permite definir las coordenadas en las que se va colocando el reflector, ángulos girados por cada articulación, número de puntos capturados en cada giro, ruido de medida según la incertidumbre del equipo empleado, etc. Asimismo incluye una representación gráfica de resultados.
3. Implementación en MATLAB del método *Circle Point* para la obtención de los parámetros cinemáticos del robot, a partir de datos sintéticos o datos reales.
4. Validación del programa mediante datos simulados: de este modo es posible correlacionar los parámetros del robot empleados para generar los datos sintéticos con los parámetros obtenidos mediante el método implementado a partir de dichos puntos. Así mismo, se analiza la sensibilidad de los parámetros calculados a diversos parámetros del proceso de medición, como incertidumbre del equipo de medida, ángulo recorrido por cada articulación, sentido de giro o excentricidad en las articulaciones, de modo que sea posible establecer numéricamente la incertidumbre asociada a la identificación de cada parámetro.

Identificación de Parámetros cinemáticos de un brazo robot mediante el método "Circle-Point". Simulación y validación experimental

2 Índice

Tabla de contenido

3 Lista de figuras y tablas

4 Introducción

Existen numerosos métodos propuestos por diversos investigadores para abordar el problema de calibración de robots. Este problema, entendido de forma general, persigue la obtención de los parámetros del modelo cinemático del robot a partir de sus valores de diseño, de modo que el error de posicionamiento del robot sea lo menor posible.

Una característica particular a la mayor parte de estos métodos es que abordan el problema desde un punto de vista matemático. Tras la captura de una serie de puntos medidos y nominales de la posición y orientación del robot, estos métodos obtienen un conjunto de parámetros que minimiza el error de posicionamiento en las posiciones capturadas, aunque no tiene en cuenta aspectos físicos del robot, de modo que el conjunto final de parámetros identificados no tiene por qué corresponderse con los parámetros físicos reales del brazo, aunque cumplen la función para la que han sido identificados. De esta forma, mientras un robot trabaje en la zona para la que se han capturado puntos para el procedimiento de identificación el error de posicionamiento se mantendrá presumiblemente por debajo de los errores máximos obtenidos en los puntos capturados, aunque el hecho de la no correspondencia con los parámetros reales, hace que este valor de error no sea extrapolable a zonas de trabajo fuera de los puntos capturados para optimización. Una de las aproximaciones más "físicas" que existen para identificar los parámetros del modelo es el método del *Circle Point*, en el que se determinan los sistemas de referencia ligados a las articulaciones a partir de mediciones realizadas en diversos tramos del robot girando alternativamente una articulación cada vez.

En este trabajo se aborda la implementación del método para identificar los parámetros de un robot KUKA KR5 sixx, y su posterior validación experimental. Las tareas que se han previsto para llevar a cabo este trabajo son las que han definido en la sección anterior (véase el apartado 1. Resumen del Proyecto).

5 Identificación de los parámetros cinemáticos de un brazo robot mediante el método de *Circle Point*, simulación y validación experimental

5.1 Modelo cinemático del robot basado en Denavit-Hartenberg

Para la obtención de un modelo cinemático del robot inicial se ha elegido el modelo Denavit-Hartenberg (1). Inicialmente desarrollado para representar ejes de máquinas, fue después extendido por Paul (2) para su aplicación en robots. El método D-H representa un conjunto de uniones mediante estructuras ubicadas en cada articulación, y representando la transformación de la articulación mediantes 4 parámetros, a saber d_j, θ_j, a_{jk}, α_{jk}. Un gran número de documentación ha usado este modelo como base para la calibración de robots (3).

Sin embargo, el modelo de Denavit-Hartenberg conduce a singularidades cuando dos articulaciones consecutivas son paralelas o casi paralelas (4). La singularidad se produce debido al hecho de que pequeñas variaciones entre el modelo nominal y real provocan variaciones muy grandes en los parámetros D-H obtenidos. Esto produce problemas relativos a estabilidad numérica cuando se está ejecutando la implementación de la identificación.

Para resolver este problema se han realizado diversas aproximaciones (5). El modelo planteado por Hayati y Mirmirani (6) modifica el modelo de Denavit-Hartenberg, reemplazando un parámetro de distancia con un parámetro de ángulo, aunque este modelo así definido sólo es efectivo para articulaciones paralelas o casi paralelas, pero no para una representación genérica. El modelo realmente usado es un modelo híbrido, que usa el modelo Denavit-Hartenberg para articulaciones en general, y el modelo de representación Hayati-Mirmirani para articulaciones paralelas, el cual añadirá para el caso aquí expuesto el parámetro ángulo β2 en la articulación paralela o casi paralela (7)(8). La inclusión de este parámetro en la ecuación que da la transformación entre articulaciones adyacentes se puede ver en la Ecuación 8 (ver página 46).

El modelo cinemático se ha basado en un robot Kuka KR 5 sixx. Sus dimensiones y el propio robot se pueden ver en la Ilustración 1.

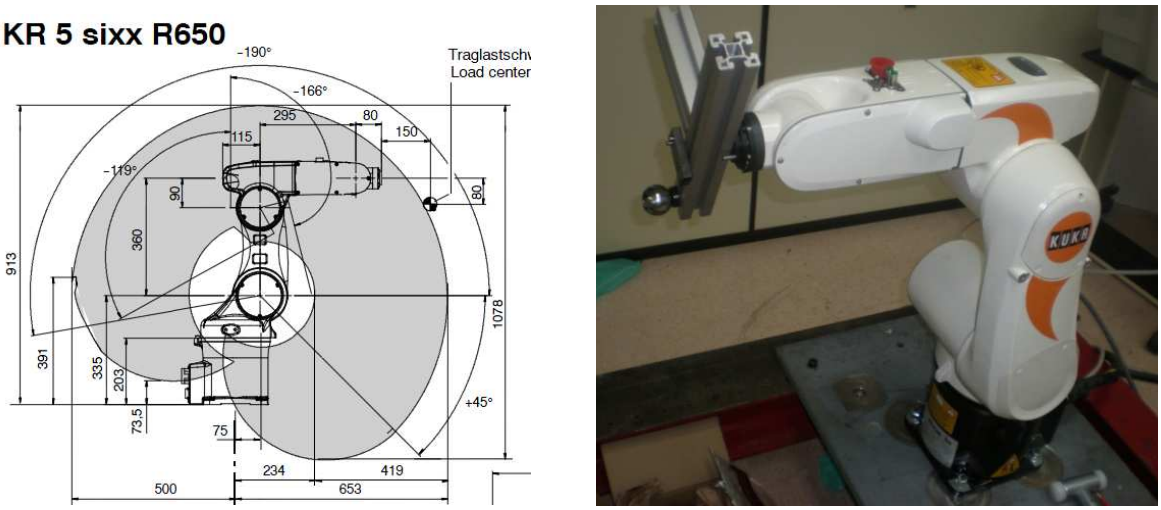


Ilustración 1 - Dimensiones y detalle de robot Kuka KR 5 sixx

Aplicando el modelo Denavit-Hartenberg, modificado por Hayati-Mirmirani, se obtiene la tabla de parámetros cinemáticos, que permite conocer la relación entre las articulaciones del robot.

Tabla 1 - Parámetros del modelo cinemático del robot Kuka

	d	θ	a	α	β
1	335	0	75	90	
2	0	0	270	0	0
3	0	90	90	90	
4	295	0	0	90	
5	0	180	0	90	
6	80	0	0	0	

5.2 Método de Circle-Point

La precisión de un brazo robot industrial típico puede ser mejorada mediante la calibración precisa o determinando los parámetros geométricos del sistema. La técnica de Círculo de Puntos es un modo de aproximación fácil y sencilla, la cual da una determinación precisa de los parámetros geométricos exactos del sistema de un brazo robot, en un sentido absoluto. Básicamente, cada una de las articulaciones de los ejes del sistema se localiza en el espacio para una única configuración. Cada par de líneas adyacentes en el espacio formado por los ejes es entonces analizado para determinar los parámetros para ese enlace.

El primer objetivo del método de medida mediante el análisis de Círculo de Puntos es ubicar cada uno de los ejes de articulación en el espacio. Hay que notar que la localización de una línea en el espacio requiere tanto la dirección de la línea como las coordenadas de un punto arbitrario en la línea. Una vez que cada línea es localizada en el espacio para la configuración final del robot, el análisis geométrico de las coordenadas de la línea da los parámetros calibrados geométricos requeridos. Las técnicas analíticas, así como el método completo *Circle-Point* (9), se muestran en profundidad en la sección Anexo I: Calibración de Parámetros Geométricos mediante el Análisis de Circle-Point (ver desde página 44). El procedimiento experimental básico consiste en los siguientes pasos:

1. El robot se coloca en la configuración inicial, que se graba (como un conjunto de ángulos de las articulaciones). Un único punto objetivo P se adhiere al brazo, después del primer eje (no en la base), a una distancia perpendicular del eje igual a un radio de valor definido R_1. Para el caso experimental, a realizar con un Láser Tracker, esto supondría colocar en el punto P la esfera de reflexión del haz del LT.
2. El robot entonces se mueve alrededor de la primera articulación únicamente, parándose en un número de localizaciones discretas. La ubicación del objetivo (que traza un camino aproximadamente circular) se mide en cada posición en cualquier sistema de coordenadas elegido convenientemente. Para el ensayo experimental, esto será en coordenadas Láser Tracker. Esto crea un círculo aproximado de puntos de datos, lo que le da el nombre a esta técnica de *Circle-Point*. Los valores de desplazamiento de la articulación para el primer eje se graban para cada posición. La Ilustración 26 ilustra los puntos en el círculo de la articulación 1, para todo un sistema en completa revolución. Nótese que todas las articulaciones excepto la primera permanecen fijas. Obviamente, cualquier robot calibrado mediante esta técnica deben ser capaces de un control de movimiento por articulación individual.

3. Los datos de puntos del círculo son entonces analizados para definir la ubicación del eje. Primero, los puntos del círculo se encajan en un plano que es el que mejor se aproxima, mediante una aproximación por mínimos cuadrados[1], que consigue definir la ecuación del plano. La normal del plano resultante define la dirección del eje de la articulación, definida como $\overrightarrow{S_1}$. Los puntos entonces se proyectan sobre el plano, en dirección perpendicular al mismo. Los puntos proyectados se encajan, otra vez usando la solución de mínimos cuadrados, en la ecuación de un círculo en el plano. El centro de este círculo, que se denota como $\overrightarrow{r_1}$, representa un punto en el eje o línea definido por el eje de la articulación.
4. El punto objetivo P se adhiere entonces al brazo en una posición después del segundo eje (es decir, en un punto afectado por el movimiento del eje 2), a una distancia perpendicular al eje aproximadamente igual al radio de valor R_2.
5. El robot se controla entonces para moverse alrededor del eje 2 solamente. Otra vez, el punto P traza un camino aproximadamente circular y se miden y graban los valores individuales de desplazamiento de las articulaciones. Nótese que la articulación 1 no se debe mover, ya que esto causaría el movimiento del eje 2. El objeto de este paso es localizar el eje 2 el cual por consiguiente permanece fijo. Los puntos del círculo son entonces analizados usando la misma técnica analítica que en el paso 3, que permite obtener la dirección $\overrightarrow{S_2}$ del eje y un punto $\overrightarrow{r_2}$ en el eje o línea. Los puntos del círculo para el eje 2 se muestran también en la figura.
6. Un proceso idéntico se debe repetir para cada eje en el sistema. Hay que prestar un cuidado extremo en evitar mover cualquier eje que esté ubicado en la cadena de enlaces cinemática antes del eje que se está moviendo en cada momento.

5.3 Implementación del método mediante MatLab

Para la ejecución del método, empezando por la parte que supone la realización de un simulador que emule la adquisición de puntos según el método de Circle-Point anteriormente descrito, se han desarrollado una serie de funciones en MatLab (10)(11), que implementan el algoritmo, tanto en su parte de simulación de adquisición de datos, como en la parte de cálculo de los vectores Sj y ajk de cada articulación, y posterior obtención de los parámetros del modelo del robot.

Primeramente se ha desarrollado el método *Genera_Puntos_Circle_Point*, el cual, considerando una serie de parámetros de entrada (como son posición inicial de los ejes, matriz de transformación del robot al equipo de medida, errores de giro en las articulaciones e incertidumbres en el equipo de medición a considerar, número de puntos a generar por articulación, ángulo a describir por cada articulación, etc.), entrega el conjunto de puntos por cada articulación obtenida. Es decir, este es el simulador del método *Circle-Point*.

A continuación, se tiene la función *Coordenadas_Plucker_Ejes*. Esta función, a partir de los puntos obtenidos del robot mediante la técnica del Circle-Point, o bien a partir de los puntos entregados por el simulador, calcula y devuelve las coordenadas Plücker (12), es decir los conjuntos (S_j, S_{0j}) de cada articulación[2], que son los vectores que identifican unívocamente la línea que define la articulación y un punto de origen arbitrario.

[1] Ver función ls3dcircle.m, que usa una aproximación por mínimos cuadrados, y algoritmo Gauss-Newton.
[2] Para mayor aclaración, ver apartado 9.5 Análisis de Línea adyacente, en página 47.

Una vez se tienen las coordenadas Plücker, es en la función *Calcula_Parametros_KM* donde se aplica exhaustivamente las indicaciones y cálculos algebraicos y vectoriales(13)(14) que marca el método Circle-Point, para conseguir calcular los parámetros del modelo del robot. Tanto en la explicación detallada de la función con su pseudo-código que hace referencia a las ecuaciones que se aplican (apartado 5.8, página 23), como en el apartado 9 - Anexo I: Calibración de Parámetros Geométricos mediante el Análisis de Circle-Point, se puede ver una explicación en detalle del método.

Por otra parte, mediante la función *Montecarlo_Circle_Point* se realizan la *n* iteraciones numéricas de cálculo de parámetros para este método, que llama a las tres funciones anteriormente citadas. Con ésta se obtiene un conjunto de parámetros, que se ha conseguido aplicando Montecarlo, según la incertidumbre del equipo de medición y errores de giro en cada una de las articulaciones.

Por último, la función *Simulación_MCP* se ha implementado para definir los parámetros de entrada, así como cada una de las simulaciones que se han determinado que se van a ejecutar.

5.4 Mejoras de la técnica del Circle-Point

La técnica del Circle-Point, en esta aplicación, se ha optimizado de las siguientes formas:

- Cálculo del parámetro de la primera articulación - d_1: El método Circle-Point parte de un sistema de referencia arbitrario. Sin embargo, a efectos de conseguir un conjunto de parámetros coincidentes con la realidad se han incluido modificaciones en el método, que permiten calcular estos datos de forma precisa. Esto se consigue de la siguiente forma:
  - Primeramente, se tiene que conseguir la matriz de cambio del robot al equipo de medición. Esto se realiza llevando el robot a varios puntos, en áreas diferentes del volumen a cubrir por el robot, y obteniendo dichos puntos en coordenadas del robot y en coordenadas del equipo de medición. Con estos puntos, se consigue la matriz de cambio buscada[3].
  - Después, la función *Coordenadas_Plucker_Ejes* devuelve el valor SR0, que es el sistema de referencia origen del robot, en coordenadas del equipo de medición.
  - Con este valor, y comparando con el valor (0,0,0) en coordenadas robot, se calcula el parámetro d1, independientemente de la posición en la que se coloque el reflector.
- Cálculo del parámetro de la primera articulación – θ_1: Al igual que en el caso anterior, el método Circle-Point no define un sistema de referencia a partir del cual establecer los cálculos, por lo que no define como calcular este parámetro θ_1. Además, el cálculo de θ_j se realiza según la Ecuación 6 y Ecuación 7, por lo que para θ_1 no es posible realizarlo, al no disponerse del valor $\overrightarrow{a_{01}}$. Sin embargo, en efecto sí que se ha obtenido este vector, ya que se ha tomado como la componente del eje X del parámetro SR0 que devuelve la función *Coordenadas_Plucker_Ejes*, con lo que una vez se tiene este vector $\overrightarrow{a_{01}}$, ya que se puede calcular de forma exacta el valor del parámetro θ_1.
- Cálculo del parámetro del modelo Hayati – β_2: Este parámetro se calcula como se explica en el apartado 9.6.3 Solución 3: Modelo cinemático de parámetro adicional[4]. Sin embargo, el cálculo

[3] Ver apartado 12 Anexo IV: Cálculo de la Matriz de Transferencia del sistema de referencia del robot al Láser Tracker, en página 74.
[4] Ver página 64.

se basa en el cálculo del coseno de β_{jk}[5] , el cual sólo permite obtener ángulos en el intervalo [0, 180]. El cálculo del seno de β_{jk} no se puede realizar, al desconocerse el vector $\overrightarrow{b_{jk}}$[6]. Para evitar esta dificultad, se calcula el seno de β_{jk} de la siguiente forma:

$$\sin\beta_{jk} = \cos\left(\overrightarrow{S_k}\ sobre\ \overrightarrow{a_{jk}}\right) = \cos Sk_{ajk} = \overrightarrow{S_k}\cdot\overrightarrow{a_{jk}} \qquad \text{Ecuación 1}$$

Que en la función se denomina *cos_skajk*. Es decir, la proyección de $\overrightarrow{S_k}$ sobre $\overrightarrow{a_{jk}}$ es el valor seno buscado. Por tanto, a partir de este valor, se calcula fácilmente β_{jk}:

$$\beta_{jk} = atan2(\cos Sk_{ajk}, \cos\beta_{jk}) \qquad \text{Ecuación 2}$$

Con lo que se obtiene un ángulo en el intervalo [-180, +180].

