MANIPULADORES MÓVILES:

MODELADO Y SIMULACIONES DE BRAZO ROBÓTICO MANFRED V3

Autores: Naiara Escudero Sánchez
Jose Pardeiro Blanco

Profesores: Dolores Blanco Rojas
Fernando Martín Monar

Índice

Índice de figuras

1. Introducción

La planificación de trayectorias y ejecución de las mismas en brazos robóticos es un tema que, a día de hoy, sigue muy activo. Por ello, el desarrollo de nuevos algoritmos o la mejora de algunos existentes se encuentra a la orden del día. Dichos algoritmos además tienen como fin ser implementados en sistemas reales sobre los que demostrar la funcionalidad del mismo.

Durante el periodo de desarrollo y testeo de nuevos algoritmos es muy usual la utilización de simuladores que faciliten comprobar los resultados del algoritmo sobre el robot modelado. Aunque los simuladores tratan casos ideales, o cercanos a los ideales, son válidos para comprobar la funcionalidad de los algoritmos y su correcto funcionamiento. De este modo se minimizan los riesgos para la integridad del brazo o las personas que se encuentren cerca del robot a la hora de probarlos en un sistema real.

Para ello existen multitud de herramientas de modelado de robots que facilitan la simulación del funcionamiento del mismo. A lo largo de la presente memoria se plantea el modelado del brazo robótico Manfred V3 en una herramienta para MATLAB llamada Robotics Toolbox, y los resultados al simular el modelado cinemático del mismo. Previamente se realiza un acercamiento teórico a los conceptos necesarios para comprender como se comporta, a nivel cinemático, el robot.

2. Fundamentos teóricos

En esta sección se detallan los fundamentos teóricos de los elementos más importantes que componen la solución propuesta. Con esta sección se pretende conseguir las bases teóricas necesarias para la comprensión del sistema descrito a lo largo de las siguientes secciones. Los fundamentos teóricos están basados, en su mayor parte, en el libro *Fundamentos de robótica*, escrito por Antonio Barrientos, Luis Felipe Peñín, Carlos Balaguer y Rafael Aracil, y el libro *Robotics, Vision and Control* escrito por Peter Corke.

2.1. Brazo robótico

Un brazo robótico es un dispositivo mecánico, programable y diseñado para realizar tareas como las que desempeña un brazo humano. Las partes que componen el brazo se conectan mediante articulaciones que permitan emular un movimiento tanto de desplazamiento como rotacional.

Estos brazos son muy utilizados en entornos industriales, ya que permiten la automatización de ciertas tareas con el fin de obtener unos resultados superiores en menor tiempo. Estos brazos han alcanzado un grado de robustez y fiabilidad bastante alto, y tienen un diseño que normalmente les permite manipular elementos de gran peso. Debido al tamaño de estos brazos, en paralelo se desarrollan otros modelos orientado a la robótica de servicios.

Los brazos para robots de servicios buscar tener una forma antropomórfica, destreza, seguridad ante golpes (evitar especialmente dañar a los humanos que se encuentren cerca en caso de accidente) mediante un brazo de bajo peso, flexible y de bajo consumo energético.

La forma antropomórfica requerida implica, de forma indirecta, que el brazo tenga 7 grados de libertad, lo cual aumenta la complejidad tanto a nivel mecánico como a nivel matemático. En la figura 1 se puede ver un ejemplo de brazo robótico en el cual quedan distribuidos los grados de libertad de la siguiente forma:

- 3 G.D.L. en el hombro.
- 2 G.D.L. en el codo.
- 2 G.D.L. en la muñeca.

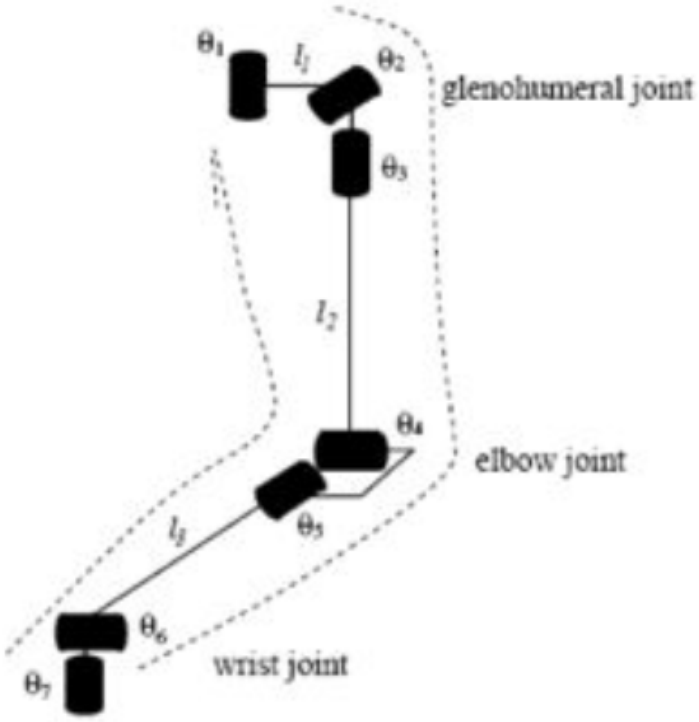


Figura 1: Ejemplo de brazo robótico.

Este tipo de brazos se encuentran muy extendidos entre los grupos de investigación de distintas empresas y Universidades. Al no existir un diseño estándar en estos momentos, en varios centros se ha optado por diseñar uno propio. En el caso de la Universidad Carlos III de Madrid se ha desarrollado desde hace varios años un brazo que recibe el nombre de Manfred, actualmente en su tercera versión.

Para el diseño de este brazo se ha establecido como necesidad que fuera lo más ligero posible sin llegar a perder su robustez y rigidez y distribuido homogéneamente. Con ello se disminuye el consumo, su momento de inercia (gracias a lo cual se reduce el posible daño al impactar con una persona), se optimiza la estabilidad del manipulador y se aumenta el ratio de carga y peso.

El brazo, además, debe poseer un diseño antropomórfico, es decir, ser capaz de realizar los movimientos de una forma cercana a como los harían los seres humanos, incluyendo la cinemática. Además debe integrar distintos sistemas mecánicos, eléctricos y sensoriales.

2.2. Cinemática del robot

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Para ello se necesita una función que describa analíticamente el movimiento espacial del robot en función del tiempo, y particularmente las relaciones entre la posición y la orientación del extremo final del robot con los que toman sus coordenadas articulares.

Dentro de la cinemática existen dos problemas fundamentales para resolver, basados en las relaciones entre la posición y orientación real del robot en el espacio y sus rotaciones articulares. El primero, conocido como cinemática directa, consiste en determinar cual es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas dados, conocidos los valores de rotación de las articulaciones. El segundo se conoce como cinemática inversa y consiste en justo lo contrario, conocer cuales son las rotaciones articulares necesarias para que el extremo alcance una posición y orientación determinada.

A nivel cinemático también se busca resolver las relaciones entre las velocidades de movimiento de las articulaciones y