- Cálculo de los ángulos α, β, θ: Los ángulos se han calculado con la función *atan2*, que permite devolver valores en un intervalo entre [-180,+180]. Sin embargo, para valores de ángulos muy próximos a ±180° este resultado genera unos resultados con una gran variabilidad, al obtener, con muy poca variación de los errores generados por Montecarlo, valores de +180,001° ó -180,001°. Por esto, también se introduce una verificación en el algoritmo, de forma que para ángulos cercanos a ±180°, el intervalo devuelto por *atan2* se convierte a un intervalo [0, +360].
- Verificación del sentido de los vectores Sj: En las funciones implementadas se ha asegurado que las direcciones de los vectores Sj obtenidos son coincidentes con las direcciones definidas por el modelo DH-Hayati nominal implementado en el robot. En caso de que no coincida, se ajusta para que coincida con las direcciones de dicho modelo.

5.5 Resumen de funciones MatLab

Para realizar tanto la identificación de parámetros cinemáticos del brazo robot, como la simulación de la toma de puntos, se han usado diversas funciones de MatLab. Un compendio de estas funciones con una breve explicación de cada una se detalla a continuación:

- Genera_puntos_circle_point: Genera los puntos que simulan la toma de puntos por cada articulación según el método Circle Point.
- Coordenadas_Plucker_Ejes: Esta función ccalcula las coordenadas *Plucker* correspondientes a los 6 ejes del robot con los datos de entrada obtenidos de una Generación de Puntos completa del robot según el método del *Circle-Point*, o a partir de los datos reales.
- Calcula_Parametros_KM: Esta función realiza el cálculo de los parámetros reales del modelo del robot, a partir principalmente de la información entregada con las coordenadas Plücker para cada articulación, que definen los ejes de cada una de ellas. Esta función, aplicando la metodología definida según el método del Circle Point, identifica los parámetros del modelo, y devuelve estos parámetros en la variable de salida *P*.
- fRobotMKR5sixxDHHpar: Devuelve las 6 matrices de cambio de coordenadas de una articulación a otra, según el modelo Denavit-Hartenberg, modificado con el 5º parámetro por Hayati-Mirmirani, más la matriz de cambio general (M0A6).

[5] Ver Ecuación 80 y Ecuación 82.
[6] Ver Ecuación 81.

- Pinta_SR: Pinta un Sistema de Referencia XYZ, de la articulación i, en el punto $r_i(i)$, con una matriz de giro y traslación M desde el origen al punto.

Además, se han utilizado otras funciones de ayuda en MatLab:

- *arrow3D*: Permite representar una flecha 3D, desde una posición XYZ, con un delta dX,dY,dZ. Esta función también usa la siguiente función *rotatePoints*.
- *lsplane*: Devuelve el plano que mejor ajusta a la matriz de puntos entregada. Para una sentencia del tipo:

  ```
  [x0, Eje, dist, normd] = lsplane(PtosEje(:,1:3));
  ```

  Se pasan los puntos en la variable PtosEje (matriz de n filas, por 3 columnas XYZ), y devuelve x0, como centroide de los puntos (matriz 1x3 – datos XYZ), Eje, como valor de los cosenos directores que definen la dirección del plano (matriz 1x3).
- *ls3dcircle*: Devuelve el círculo 3D que mejor ajusta a la matriz de puntos entregada. Para una sentencia del tipo:

  ```
  [x0n, an, rn, dc, e, f, sigmah, conv, Vx0n, Van, urn, GNlog, a, R0, R]
  = ls3dcircle(PtosEje(:,1:3), x0, Eje, r0(i), 1e-6, 1e-6);
  ```

  Se pasa la matriz de puntos en la variable PtosEje (matriz de n filas, por columnas XYZ), x0 es el centro del círculo estimado (matriz 3x1), Eje es la normal al plano estimada (matriz 3x1), y r0(i) es el radio estimado. Esta función devuelve en x0n el centro del círculo estimado (matriz 3x1), en an la dirección normal al plano (matriz 3x1), en rn el radio del círculo estimado (matriz 3x1), en dc el vector de distancias de los puntos al círculo (matriz mx1), en e el vector de distancias de los puntos al plano que contiene el círculo (matriz mx1), en f el vector de distancias de los puntos al cilindro que contiene el círculo. Esta función también se sirve de las siguientes funciones para realizar los cálculos necesarios: nlss11, gncc2, fg3dcircle, rot3z, gr, fgrrot3, frrot3, drrot3.
- subplot_title: Muestra un título interno en una figura subplot.

5.6 Función Genera_Puntos_Circle_Point: Simulación de toma de puntos siguiendo el método *Circle Point*

Esta función genera los puntos que simulan la toma de puntos, para cada articulación, según el método Circle-Point.

Esta función tiene la siguiente línea de llamada:

- Función: `[Ptos_Circle_Point, Ptos_Original, Randomnoise] = Genera_puntos_circle_point (Initpos, SRLT, Preflector, Angcir, Nptos, Ruidomed, Err, Param5, ShowFigure)`
- Parámetros de entrada:
  0. Initpos: Posición inicial del robot a partir de la cual se generarán los círculos. Vector 1x6 con los ángulos de las articulaciones.

1. SRLT: Matriz de transformación que pasa puntos del SR (Sistema de Referencia) global del robot al SR de medición del LT (Láser Tracker). Los puntos generados estarán expresados en el SR del LT.
2. Preflector: Matriz 6x3 con las coordenadas X Y Z de la posición del reflector del LT (en posición inicial) expresadas en el SR de la articulación que gira (fijo) para todas las articulaciones:[X1 Y1 Z1; X2 Y2 Z2;...;X6 Y6 Z6]
3. Angcir: Vector 1x6 con el ángulo de giro que cubrirá cada articulación en el ensayo.

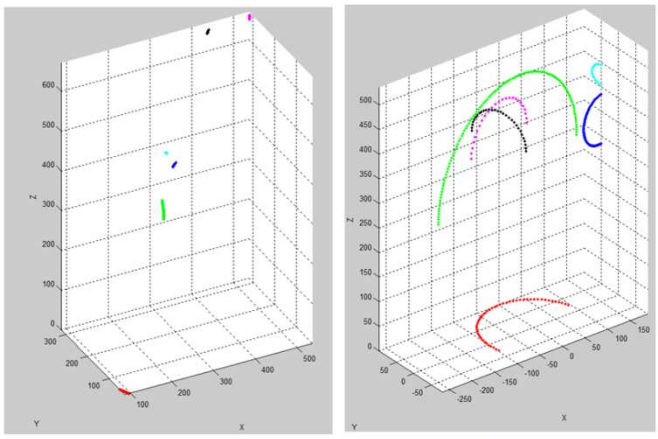


Ilustración 2 - Gráfica puntos según ángulo

4. Nptos: Vector 1x6 con el número de puntos que se generarán en cada círculo.

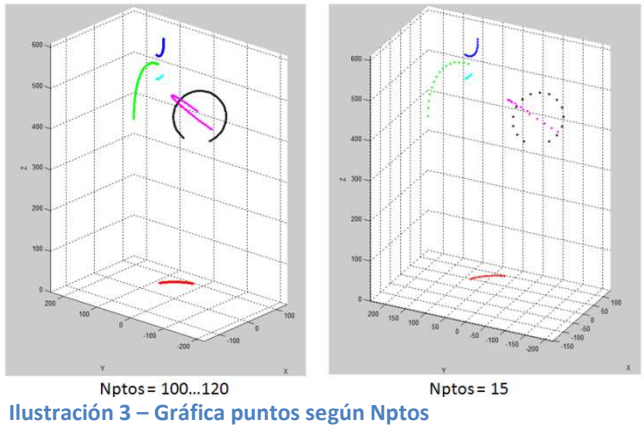


Ilustración 3 – Gráfica puntos según Nptos

5. Ruidomed: Desviación estándar de la distribución normal con la que se generará aleatoriamente ruido normal **debido al instrumento de medida**, centrado en el valor nominal en cada coordenada. Consideraremos con carácter general un valor de desviación tal que 2σ sea aproximadamente el 95% del error máximo del instrumento de medida (LT, Estación total, Indoor GPS, etc.). Cero si no se desea ruido. Este parámetro viene expresado en milímetros. Vector 1x2, que significa: Ruido = A + B*L, siendo A el primer valor, en micrómetros, y B en µm/m.
6. Err: Matriz 6x20 con los errores de giro a inducir en cada una de las articulaciones. Cada fila tiene estas componentes: [dx dy dz Ex Ey T1 T2 T3 T4 T5 fi1 fi2 fi3 fi4 fi5 R1 R2 R3 R4 R5]. Son los términos correspondientes al modelado de errores en ejes giratorios (matriz Slocum). En ellos, dx, dy, dz son los deltas en cada uno de los ejes para la traslación respecto al sistema de referencia fijo y Ex, Ey son giros entorno a X e Y del sistema fijo. Ez no se considera porque la rotación es entorno a Z. Todos los errores se van a generar dependientes del valor del ángulo de giro de la articulación en cada momento, por lo que lo que se introducirá en esta matriz serán las amplitudes máximas

de estos errores. Su valor final se modelará como: $Error = A \cdot \sin\left(\left({}^{2\pi}/_{T}\right) \cdot \theta + \varphi\right)$, por tanto, dependiente del valor de giro en cada momento, siendo:

- A: el valor introducido en los términos de error (amplitud). En la matriz milímetros para desplazamientos y grados para giros. Si no se desean errores los 5 primeros términos igual a 0.
- T: Periodo de la oscilación, en radianes (normalmente consideraremos 2π). En la matriz se especifica uno para cada error.
- θ (thita): Será el valor de giro de la articulación en cada momento
- φ (fi): fase (en radianes), una para cada error en la matriz. (normalmente consideraremos 0).

Modelando este error en función de thita, en cada posición de giro de una articulación, su SR estará en una posición distinta para cada thita. Se obtendrá la matriz de este sistema al SR nominal de la articulación para cada ángulo. Los parámetros R1 a R5 son desviaciones estándar de la distribución normal con la que se generará aleatoriamente ruido normal al valor de cada error. Se pondrá 0 si no se desea generar ruido.

7. Param5: Vector con los parámetros DH del robot
8. ShowFigure: Variable lógica, que indica si se deben mostrar los gráficos o no.

- Parámetros de salida:
  - Ptos_Circle_Point: Contiene los puntos generados contemplando defectos y ruido. Vector de 9 elementos por fila, de los que 3 son XYZ del punto, y los otros 6 la posición de los ángulos de cada articulación.
  - Ptos_Original: Contiene la trayectoria nominal sin considerar defectos ni ruido. Estructura como la del parámetro anterior.
  - Randomnoise: Vector 6x3xn con valores aleatorios sumados a cada coordenada dentro del intervalo especificado en Ruidomed.

El pseudo-código de esta función es el siguiente:

Inicializa variables
Para cada articulación i desde 1 a 6
Calcula incremento del ángulo para generar el círculo
Para cada punto j espaciado un ángulo IncAng
Calcula los errores, según la matriz de errores Err (parámetro entrada)
Calcula la matriz de giro GZ para ese punto incluyendo los errores → Ver Ecuación 93
Calcula el punto del círculo con errores
Calcula la matriz de cambio M desde el origen del robot a la articulación actual
Cambia el punto según matriz de cambio M y según matriz de cambio SRLT
Añade ruido aleatorio debido al instrumento de medida
Guarda ruido en Randomnoise
Guarda el punto con/sin ruido debido al instrumento de medida
Fin Para

Guarda puntos del círculo de la articulación actual
Guarda la matriz de error GZi de la articulación actual i, según última/actual posición
Fin Para
Si ShowFigure entonces muestra la figura con los puntos – círculos

5.7 Función Coordenadas_Plucker_Ejes: Calcula las coordenadas Plücker

Esta función calcula las coordenadas *Plücker* correspondientes a los 6 ejes del robot, bien sea con los datos de entrada obtenidos de una Generación de Puntos completa del robot según el método del *Circle Point*, o bien sea con los datos de las lecturas tomadas con un instrumento de medición (por ejemplo, un Láser Tracker) de los puntos de cada articulación del robot. En las simulaciones se utilizará esta función, que permite detectar automáticamente si las normales obtenidas están o no en la misma dirección que los ejes Z del robot. Si no lo están se cambia su sentido automáticamente. Con datos reales, se necesita suministrar a esta función la matriz de transformación que pasa del SR_{LT} (Sistema de Referencia del Láser Tracker) al SR_R (Sistema de Referencia del Robot).

Esta función tiene la siguiente línea de llamada:

- Función: `[Plucker, P_Circulos, SR0, Distcirculo, Distplano, Distcilindro, sigmaPlano, sigmaCirculo, sigmaCilindro, Angfin]= Coordenadas_Plucker_Ejes(Ptos_Circle_Point, Nptos, r0, Param5, ShPoints, ShBestfitCircles, ShNormals, ShrefS, ShEx, SRLT)`

- Parámetros de entrada:
  1. Ptos_Circle_Point: Matriz 6x1000x9 con los puntos de los círculos medidos en SRLT.
  2. Nptos: Vector 1x6 con el número de puntos de cada círculo.
  3. r0: Vector 1x6 con los estimadores de los radios de los círculos descritos por el reflector en cada articulación.
  4. Param5: Parámetros del robot, según el método Denavit-Hartenberg, con la mejora del parámetro adicional por el método Hayati-Mirmirani.
  5. ShPoints, ShBestfitCircles, ShNormals, ShrefS, ShEx: Indican si se representarán en el resultado gráfico los puntos, círculos aproximados, normales a los planos, sistemas de referencia del robot y gráficas de excentricidad. Son parámetros booleanos, que pueden ser: *true* ó *false*.
  6. SRLT: Matriz de transformación que pasa puntos del SR global del robot al SR de medición del LT, utilizada en la generación de puntos o en la lectura real de los mismos. Los puntos estarán expresados en el SR del LT.
- Parámetros de salida:
  1. Plucker: Matriz 6x6 que contiene por filas las coordenadas Plucker de los ejes del robot obtenidas mediante el método Circle Point.
  2. P_Circulos: Matriz 6x7 que contiene por filas Centro, radio y normal del círculo ajustado en cada eje.
  3. SR0: Devuelve la matriz de cambio del Sistema de Referencia 0, es decir, del origen de coordenadas del robot. Contiene los vectores de cambio del eje X,Y,Z, por columnas. (eje X=columna 1, etc). Matriz 4x4.
  4. Distcirculo: Vector 6x1000 que contiene por filas las distancias de los puntos al círculo de ajuste en cada eje.
  5. Distplano: Vector 6x1000 que contiene por filas las distancias de los puntos al plano de ajuste en cada eje.
  6. Distcilindro: Vector 6x1000 que contiene por filas las distancias de los puntos al cilindro que contiene al círculo de ajuste en cada eje.
  7. SigmaPlano: Vector 1x6 con las desviaciones estándar de la distribución de distancias al plano para los 6 ejes.
  8. SigmaCirculo: Vector 1x6 con las desviaciones estándar de la distribución de distancias al círculo para los 6 ejes.

9. SigmaCilindro: Vector 1x6 con las desviaciones estándar de la distribución de distancias al cilindro para los 6 ejes.

El pseudo-código de esta función es el siguiente:

Inicializa variables
Para cada articulación i desde 1 a 6
Obtiene el plano y círculo que mejor se ajusta a los puntos
Guarda los valores obtenidos (Si, ri, rni, P_Circulos, distancias, sigmas)
Fin para
Comprueba si normal eje Z calculado (Si) coincide con normal SRLT
Si normal es diferente → entonces cambia sentido a Si
Guarda coordenadas Plücker y S0i

Si tiene que mostrar alguna gráfica
Para cada articulación i desde 1 a 6
Si ShPoints → muestra los puntos del círculo y el número de articulación
Proyecta el punto 1 del círculo en el plano XY de la articulación
Para cada punto m desde 1 hasta Nptos(i)
Obtiene PProy = proyección del punto sobre el plano normal a Si
Calcula distancia axial (excentricidad) del punto al plano – punto proyectado
Calcula distancia radial (excentricidad) del punto proyectado a la circunferencia
Fin Para
Si ShEx → Muestra gráfico excentricidad, y las excentricidades radial y axial calculadas
Si (ShBestfitCircles OR ShrefS) → entonces
Calcula los puntos del círculo ajustado XYZ con el radio obtenido
Calcula la matriz de giro MG del SR_articulación al SRLT
Gira los puntos XYZ que forman el círculo con esta matriz MG
Si ShBestfitCircles → representa el círculo
Si ShNormals → Representa las normales con dirección
Si (ShrefS y es articulación 1) → Pinta los sistemas de referencia, guarda SR0
Fin para
Y si no tiene que mostrar ninguna gráfica
Calcula SR0
Fin Si
Si ShrefS → Muestra la imagen de los ejes del robot

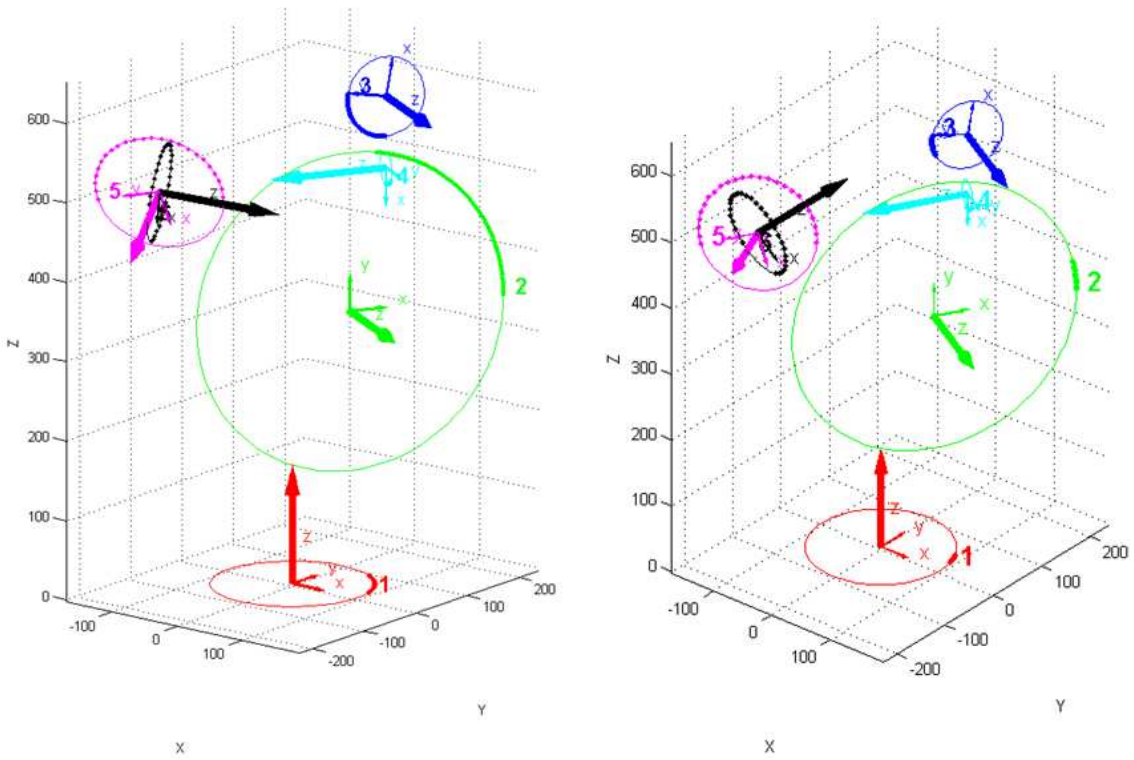


Ilustración 4 - Representación ejes y puntos según Coordenadas Plucker

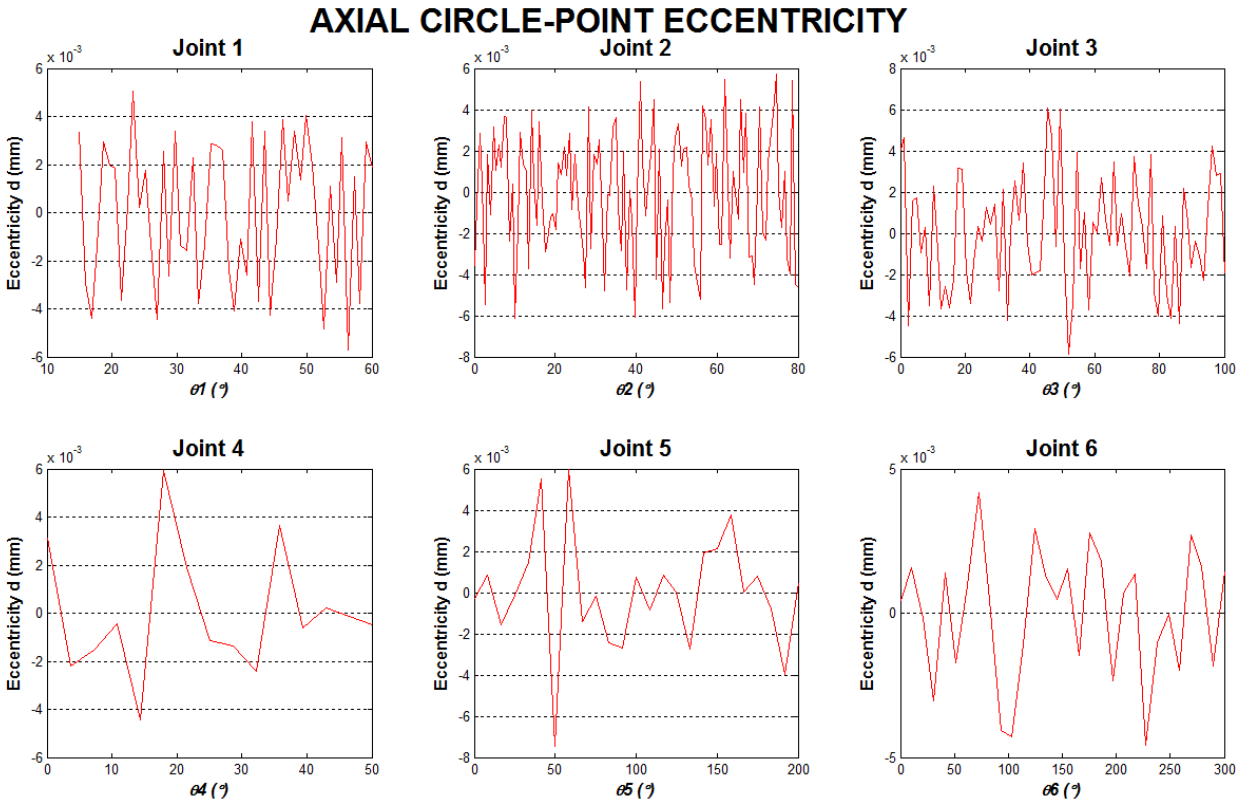


Ilustración 5 - Excentricidad Axial

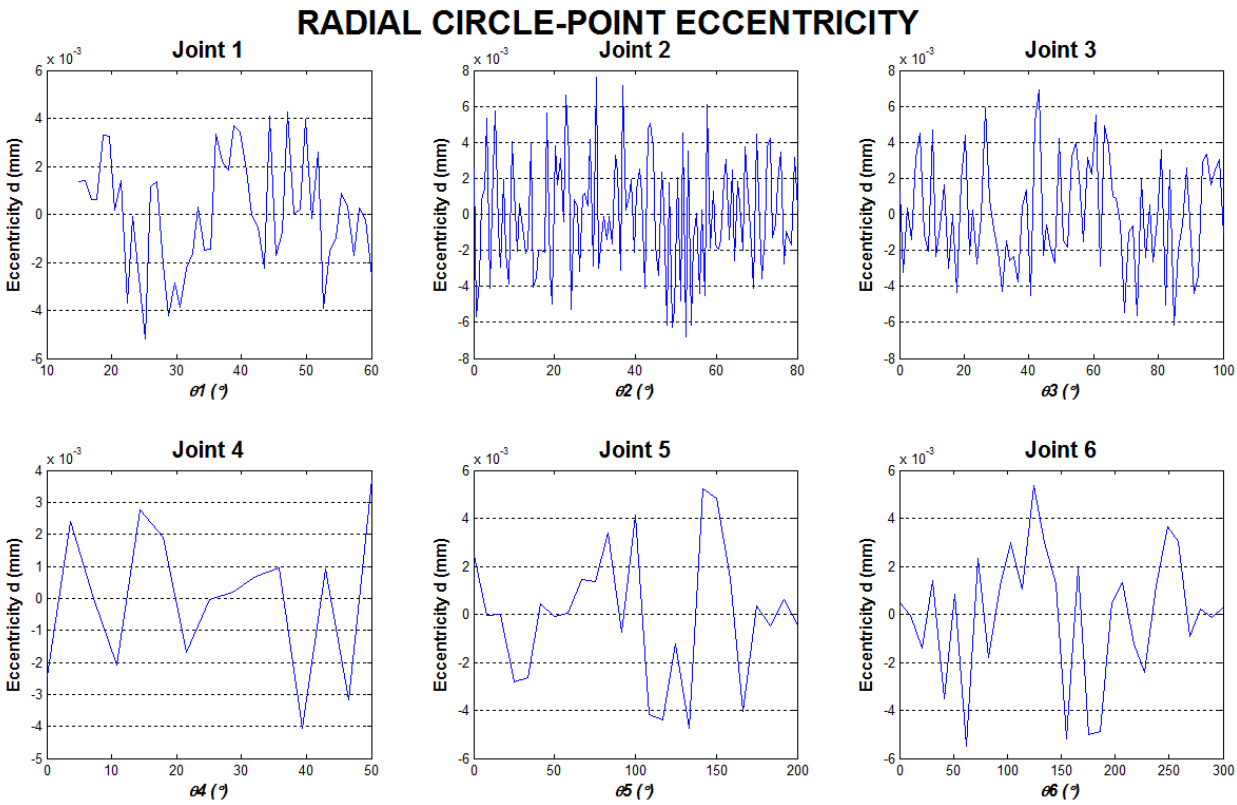


Ilustración 6 - Excentricidad Radial

5.8 Función Calcula_Parámetros_KM: Cálculo de los parámetros DH a partir de los datos tomados con método Circle Point

Esta función realiza el cálculo de los parámetros reales del modelo del robot, a partir principalmente de la información entregada con las coordenadas Plücker para cada articulación, que definen los ejes de cada una de ellas. Esta función, aplicando la metodología definida según el método del Circle-Point, identifica los parámetros del modelo, y devuelve estos parámetros en la variable de salida *P*.

- Función: `function P= Calcula_Parametros_KM (Puckler, P_Circulos, Param5, Angfin, CheckX, SRLT, SR0, ShowFigure)`
- Parámetros de entrada:
    1. Plucker: Matriz 6x6 que contiene por filas las coordenadas Plücker (Sj, S0j) de los ejes del robot, obtenidas según el método *Circle Point*.
    2. P_Circulos: Resultado del ajuste de los círculos de puntos, tal y como lo da la función *Coordenadas_Plucker_Ejes*.
    3. Param5: Parámetros nominales del modelo del robot, según modelo Denavit-Hartenberg, modificado según Hayati-Mirmirani. Matriz 1x30, con el formato [d1..d6 tita1..6 a1..6 alfa1..6 beta1..6], para cada articulación.
    4. Angfin: Ángulo de giro que indica el robot en la última posición de captura de datos. Vector 1x6.
    5. CheckX: Comprueba que la dirección de los vectores ajk coincide con el eje X nominal de la siguiente articulación. Variable booleana (*true* o *false*).

6. SRLT: Matriz de transformación que pasa puntos del SR global del robot al SR de medición del LT utilizada en la generación de puntos. Los puntos generados estarán expresados en el SR del LT.
7. SR0: Matriz de cambio del Sistema de Referencia 0, es decir, del origen de coordenadas del robot. Contiene los vectores de cambio del eje X,Y,Z, por columnas. (eje X=columna 1, etc). Matriz 4x4.
8. ShowFigure: Variable lógica, que indica si se deben mostrar los gráficos o no.

- Parámetros de salida:
  1. P: Parámetros reales del modelo del robot, en el formato `P=[dj titaj ajk alfajk beta_jk]`. Matriz 1 x 30.

El pseudo-código de esta función es el siguiente:

Para cada articulación j desde 1 a 5
Toma Sj, S0j, Sk, S0k de las coordenadas Plücker, cos(alfa_jk), calcula matriz M, Mant
cos(alfajk)=Sj·Sk

Si cos(alfajk)=+1 ó -1
//EJES PARALELOS
*a_jk=Sj·(cos(alfa_jk)*S0k – S0j)* → *Ver Ecuación 85*
alfa_jk=arc cos(Sj·Sk) → *Ver Ecuación 86*
Fin si
Si cos(alfajk)≈-1 ó +1
//EJES CASI PARALELOS: Se soluciona según Solución 3 del método Circle Point.
Obtener puntos de intersección Pn y Pp
*Pn = Pp_prev + dj*Sj* → *Ver* Ecuación 63
*Pp = (Sj x S0k + (Pn · Sj)*Sk) / (Sj · Sk)* → *Ver* Ecuación 67
*Skp=Sk – (Sk·ajk)*ajk* → *Ver* Ecuación 79
$\cos\beta_{jk} = \sqrt{\left[1 - (\overrightarrow{S_k} \cdot \overrightarrow{a_{jk}^{nom}})^2\right]}$ → *Ver* Ecuación 82
Cos(alfa_p_jk)=(Skp / |Skp|) · Sj → *Ver* Ecuación 83
Sen(alfa_p_jk)=((Sj x Skp)/ |Skp|)· ajk → *Ver* Ecuación 84
alfa_jk= arc tan(sen(alfa_p_jk)/cos(alfa_p_jk))
y si no
//EJES NO PARALELOS – Hay que calcular el momento mutuo
MM=Sj · S0k + Sk · S0j → *Ver* Ecuación 26
Si MM=0
// MM=0 → Las líneas se cortan => Pn=Pp
A_jk=0
A_jk_v=(Sj x Sk)/ |Sj x Sk| → *Ver*Ecuación 30
Pn=Pp=(S0j x S0k) / (Sk · S0j) → *Ver* Ecuación 48
Sen(alfa_jk)=| Sj x Sk | → *Ver* Ecuación 17
alfa_jk= arc tan(sen(alfa_p_jk)/cos(alfa_p_jk))
y si no
// MM<>0 → Las líneas son oblicuas
*A_jk_v=-signo(MM) * ((Sj x Sk)/ |Sj x Sk|)* → *Ver* Ecuación 31
Si CheckX=1
//Comprueba que la dirección de $\overrightarrow{a_{jk}}$
// coincide con la dirección del eje X nominal del modelo
Calcula matrices giro del modelo
Cos(ang)=Mx · $\overrightarrow{a_{jk}}$
Ajk_signo=positivo
Si (cos(ang)<0) → entonces

Cambia sentido ajk_v, cambia sentido MM, cambia ajk_signo
Fin si
Fin si
*Pn= (((ajk x Sk) x S0j) – (S0k · ajk)*Sj) / ((ajk x Sk) · Sj)* → *Ver* Ecuación 39
*Pp= (((ajk x Sj) x S0k) – (S0j · ajk)*Sk) / ((ajk x Sj) · Sk)* → *Ver* Ecuación 40
Ajk=|Pp – Pn| ajk_signo* → *Ver* Ecuación 27
Sen(alfa_jk)= ajk · (Sj x Sk) → *Ver* Ecuación 17Ecuación 17
alfa_jk= arc tan(sen(alfa_p_jk)/cos(alfa_p_jk))
Fin si
Fin si

Si j=1
// Para la primera articulación el cálculo es algo distinto
dj(1)=Distancia euclídea desde Pn(1) hasta Origen_robot
Si ShowFigure → muestra los vectores ajk_{prev}, ajk_{actual}, *Sj,* (ajk_{prev} *x* ajk_{act})
Calcula thita(1) con respecto al eje X (1 0 0) del SR0
Y si no
// Para el resto de articulaciones
dj(j)=Distancia euclídea desde Pn(j) hasta Pp(j-1) → *Ver* Ecuación 91
Si ShowFigure → muestra los vectores ajk_{prev}, ajk_{actual}, *Sj,* (ajk_{prev} *x* ajk_{act})
Calcula thita(j) → *Ver* Ecuación 6 *y* Ecuación 7
Fin si
Ajusta el ángulo ajk, para que coincida con el ángulo del eje Xi – matriz M
Fin para (cada articulación j desde 1 a 5)
Ajusta ángulo thita según los ángulos finales Angfin que se tienen
Asigna parámetros al último sistema de referencia (j=6)
Devuelve dato de salida P

La siguiente gráfica muestra las direcciones de los vectores a_{jk} y S_j obtenidos.

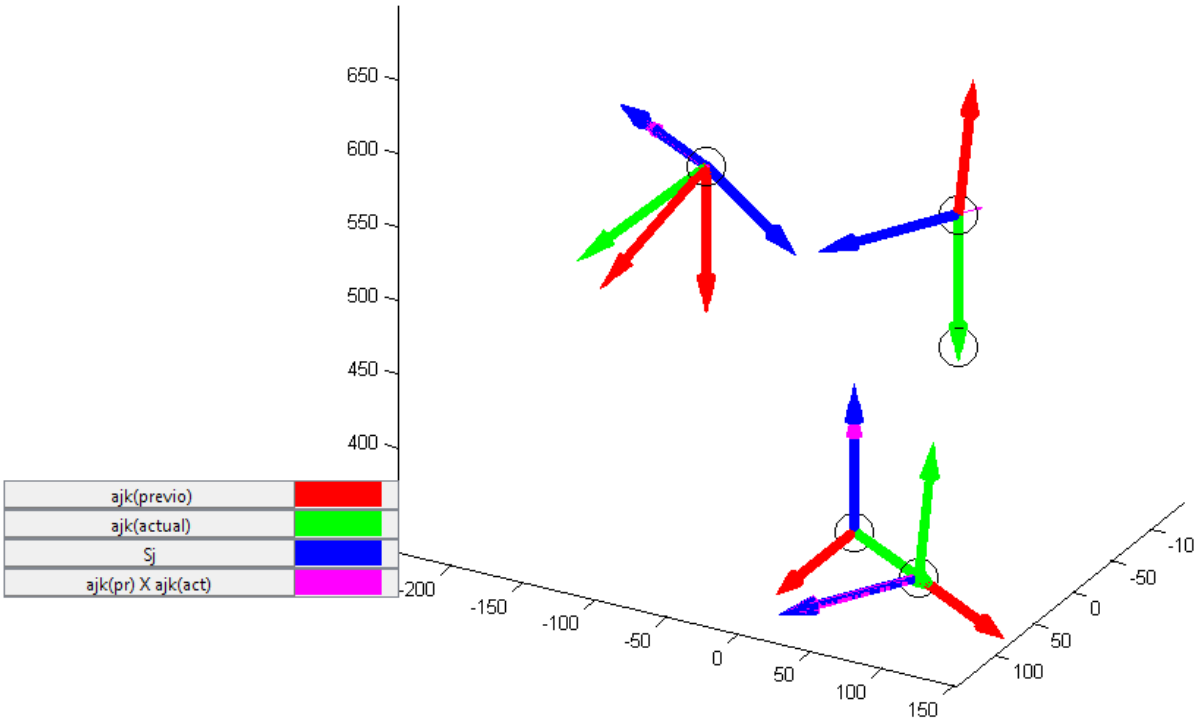


Ilustración 7 - Vectores ajk, ajk_prev, Sj, producto vectorial ajk x ajk_prev

5.9 Función: Montecarlo Circle-Point

Esta función realiza ***n*** simulaciones numéricas del cálculo de los parámetros, mediante el método de *Circle-Point*, llamando a las funciones anteriormente vistas. En la salida obtiene la media y desviación estándar de los valores de cada parámetro, obteniendo por tanto el valor medio y la incertidumbre en la obtención de cada parámetro cinemático del modelo.

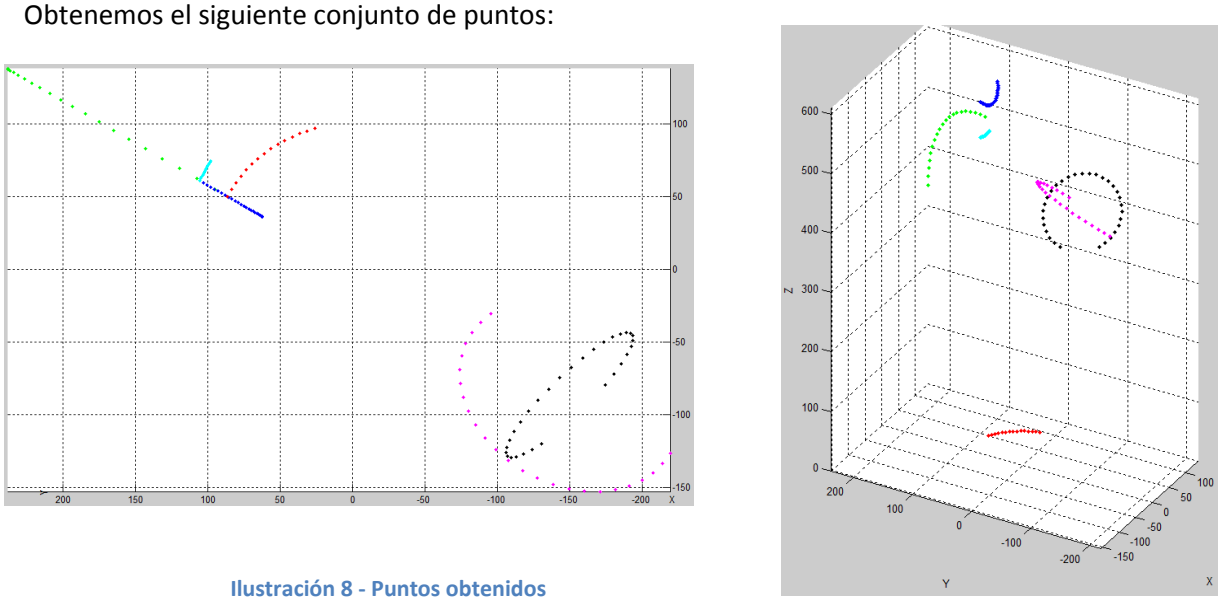
- Función: `function [Pmed,uPmed,I_inf,I_sup, maxmed, minmed, DifP, CP_Param, timefunc] = Montecarlo_Circle_Point(n, Measunc, Initpos, SRLT, Preflector, Angcir, Nptos, Err, Param5, r0, IntCov)`

- Parámetros de entrada:
  1. n: Número de iteraciones.
  2. MeasUnc: Vector 1x2 con la Incertidumbre de medida nominal del equipo con el que se han capturado los puntos. Por ejemplo: para un Láser Tracker de incertidumbre 5µm + 0.3µm/m, este parámetro sería: Measunc=[5 , 0.3].
  3. IntCov: Porcentaje de datos para el intervalo de cobertura. Por ejemplo, 95%.
     El resto de parámetros se han explicado en las anteriores funciones.

- Parámetros de salida:
  1. Pmed: Valor medio del vector de parámetros. Matriz 1 x 30.
  2. uPmed: Desviación estándar (incertidumbre típica) de cada parámetro 1x30.
  3. I_inf, I_sup: Vectores con los intervalos de confianza de cada parámetro. Vectores 1x30.
  4. Maxmed, minmed: Valores máximos y mínimos de los parámetros obtenidos. Vector 1x30.
  5. DifP: Error en la determinación de cada parámetro (medio – nominal). Vector 1x30.
  6. CP_Param: Matriz de parámetros obtenida. Matriz n x 30.
  7. Timefunc: Tiempo que ha necesitado la función en ejecutarse.

El pseudo-código de esta función es el siguiente:

Inicializa variables
Para cada iteración n
    Genera puntos Circle-Point
    Calcula Coordenadas_Plucker_Ejes_SIM
    Calcula_Parámetros_KM
    Guarda datos de parámetros
Fin Para
Obtiene datos de valor medio, incertidumbre, límite inferior y superior, valor máximo y mínimo
Guarda datos de salida
Muestra en pantalla datos de histograma de los parámetros D-H → d, θ, a, α, β
Muestra tabla de datos resumen
Muestra el tiempo empleado



5.10 Generación de un conjunto de puntos con el Simulador

Mediante el simulador se consigue generar un conjunto de puntos, que sería equivalente a los que obtendríamos en caso de realizar el mismo proceso con un sistema robot – equipo de medición. Por tanto, se ejecuta la instrucción:

>> [Ptos_Circle_Point, Ptos_Original]=Genera_puntos_circle_point(Initpos, SRLT, Preflector, Angcir, Nptos, Ruidomed, Err, Param5);

Teniendo que las variables introducidas las siguientes:

Tabla 2 - Asignación de valores a las variables para la generación de puntos

Initpos						
15	0	0	0	0	0	Posición inicial del robot - vector con ángulos de cada articulación

SRLT				
1	0	0	0	Matriz de transformación del Sistema de Referencia del robot al SR del Láser Tracker
0	1	0	0	
0	0	1	0	
0	0	0	1	

Preflector			
100	0	0	Matriz 6x3 con las coordenadas X Y Z de la posición del reflector del LT (en posición inicial) expresadas en el SR de cada articulación
200	0	0	
0	50	0	
20	0	0	
0	80	0	
0	60	0	

Angcir						
45	80	100	50	200	300	Vector 1x6 con el ángulo de giro que cubrirá cada articulación

Nptos						
15	20	30	15	25	30	Vector 1x6 con el número de puntos que se generarán en cada círculo

Ruidomed		
0	0	Desviación estándar de la distribución normal con la que se generará aleatoriamente ruido normal debido al instrumento de medida

Err

dx	dy	dz	Ex	Ey	T1	T2	T3	T4	T5	fi1	fi2	fi3	fi4	fi5	R1	R2	R3	R4	R5
0	0	0	0	0	6,2832	6,2832	6,2832	6,2832	6,2832	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	6,2832	6,2832	6,2832	6,2832	6,2832	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	6,2832	6,2832	6,2832	6,2832	6,2832	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	6,2832	6,2832	6,2832	6,2832	6,2832	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	6,2832	6,2832	6,2832	6,2832	6,2832	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	6,2832	6,2832	6,2832	6,2832	6,2832	0	0	0	0	0	0	0	0	0	0

Matriz 6x20 con los errores de giro a inducir en cada una de las articulaciones

Param5					
335	0	75	90	0	Vector con los parámetros DH del robot (d, thita, a, alfa, beta)
0	0	270	0	0	Es matriz 1x30
0	90	90	90	0	
295	0	0	90	0	
0	180	0	90	0	
80	0	0	0	0	

Obtenemos el siguiente conjunto de puntos:

Ilustración 8 - Puntos obtenidos

5.11 Validación de las funciones realizadas

Se han ejecutado las tres funciones desarrolladas, empezando por el simulador, para comprobar la validez de las mismas. Se ha ejecutado sin considerar ruido ni posibles errores. El sistema de referencia del equipo de medición (SRLT) se considera en la base del robot (matriz identidad).

>> shfig=true; Ruidomed=[0 0]

>> [Ptos_Circle_Point, Ptos_Original] = Genera_puntos_circle_point(Initpos, SRLT, Preflector, Angcir, Nptos, Ruidomed, Err, Param5, shfig);

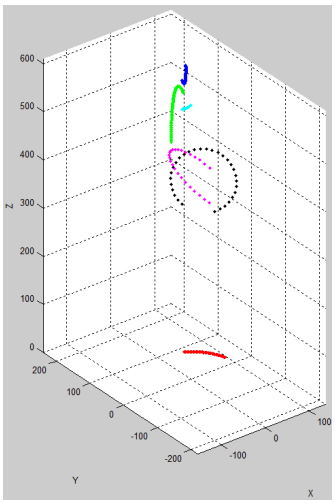


Ilustración 9 - Generación puntos

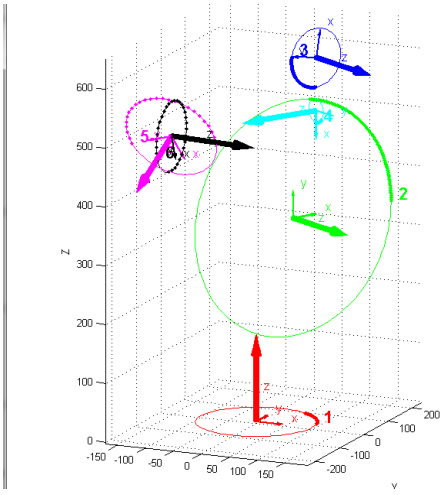


Ilustración 10 - Cálculo coordenadas Plücker

A continuación se ha ejecutado la función *Coordenadas_Plucker_Ejes*, para obtener los datos de coordenadas Plücker.

>> [Plucker, P_Circulos, SR0, Distcirculo, Distplano, Distcilindro, sigmaPlano, sigmaCirculo, sigmaCilindro, Angfin] = Coordenadas_Plucker_Ejes(Ptos_Circle_Point, Nptos, r0, Param5, shfig, shfig, shfig, shfig, shfig, SRLT)

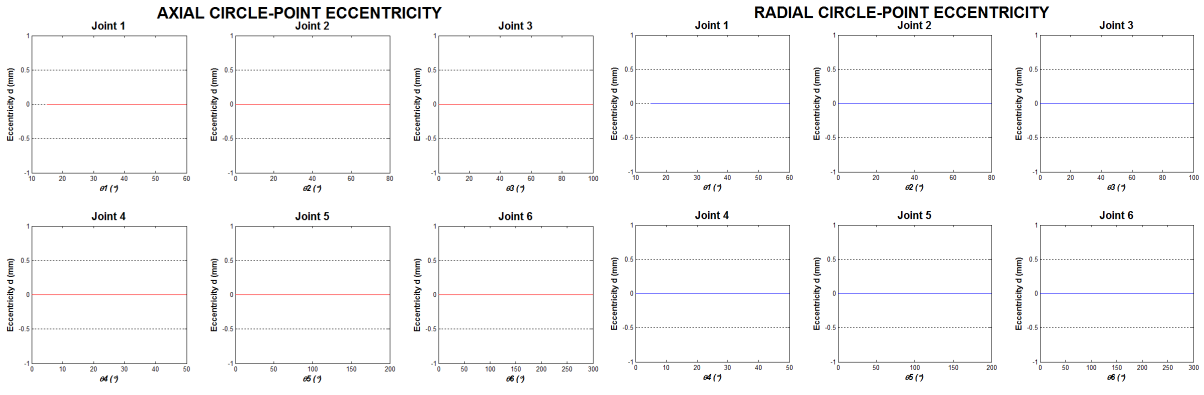


Ilustración 11 - Excentricidad obtenida sin ruido

Por último, se ha ejecutado la función *Calcula_Parametros_KM*, para obtener los parámetros (30) del modelo, según los cálculos realizados.

>> P=Calcula_Parametros_KM(Puckler,P_Circulos,Param5, Angfin, CheckX, SRLT, SR0, shfig)

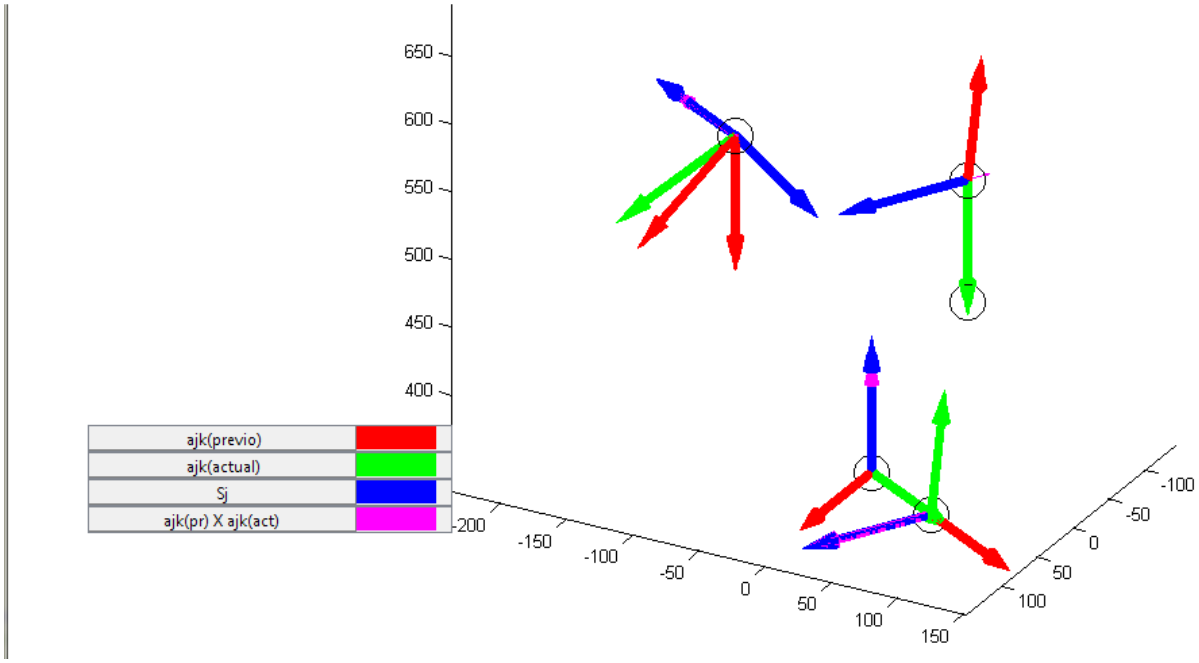


Ilustración 12 - Vectores ajk, ajk_prev, Sj, producto vectorial ajk x ajk_prev

Los parámetros obtenidos con esta función son los siguientes:

Tabla 3 - Parámetros D-H obtenidos en validación

	d	θ	a	α	β
1	335	$-7{,}11*10^{-15}$	75	90	0
2	0	$4{,}26*10^{-14}$	270	$3{,}38*10^{-15}$	$4,83*10^{-16}$
3	$2{,}01*10^{-14}$	90	90	90	0
4	295	$-7{,}11*10^{-15}$	0	90	0
5	$-9{,}80*10^{-13}$	180	0	90	0
6	80	0	0	0	0

Estos parámetros son muy similares a los calculados teóricamente, según el modelo Denavit-Hartenberg (ver Tabla 1 - Parámetros del modelo cinemático del robot Kuka, página 13).

5.12 Validación de la función de cálculo de los parámetros del modelo con Montecarlo, sin ruido ni errores

Se ha realizado la validación de esta función, igual que en el caso anterior, ejecutándose sin tener en cuenta ruido ni errores (todo cero). El sistema de referencia del equipo de medición (SRLT) se ha considerado en la base del robot (matriz identidad). Para esta validación se ha considerado realizar un cálculo de 10.000 iteraciones.

Tabla 4- Datos obtenidos sin ruidos ni errores

	d1	d2	d3	d4	d5	d6	thita 1	thita 2	thita 3	thita 4	thita 5	thita 6
Media	335.0000	0	1.1479e-13	295	-6.1024e-13	80	-7.1054e-15	2.8422e-14	90.0000	0	180.0000	0
Incertidumbre	5.7130e-14	0	1.5222e-28	0	8.1186e-28	0	0	0	9.9977e-14	0	5.7130e-14	0
Lim.Sup	335.0000	0	1.1479e-13	295	-6.1024e-13	80	-7.1054e-15	2.8422e-14	90.0000	0	180.0000	0
Lim.Inf	335.0000	0	1.1479e-13	295	-6.1024e-13	80	-7.1054e-15	2.8422e-14	90.0000	0	180.0000	0
Maximo	335.0000	0	1.1479e-13	295	-6.1024e-13	80	-7.1054e-15	2.8422e-14	90.0000	0	180.0000	0
Minimo	335.0000	0	1.1479e-13	295	-6.1024e-13	80	-7.1054e-15	2.8422e-14	90.0000	0	180.0000	0

	a1	a2	a3	a4	a5	a6	alfa 1	alfa 2	alfa3	alfa 4	alfa 5	alfa 6
Media	75.0000	270.0000	90.0000	0	0	0	90	6.7514e-15	90	90	90	0
Incertidumbre	1.1426e-13	1.1426e-13	4.2847e-14	0	0	0	0	4.7570e-30	0	0	0	0
Lim.Sup	75.0000	270.0000	90.0000	0	0	0	90	6.7514e-15	90	90	90	0
Lim.Inf	75.0000	270.0000	90.0000	0	0	0	90	6.7514e-15	90	90	90	0
Maximo	75.0000	270.0000	90.0000	0	0	0	90	6.7514e-15	90	90	90	0
Minimo	75.0000	270.0000	90.0000	0	0	0	90	6.7514e-15	90	90	90	0

	beta 1	beta 2	beta 3	beta 4	beta 5	beta 6
Media	0	1.1611e-14	0	0	0	0
Incertidumbre	0	6.3427e-30	0	0	0	0
Lim.Sup	0	1.1611e-14	0	0	0	0
Lim.Inf	0	1.1611e-14	0	0	0	0
Maximo	0	1.1611e-14	0	0	0	0
Minimo	0	1.1611e-14	0	0	0	0

5.13 Validación de la función de cálculo de los parámetros del modelo con Montecarlo, incluyendo ruido

Se ha realizado esta simulación de forma similar al caso anterior, ejecutándose incluyéndose ruido debido al equipo de medición, pero no errores (todo cero). El ruido considerado es 5µm+0.3µm/m. El sistema de referencia del equipo de medición (SRLT) se ha tomado en la base del robot (matriz identidad). Esta validación se ha determinado que se realice con un cálculo de 10.000 iteraciones.

Tabla 5 - Datos obtenidos del modelo incluyendo ruido

	d1	d2	d3	d4	d5	d6	thita 1	thita 2	thita 3	thita 4	thita 5	thita 6
Media	335.0000	0	-1.8907e-04	295.0005	-0.0011	80	-1.0422e-05	2.3049e-05	90.0004	-1.9093e-05	179.9996	0
Incertidumbre	0.0122	0	0.0268	0.0964	0.1522	0	8.3458e-04	0.0091	0.0614	0.0025	0.0614	0
Lim.Sup	335.0237	0	0.0527	295.1906	0.2941	80	0.0016	0.0178	90.1218	0.0049	180.1186	0
Lim.Inf	334.9761	0	-0.0535	294.8115	-0.2974	80	-0.0016	-0.0179	89.8801	-0.0050	179.8794	0
Maximo	335.0439	0	0.1045	295.3343	0.5665	80	0.0031	0.0330	90.2125	0.0099	180.2243	0
Minimo	334.9512	0	-0.1000	294.6514	-0.5327	80	-0.0033	-0.0359	89.7791	-0.0104	179.7844	0

	a1	a2	a3	a4	a5	a6	alfa 1	alfa 2	alfa3	alfa 4	alfa 5	alfa 6
Media	75.0003	270.0000	89.9996	-0.0016	2.3786e-06	0	90.0000	4.3251e-05	89.9999	89.9998	90.0000	0
Incertidumbre	0.0558	0.0037	0.0219	0.3168	0.0017	0	0.0040	0.0024	0.0297	0.0295	9.4698e-04	0
Lim.Sup	75.1095	270.0075	90.0419	0.6181	0.0032	0	90.0077	0.0046	90.0575	90.0565	90.0018	0
Lim.Inf	74.8887	269.9927	89.9574	-0.6228	-0.0032	0	89.9922	-0.0047	89.9419	89.9418	89.9981	0
Maximo	75.2142	270.0152	90.0805	1.1727	0.0066	0	90.0157	0.0083	90.1107	90.1097	90.0038	0
Minimo	74.7999	269.9854	89.9214	-1.1419	-0.0061	0	89.9854	-0.0093	89.8998	89.8987	89.9964	0

	beta 1	beta 2	beta 3	beta 4	beta 5	beta 6
Media	0	2.3890e-05	0	0	0	0
Incertidumbre	0	0.0028	0	0	0	0
Lim.Sup	0	0.0054	0	0	0	0
Lim.Inf	0	-0.0055	0	0	0	0
Maximo	0	0.0100	0	0	0	0
Minimo	0	-0.0105	0	0	0	0

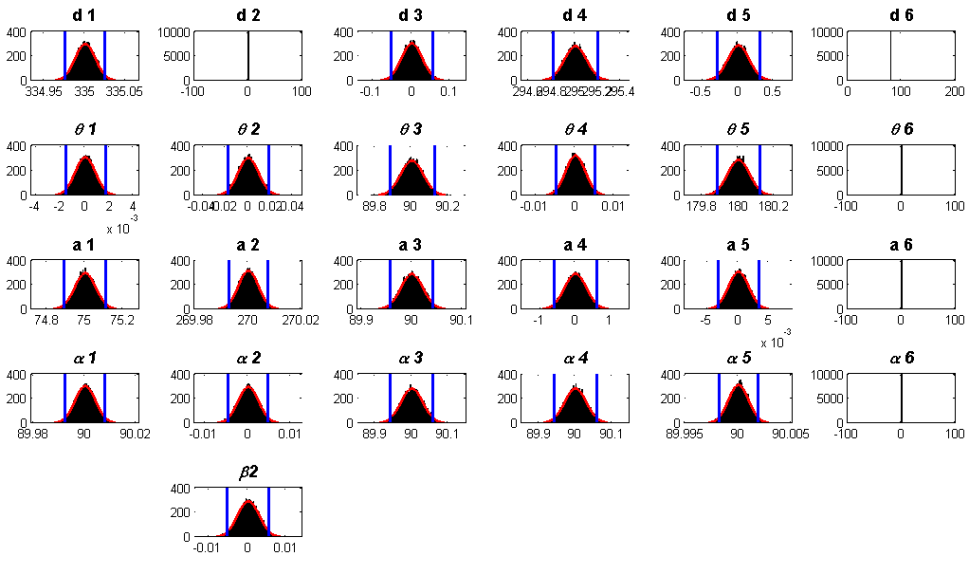


Ilustración 13 - Gráfica datos obtenidos incluyendo ruido

6 Simulaciones realizadas y resultados obtenidos

Se han realizado un conjunto de simulaciones, con el objetivo de verificar los valores medios y las incertidumbres de medición obtenidas en las diferentes condiciones consideradas a continuación:

1. Se ha determinado primeramente el uso de diferentes equipos de medición (ver los 5 equipos de medición mostrados en la Tabla 13 - Incertidumbre de equipos de medición, en página 153).
2. Con un equipo de medición definido (p.ej., Láser Tracker), se ha tomado una articulación y se ha analizado la influencia de la variación del ángulo cubierto en la incertidumbre obtenida. Para esta simulación se han tomado los ángulos de 15°, 40°, 90° y 180°.
3. Con el mismo equipo de medición tomado anteriormente, se ha modificado la posición del reflector (variable *Preflector*), en una articulación (p.ej., la articulación 3), fijando 3 valores diferentes, y se ha comprobado su influencia en la incertidumbre obtenida.
4. Para el mismo caso de equipo de medición, se ha variado el número de puntos considerados (10, 40, 100).
5. Para el mismo caso de equipo de medición, se ha cambiado la matriz de cambio del robot al Láser Tracker, considerando 3 valores diferentes (ver los desplazamientos tomados y las matrices calculadas en el apartado 15.2 Cálculo de las diferentes matrices de transformación para la simulación, página 82).
6. Asimismo, se han realizado dos conjuntos de simulaciones más:
   6.1. Para un ordenador PC i7 64 bits, con 4Gb de RAM, S.O. Windows 7, se ha tomado un tipo de simulación, y se ha variado el número de iteraciones, para comparar tiempo de ejecución, incertidumbre y exactitud de cálculo obtenida.
   6.2. Se ha hecho el mismo tipo de variación de número de iteraciones, en este caso para un PC portátil, con S.O. Windows XP SP2, y 2Gb RAM, tomando un tipo de simulación definido.

Las variables iniciales con las que se han llevado a cabo estas simulaciones se muestran en el capítulo 13. Asimismo, se ha creado la función *Simulacion_MCP*, para poder ejecutar cada una de las simulaciones aquí propuestas de forma rápida.

Para realizar una comparación adecuada entre simulaciones se calcula la exactitud de los datos obtenidos en la simulación con el parámetro *CpK* (15), que nos permite combinar en un parámetro la calidad de la media y de la incertidumbre obtenida. Este parámetro se calcula según la fórmula:

$$CpK = \frac{\min\left(Lim_{sup} - \bar{x}, \bar{x} - Lim_{inf}\right)}{3\sigma}$$ Ecuación 3

tomándose como valores medios, los parámetros teóricos del modelo, y como límites superior e inferior se toma 0,1mm. Aunque este parámetro se usa para verificar la estabilidad de un proceso productivo, en este caso nos sirve al poder combinar en uno solo los dos parámetros de media e incertidumbre que nos interesan, y así poder representar en una sola gráfica el parámetro resultante, obteniendo un valor equiparable entre los distintos parámetros, y entre las simulaciones consideradas.

El detalle de datos obtenidos en las simulaciones se puede ver en el Anexo VIII: Resultados en detalle de las simulaciones, en página 84.

6.1 Análisis de la influencia del uso de diferentes equipos de medición

Se considera en este análisis el uso de diferentes equipos de medición (ver los 5 equipos de medición mostrados en la Tabla 13 - Incertidumbre de equipos de medición, en página 153).

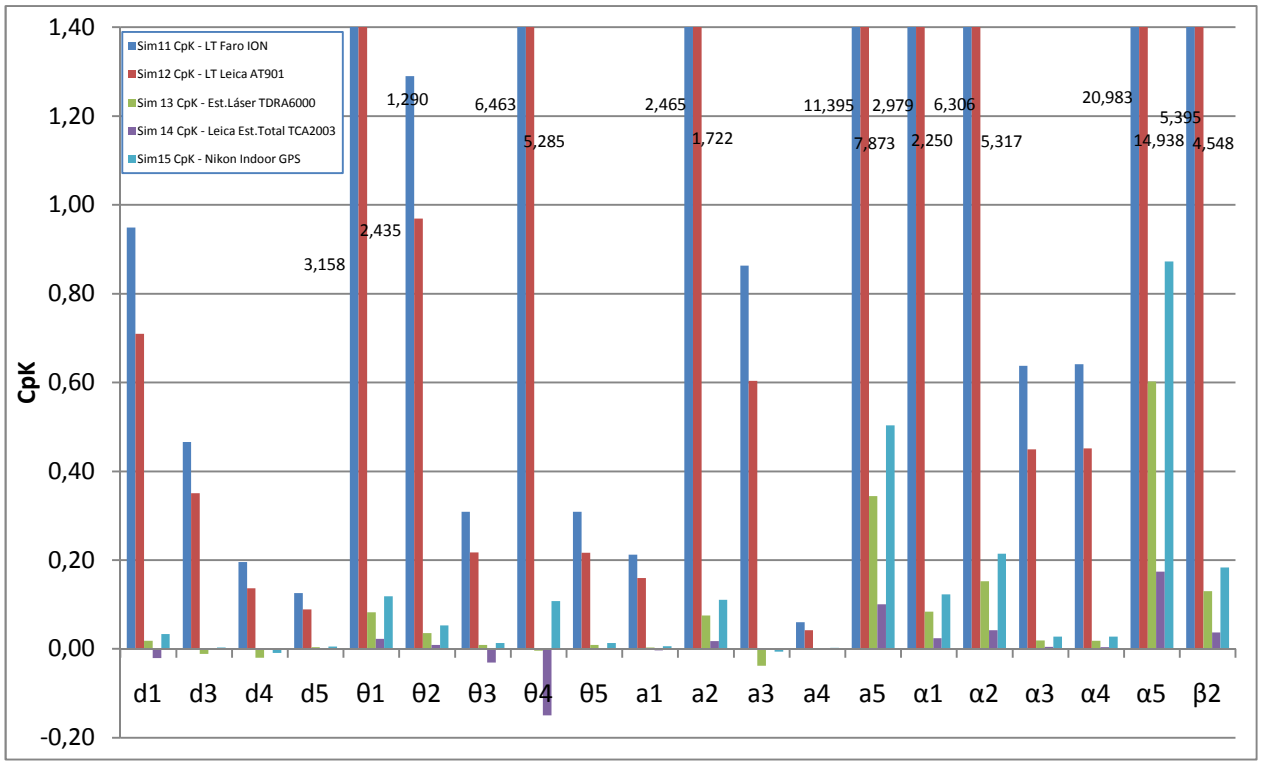


Ilustración 14 - CpK vs diferentes equipos de medición

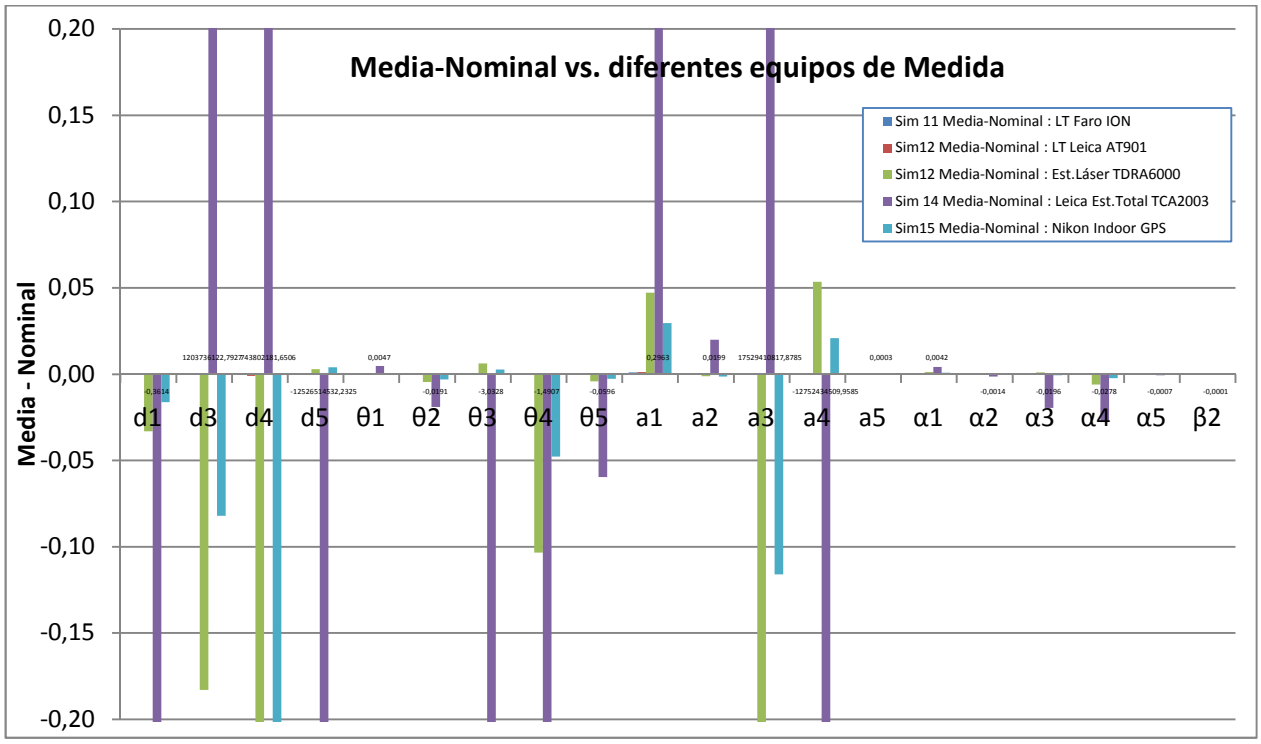


Ilustración 15 - (Media - Nominal) vs diferentes equipos de medición

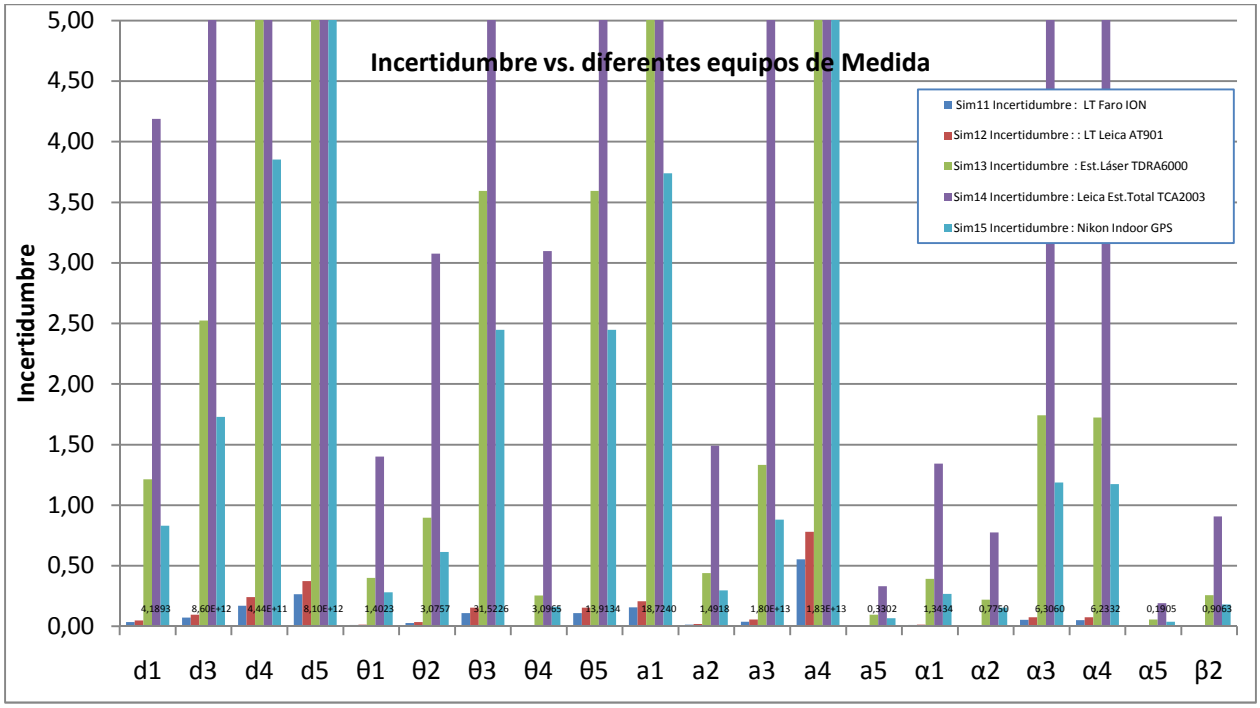


Ilustración 16 - Incertidumbre vs diferentes equipos de medida

Primeramente, se puede contrastar que los resultados obtenidos en la gráfica de Media-Nominal y en la de Incertidumbre, son equiparables a los vistos en la gráfica CpK, ya que en aquellos casos en que la diferencia entre media y nominal, y la incertidumbre, presenta un valor bajo, el valor de CpK es muy alto, y viceversa. Por tanto, un valor alto de CpK > 1.33[7] sería un valor bueno, y cuanto más alto mejor, y un valor inferior, sería malo, e incluso valores negativos implica una media e incertidumbre pésimas. Esto siempre teniendo en cuenta que este "proceso" (establecido arbitrariamente para obtener unas gráficas únicas), este valor de CpK, se ha calculado sobre la base de unos límites de proceso superior e inferior de ±0,1mm.

Se puede ver que los dos primeros equipos de medición, tanto el Láser Tracker Faro ION, como el Láser Tracker Leica AT901, dan unos valores de CpK en muchos casos por encima de 1.00, lo cual es un valor bueno de proceso, incluso por encima de 1.33. Tanto el equipo Nikon Indoor GPS como la Estación Láser TDRA6000 de Leica se quedan lejos de los anteriores, ya sólo para algunos valores , como a5 y α5, llegan a superar sólo 0.2, lo cual son valores bajos de CpK, situación que por otra parte viene dada por la incertidumbre que ya viene dada por sus características. En cuanto a la Estación Total TCA2003 de Leica, da valores que, si bien para aplicaciones de Obras Públicas, en los que las distancias a cubrir son del entorno a 1000-2000m, en esta aplicación se queda muy lejos de los otros equipos, e incluso en la simulación ha dado algunos errores, ya que en el cálculo del círculo (por el método de Gauss-Newton) llegaba a dar error, al obtener matrices singulares que no permitían realizar el cálculo correcto del círculo correspondiente al conjunto de puntos dados, debido a la elevada incertidumbre del equipo.

[7] Valor dado para un proceso estable y capaz, según fabricantes de automóviles, p.ej., General Motors, Ford (15).

6.2 Análisis de la influencia de la variación del ángulo cubierto en la incertidumbre

Tomando un equipo de medición, que será el Láser Tracker, se elige la articulación 4, y se lanzan simulaciones, tomándose los siguientes ángulos: 15°, 40°, 90° y 180°.

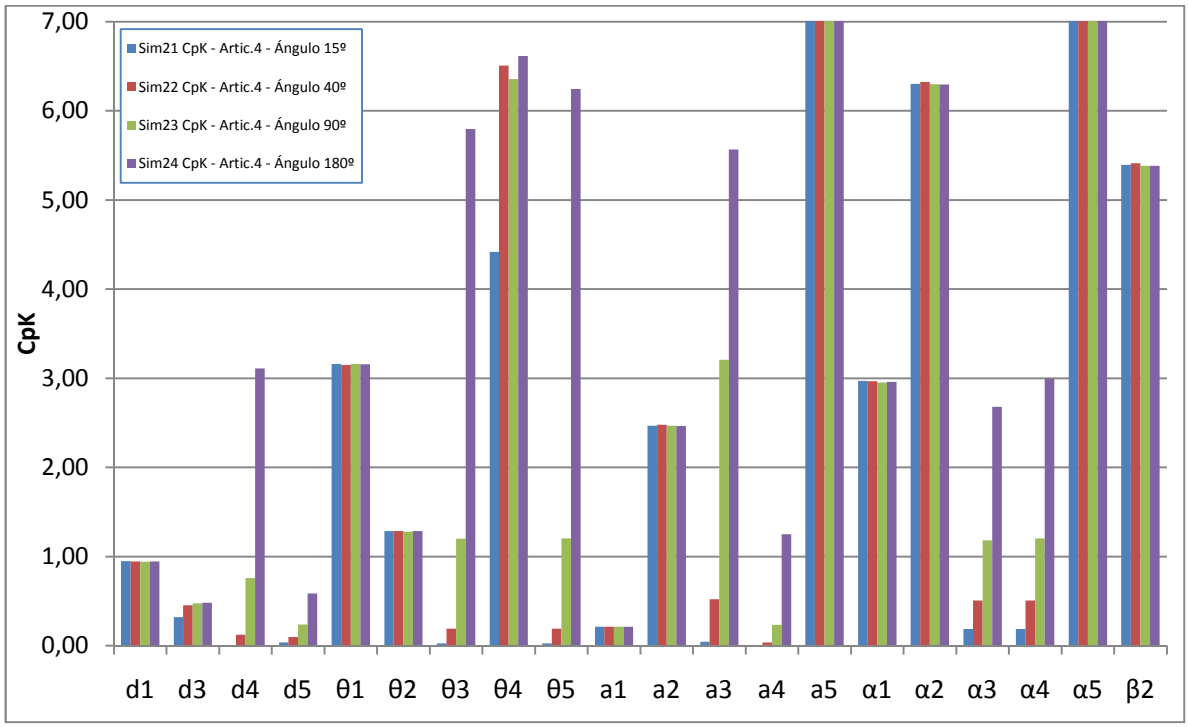


Ilustración 17 - CpK vs ángulo cubierto por el reflector

Como se puede ver en la gráfica, al haber variado el ángulo que recorre el reflector en la articulación, es en esta articulación, y en la anterior y en la siguiente, en las que se ven los efectos de esta variación. Los parámetros d4, d5, θ3, θ5, a3, a4, α3, α4, y en menor medida θ4, consiguen unos valores de CpK mayores cuanto mayor es el ángulo recorrido por el reflector, lo cual es lógico, ya que cuanto mayor sea el ángulo que recorre el reflector mejor será el calculo que realizará el método, y menor será la incertidumbre del proceso.

6.3 Análisis de la influencia de la variación de la posición del reflector en la incertidumbre

Con el mismo equipo de medición tomado anteriormente, se modifica la posición del reflector (variable *Preflector*), en una articulación (nº 3), fijando 3 valores, y comprobando su influencia en la incertidumbre obtenida. Se fijan los valores (0,50,0), (0,70,20) y (20,100,50), en coordenadas (X,Y,Z).

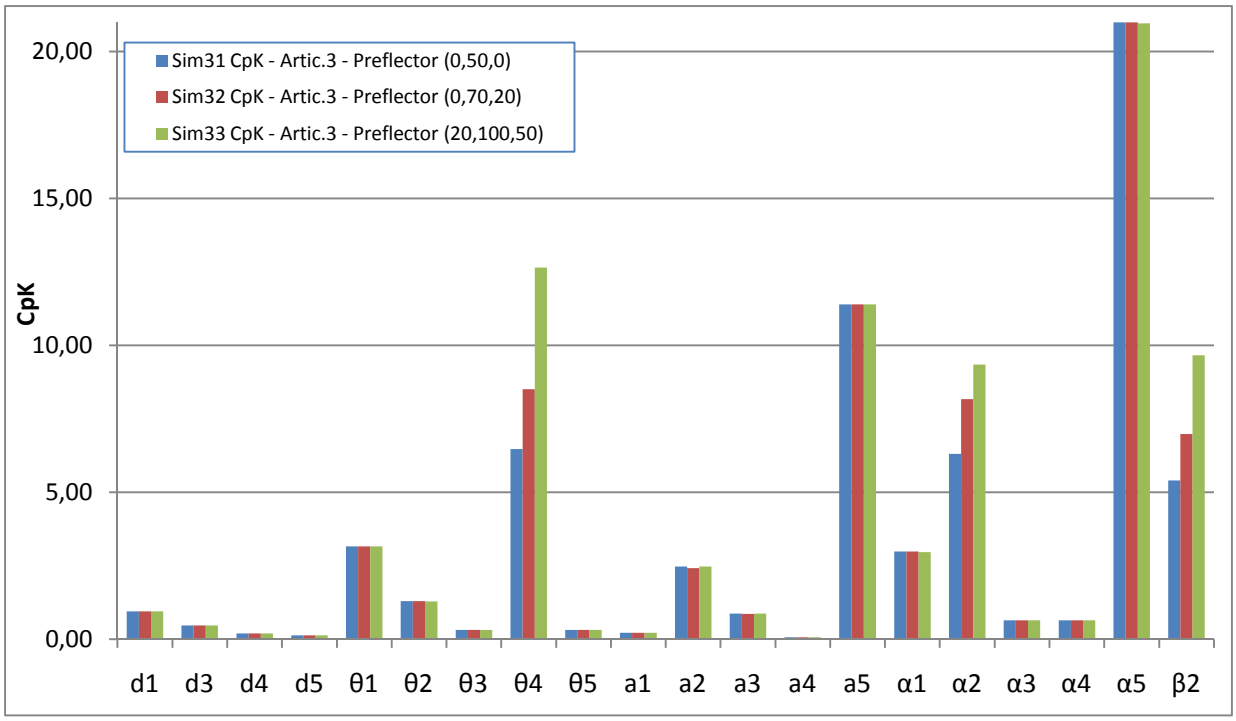


Ilustración 18 - CpK vs variación posición del reflector

En la gráfica comparativa del CpK se observa que la posición del reflector tiene poca influencia en los datos finales obtenidos en la simulación y cálculo por el método del Circle-Point, y que únicamente los parámetros θ4, α2 y β2 se ven ligeramente afectados, obteniéndose eso sí mejores valores para posiciones del reflector más alejadas. Sin embargo, esta variación es en una medida muy baja: por ejemplo, para θ4, supone pasar, en la simulación 31, de valores de límites superior/inferior de ±10µm, a valores de ±7µm, en la simulación 32, y ±5µm, en la simulación 33. En la siguiente tabla se puede ver la leve variación que afectan a estos parámetros.

Tabla 6 - Parámetros afectados por variación de la posición del reflector

		θ4	α2	β2
	Nominal	0	0	0
	Tol	0,1	0,1	0,1
	Tol.Sup.	0,1	0,1	0,1
	Tol.Inf.	-0,1	-0,1	-0,1
31	Media	-0,0001	0,0000	0,0000
Artic. 3	Desviación Estándar	0,0052	0,0053	0,0062
Preflector=[0 50 0]	Sim31 CpK - Artic.3 - Preflector (0,50,0)	6,4628	6,3056	5,3953
32	Media	-0,0001	0,0000	0,0000
Artic. 3	Desviación Estándar	0,0039	0,0041	0,0048
Preflector=[0 70 20]	Sim32 CpK - Artic.3 - Preflector (0,70,20)	8,5058	8,1634	6,9793
33	Media	-0,0001	0,0000	0,0000
Artic. 3	Desviación Estándar	0,0026	0,0036	0,0035
Preflector=[20 100 50]	Sim33 CpK - Artic.3 - Preflector (20,100,50)	12,6471	9,3429	9,6557

6.4 Análisis de la influencia de la variación del número de puntos considerado que se miden para cada articulación

Para el mismo caso considerado anteriormente, se varía el número de puntos considerados (10, 40, 100).

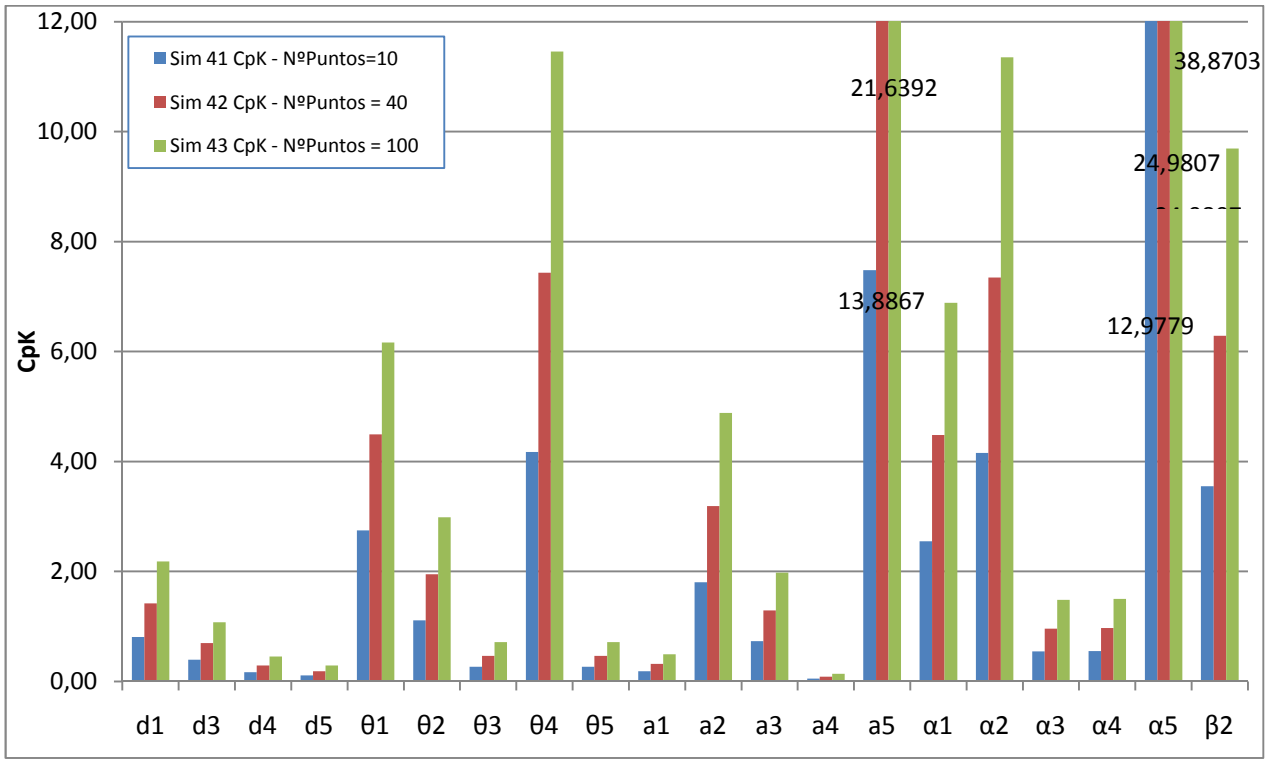


Ilustración 19 - CpK vs Variación del número de puntos considerado

En el gráfico se puede ver que cuanto mayor es el número de puntos que se considera para realizar la toma de datos para la posterior definición del círculo, mayor es la precisión obtenida, y por tanto, mayor es el parámetro CpK, es decir, más ajustada será la media y menor será la incertidumbre obtenida.

6.5 Análisis de la influencia de la variación de la matriz de transformación del SR del robot al equipo de medición

Para el mismo caso de equipo de medición, se cambia la matriz de cambio del robot al Láser Tracker, considerando 3 valores diferentes (ver los desplazamientos tomados y las matrices calculadas en el apartado 15.2 Cálculo de las diferentes matrices de transformación para la simulación, página 82).

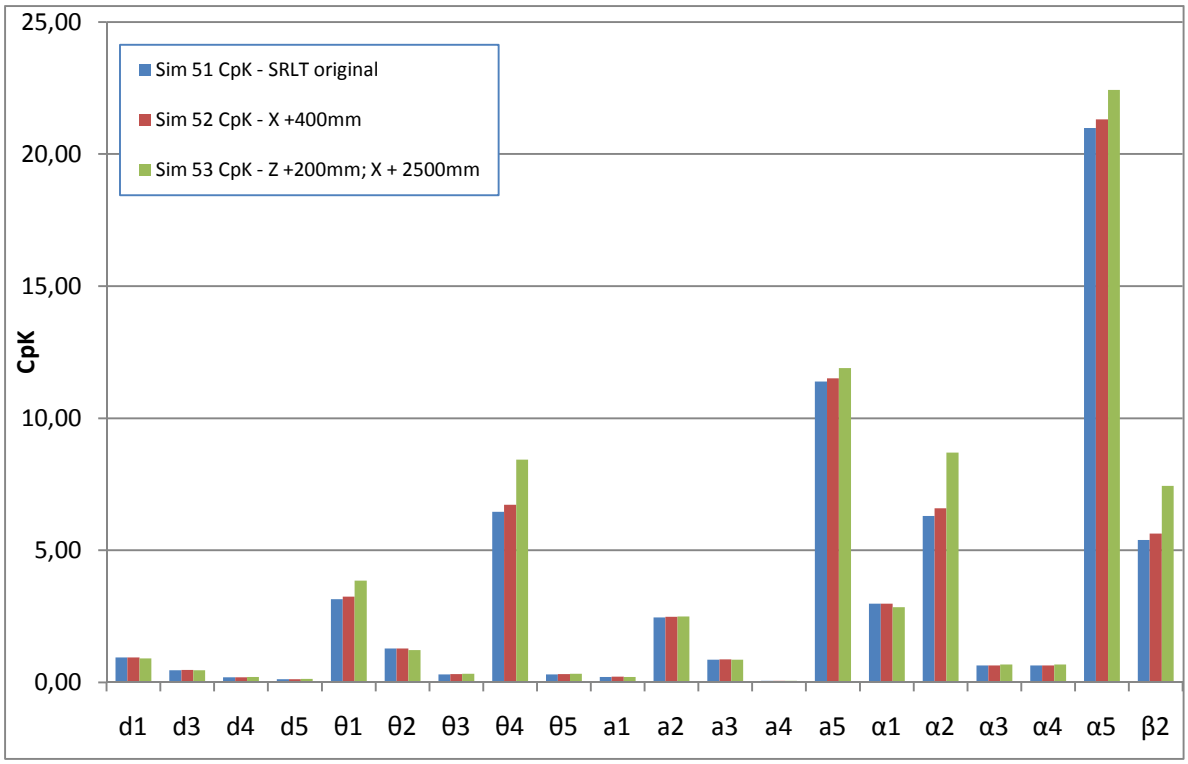


Ilustración 20 - CpK vs Variación de la matriz de transformación (posición del equipo de medición)

En la gráfica se puede observar que, en lo que supone la modificación de la matriz de cambio de base del sistema de referencia del robot al sistema de referencia del equipo de medición, no hay diferencias relevantes en los resultados obtenidos.

6.6 Análisis de la influencia de la variación del número de iteraciones ejecutado en los cálculos

Se consideran dos equipos, un PC i7 64 bits, y un ordenador portátil con CPU Intel Core 2 T5600.

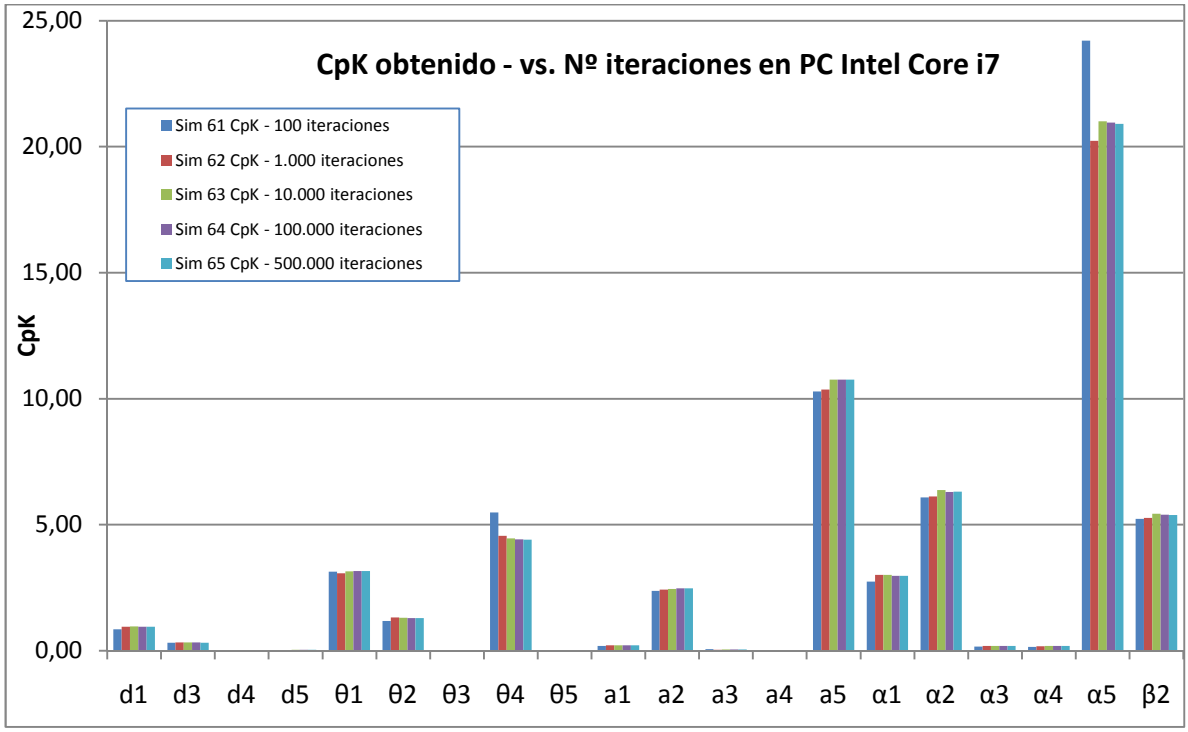


Ilustración 21 - CpK vs Número iteraciones en PC Intel Core i7 64 bit

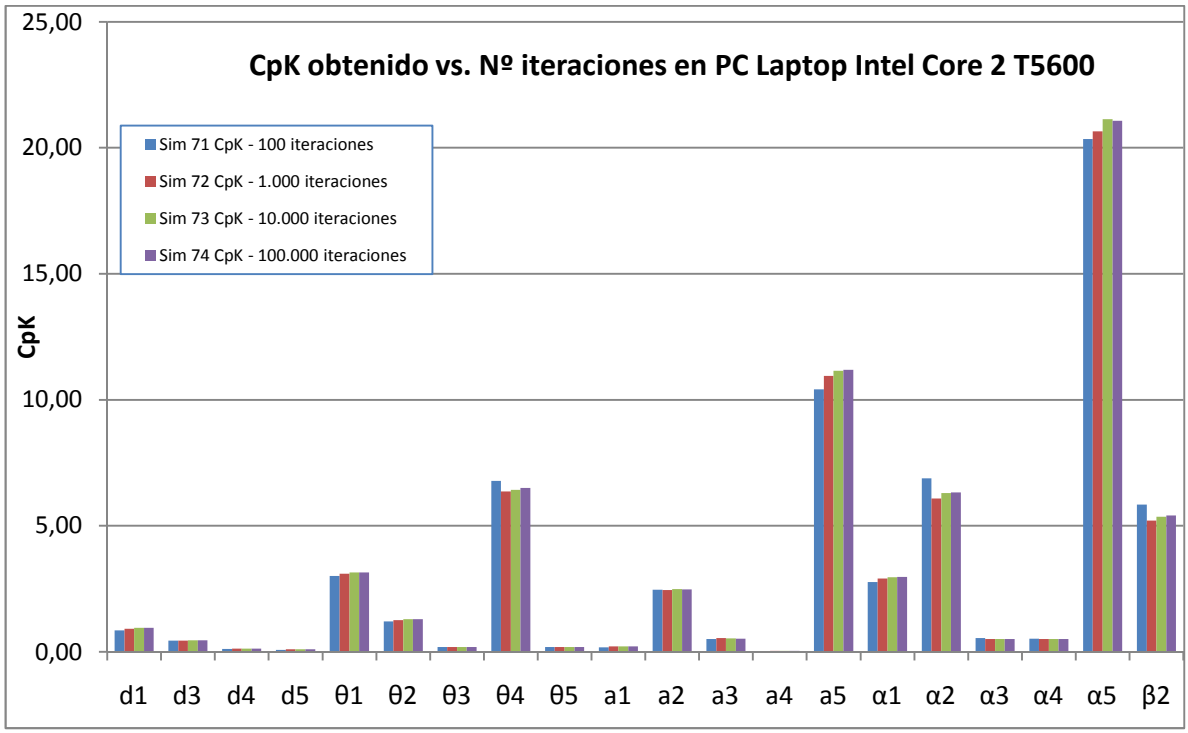


Ilustración 22 - CpK vs Número iteraciones en PC Intel Core 2 T5600

Asimismo se adjuntan las gráficas del valor medio menos la nominal obtenido en estas simulaciones.

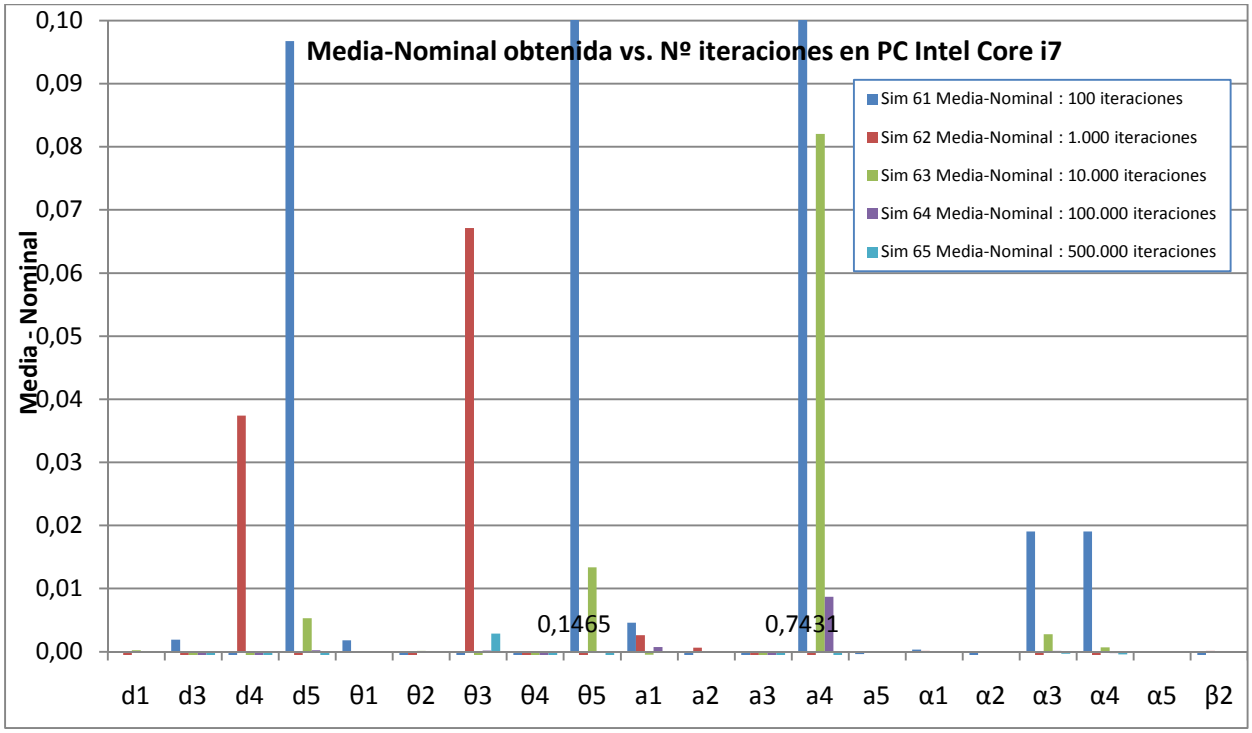


Ilustración 23 - Media-nominal vs. número de iteraciones en PC i7

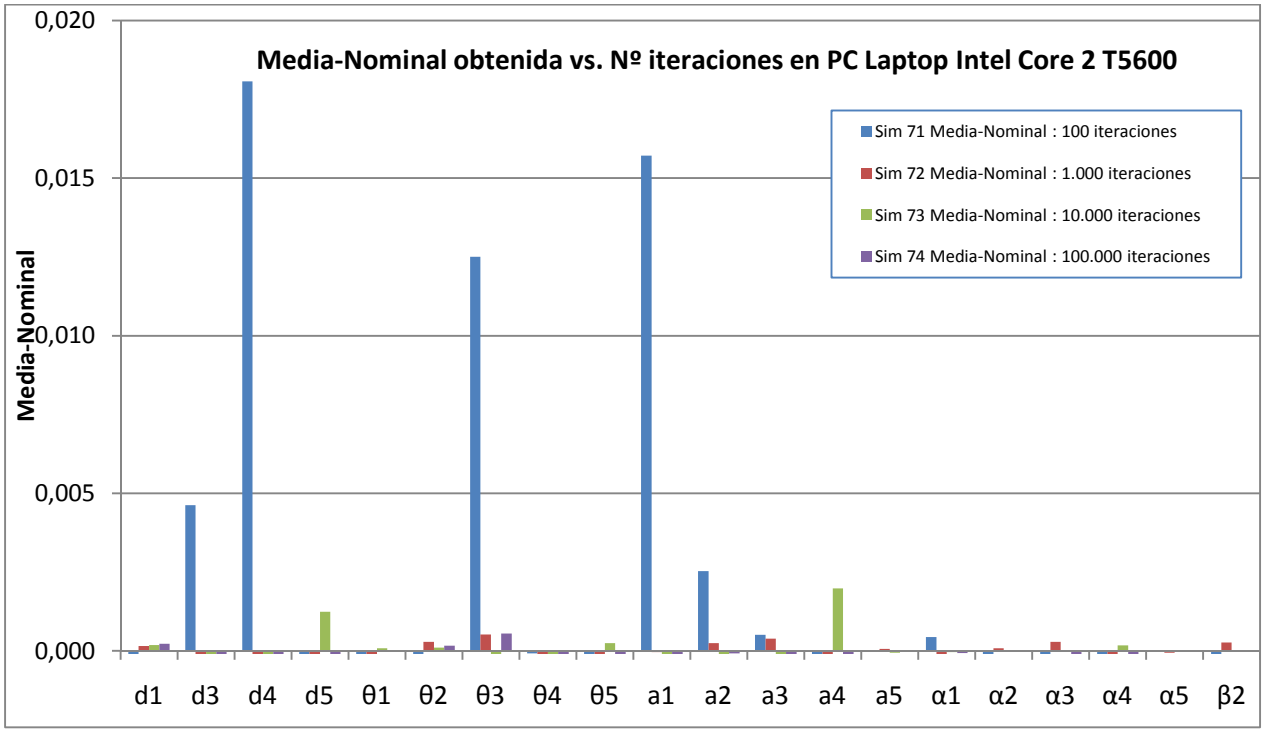


Ilustración 24 - Media-Nominal vs Número iteraciones en PC Intel Core 2 T5600

Como se puede ver en las gráficas, las diferencias entre 10^4 y 10^5 iteraciones son pequeñas, y prácticamente inapreciables ya entre 10^5 y $5*10^5$ iteraciones.

6.7 Tiempos de ejecución

Para el cálculo del proyecto se han empleado varios ordenadores, cuyas características principales se pueden ver en la siguiente tabla. Se incluyen los tiempos de ejecución que se han obtenido de la simulación por Montecarlo, para cada ordenador, en función del nº de iteraciones.

No se han realizado ya simulaciones de 1.000.000 iteraciones, ya que el tiempo empleado sería excesivo (superior a 36 horas).

Tabla 7 - Comparativa tiempos de ejecución

Iteraciones	Ordenador 1 CPU: Intel Core i7 (64bits) Velocidad: 2.80 GHz S.O.: Windows 7 Professional Memoria RAM: 4 Gb	Ordenador 2 CPU: Intel Core 2 T5600 (32 bits) Velocidad: 1.83 GHz S.O.: WinXP Home Edition v2002 SP3 Memoria RAM: 2 Gb
100	18,2"	13,38"
1.000	2' 57"	2' 30"
10.000	30' 40"	24' 46,2"
100.000	4h 42' 13,7"	4h 8' 37"
500.000	18h 8' 52,0"	---[8]

[8] En la simulación con 500.000 iteraciones ha aparecido el error "Out of memory", no pudiéndose computar.

7 Próximos pasos

Como próximos pasos y trabajo futuro a realizar, a partir de la situación actual del proyecto, se tendrían los siguientes:

- Optimización de la rutina de cálculo del círculo para evitar errores por matriz singular, con errores muy elevados (p.ej., en la simulación nº 14).
- Programar módulos independientes de modelos de incertidumbre, de cada equipo de medición, en función de su estructura cinemática y principio de medida. Por ejemplo, dicho módulo daría la posibilidad de definir el ruido de entrada (ver el parámetro *Ruidomed* de la función *Genera_Puntos_Circle_Point*), para equipos de medida como un Láser Tracker, en 2 partes:
  - ✓ Una como un ruido en la dirección del láser, debido al interferómetro,
  - ✓ Y otra como un ruido, según el ángulo, debido al error de los encoder del Láser Tracker.
- Realización de simulaciones incluyendo el estudio de los errores de excentricidad, en el simulador, para contrastar el resultado obtenido por el método de *Circle-Point*.
- Realización de simulaciones en sistemas de Cálculo Paralelo (de mayor velocidad de cálculo que un ordenador personal), que permitirían realizar iteraciones del orden de 10^6, para comparar resultados obtenidos en él con los obtenidos en el presente trabajo.
- Validación experimental, es decir, comprobación del método con datos reales, ensayando el método con un robot y Láser Tracker. La medición física debería incluir la verificación en sentido de ida y vuelta, en cada articulación. En este sentido también se debería de verificar que los parámetros obtenidos en la práctica están dentro de los obtenidos teóricamente.
- Validación experimental, como se ha comentado anteriormente, pero con un robot distinto, confirmando la característica modular de las funciones desarrolladas.
- Comparación con otros métodos de identificación basados en optimizaciones tradicionales no lineales (16).

Además de los estudios citados anteriormente, que podrían considerarse como una primera fase, se podría pasar a una segunda fase de trabajos y estudios, en la que se abordarían estudios y análisis de mayor complejidad y laboriosidad:

- ➢ Análisis de la influencia de la incertidumbre del conjunto de parámetros obtenidos sobre el error final de posicionamiento del brazo del robot, para *n* iteraciones. De esta forma se podría comprobar la influencia de la incertidumbre del método en los posibles errores finales de posicionamiento de la herramienta en la articulación final (*tool frame*). Será importante, en este análisis, definir de forma precisa el volumen a considerar para esta comprobación, dado que la naturaleza no lineal del modelo cinemático hace que la influencia del valor de los parámetros del modelo sobre el error final dependa de la posición del robot.
- ➢ Modelar las variaciones de los parámetros con la temperatura, sobre todo en parámetros de distancia, para ver la influencia de éste. Debido a la dificultad que este modelado encierra, y a la gran cantidad de variables que se deberían tener en cuenta, este modelado se realiza empíricamente habitualmente.
- ➢ Identificación del mejor conjunto de parámetros que identifican de forma óptima al robot: Para realizar esta identificación, se debería hacer un análisis del error de posicionamiento del brazo robot, en su posición de la herramienta en la articulación final, en diferentes posiciones y diferentes puntos en el volumen del robot. Para cada posición se tendría un distribución normal

de posicionamiento, en función de los parámetros del modelo dados, y entonces habría un conjunto de parámetros de permitiría obtener error cero. Con un análisis ulterior se podrían combinar estos conjuntos de parámetros de error cero, para obtener el mejor conjunto de parámetros general.

8 Conclusiones

El presente proyecto se ha basado en el método de Circle-Point, como herramienta para conseguir definir los parámetros del modelo cinemático del robot, a partir de una serie de puntos obtenidos, bien sea mediante medición en el robot, bien sea a través de la simulación. El punto clave de esta técnica es que es capaz de establecer las dimensiones físicas específicas, sin necesidad de un modelo previo, es decir, permite medir físicamente la articulación. Por otra parte, esta técnica no es capaz dar por sí sola los parámetros de la base del robot, en su articulación 1, y requiere de otros métodos para conseguirlo. Esto es lo que, en efecto, se ha hecho en el proyecto, por lo que hay que resaltar que es importante empezar con la captura de varios puntos para poder obtener, por mínimos cuadrados, la matriz de transformación del robot al equipo de medición. Esto supone una mejora a destacar.

Como base de esta técnica se ha tenido el modelo cinemático de Denavit-Hartenberg, que define un conjunto de 4 parámetros (d_j, θ_j, a_{jk}, α_{jk}) para cada articulación, modificado según Hayati-Mirmirani, que añade un quinto parámetro β_2, en casos como el que se da en este trabajo, en el que dos articulaciones consecutivas tienen ejes paralelos o casi paralelos, como son las articulaciones 2 y 3.

En el inicio del proyecto se ha tenido que considerar la técnica experimental para la toma de puntos que define el método Circle-Point, ya que es en base a ésta técnica como se ha definido la función simulador (*Genera_Puntos_Circle_Point*). A partir de este conjunto de puntos la siguiente función (*Coordenadas_Plucker_Ejes*) calcula las coordenadas Plucker de cada articulación, y la siguiente (*Calcula_Parametros_KM*) es la que permite obtener los parámetros del modelo, a partir de estas coordenadas.

En todas estas funciones se han incluido representaciones gráficas de los resultados que permiten una mejor comprensión en el espacio de los valores de los vectores obtenidos en cada una, lo cual supone una mejora sustancial, al dotar a la función de una forma clara de validación gráfica. Además, se ha realizado la validación del conjunto de funciones, tanto sin error como con error, lo cual permite confirmar la bondad del funcionamiento del conjunto de funciones.

Aunque la técnica de Circle-Point establece las dimensiones físicas del modelo, y no es necesario tener los parámetros teóricos del mismo previamente, este dato se ha incluido, ya que en las simulaciones lanzadas con la función iterativa (*Montecarlo_Circle_Point*), en parámetros de valor teórico 0, pequeñas variaciones introducidas por Montecarlo suponen que el valor puede pasar a valores negativos, lo cual puede llegar a generar vectores directores de la articulación cuya dirección es la opuesta a la dirección real de la articulación. Por tanto, en las funciones antes vistas esta situación se controla y se asegura que la dirección del vector calculado (tanto S_j como a_j) coincide con el vector del eje del modelo nominal. Esto supone una mejora sustancial de la técnica, al evitar estos errores durante las sucesivas iteraciones.

Por otro lado, la realización de la batería de simulaciones ha permitido obtener unos resultados comparativos y una verificación de la simulación en función de variaciones de diversos parámetros (modificando ángulo descrito, número de puntos, número de iteraciones, posición del reflector, posición del equipo de medida, etc.), obteniéndose unos datos interesantes, tanto por esperados en algunos casos, como por distintos a las expectativas, en otros.

El desarrollo de los procedimientos en MatLab ha implicado un considerable esfuerzo de síntesis, organización, metodología y orden en la elaboración de estas funciones, que se puede ver recompensado al final en dos áreas del proyecto:

- en la generación de un pseudo-código explicativo que, si bien no es un código nemotécnico al uso, sí que permite una comprensión clara de la función, de una forma conjunta,
- y en la correlación de este pseudo-código con el código MatLab existente, que sigue la misma ordenación, lo que permite un seguimiento y comprensión clara de la función, y de su desarrollo matemático.

La realización del proyecto en conjunto ha supuesto un reto importante, ya que:

- por un lado, implica la aplicación de principios de geometría y álgebra (cálculos de perpendicularidad de líneas, planos que cruzan a líneas, etc.), definiciones y relaciones con las cuales no se estaba en absoluto familiarizado,
- y por otro, el desarrollo de funciones complejas en MatLab, entorno de trabajo y programación que supone ya de por sí un reto, lo es aún más por la complejidad y elaboración de las funciones aquí desarrolladas.

Estas dificultades han supuesto una mayor inversión de tiempo en el desarrollo y aseguramiento de los pasos, funciones y simulaciones comentados hasta este punto, por lo que se ha llegado hasta aquí sin poder llegar a concluir la fase de validación experimental, si bien parte de la misma ya se había llevado a cabo. Asimismo, esta validación experimental supone un desarrollo específico de funciones de MatLab, para poder integrar la simulación teórica en una simulación real en el robot, con lo que también implica un desarrollo de software en el lenguaje del robot Kuka. Este apartado queda por tanto como trabajo a desarrollar en el futuro.

Por último, como elementos a destacar del método Circle-Point, y de este desarrollo serían que:

- Se ha desarrollado una herramienta que permite offline obtener las incertidumbres asociadas a cada parámetro del modelo, según se ha visto en las simulaciones ejecutadas, permitiendo de este modo obtener el rango de posibles valores que se obtendrán en una medición real en las condiciones de medida consideradas. Esta circunstancia permite valorar las condiciones de medida ideales para obtener la máxima precisión posible en la identificación de parámetros.
- Es una herramienta completa de simulación, que establece una sistemática de trabajo que supone una nueva aproximación al problema de identificación por obtener la incertidumbre asociada a cada parámetro en función de las condiciones de medida no recogida hasta la fecha en la bibliografía. Supone, por tanto, una innovación en el campo de los procedimientos de identificación de parámetros cinemáticos.
- Mantiene en mejor medida el enlace físico-matemático entre los resultados del procedimiento de identificación y los valores reales del robot, ya que mide físicamente la articulación.

20 Bibliografía

1. **Denavit J., Hartengerg R.S.** *A kinematic notation for lower pair mechanisms based on matrices.* s.l. : Journal of Applied Mechanics, 1955, Vol. v.22, págs. 215-221.

2. **R.P., Paul.** *Robot Manipulators: Mathematics, Programming and Control.* s.l. : The MIT Press, 1982.

3. **Kirchner H.O.K., Gurumoorthy B., Prinz F.B.** *A pertubation approach to robot calibration.* 1987, Vol. 6, págs. 47-59.

4. **K., Schroer.** Theory of kinematic modelling and numerical procedures for robot calibration. s.l. : Bernhardt R. and Albright S., 1993, págs. 157-193.

5. **Ibarra R., Perreira N.D.** Determination of linkage parameter and pair variable errors in open chain kinematic linkages using minimal set of pose measurement data. s.l. : Journal of Mechanisms, Transmissions, and Automation in Design, 1986, págs. 159-166.

6. **Hayati S.A., Mirmirani M.** Improving the absolute positioning accuracy of robot manipulators. s.l. : Journal of Robotics Systems, 1985, Vol. 2, págs. 397-413.

7. **Judd R.P., Knasinski A.B.** A technique to calibrate industrial robots with experimental verification. s.l. : Proceedings of IEEE International Conference of Robotics and Automation, 1987, págs. 351-357.

8. **Sugimoto K., Okada T.** Compensation of positioning errors caused by geometric deviations in the robot system. s.l. : Robotics Research: The Second International Symposium, 1985, págs. 231-236.

9. **Sklar, Michael E.** *Geometric calibration of industrial manipulators by Circle Point analysis.* F880, McDonnell Douglas Space Systems Co. págs. 198-202.

10. **The MathWorks Inc.** *MatLab Getting Started Guide.* 2010.

11. **MatLab.** MathWorks. [En línea] Noviembre de 2010. http://www.mathworks.com/.

12. **Wikipedia.** Wikipedia - Coordenadas Plücker. [En línea] http://en.wikipedia.org/wiki/Pl%C3%BCcker_coordinates.

13. **Mathworld.** Mathworld - Cálculos intersección de líneas. [En línea] http://mathworld.wolfram.com/Line-LineIntersection.html.

14. **De la Peña Hernández, Jesús.** *Máquinas tridimensionales de medir por coordenadas: Fundamentos geométricos y consideraciones prácticas.* Ingenieria de Calidad, I.C.A.I. s.l. : Comite de Metrología de la Asociación Española para el Control de la Calidad, 2006.

15. **Tilo Pfeifer, Fernando Torres.** Manual de gestión e ingenería de la calidad. s.l. : Mira Editores, 2002, págs. 185-207.

16. **Horning, Robert John.** *A comparison of identification techniques for robot calibration.* Electrical Engineering, Case Western Reserve University. 1998.

17. **Juneja, Nitin.** *Kinematic Calibration of a Reconfigurable Robot (RoboTwin).* Toronto : University of Toronto, 1996.

18. **Bruno Siciliano, Oussama Khatib.** *Springer Handbook of Robotics.* 2008.

19. **Horn, Berthold K.P.** Closed-form solution of absolute orientation using unit quaternions. Manoa, Honolulu, Hawai : Journal of the Optical Society of America, april 1987, Vol. 4, pág. 629.

20. **Hexagon Metrology.** Hexagon - Sistemas Láser Tracker. [En línea] Noviembre de 2010. http://www.hexagonmetrology.es/sistemas-laser-tracker_108.htm.

21. **Faro.** Láser Tracker Faro ION. [En línea] http://www.faro.com/lasertracker/es/whats-new/.

22. **Leica.** Láser Tracker Leica. [En línea] http://www.leica-geosystems.es/es/Productos-Sistemas-Laser-Tracker_69045.htm.

23. **Nikon Metrology.** Nikon - Sistemas Láser Tracker. [En línea] http://www.nikonmetrology.com/automotive/.

24. **International Organization for Standarization.** *Norma Internacional ISO 690: Documentación - Referencias bibliográficas: contenido, forma y estructura.* Ginebra : s.n., 1987.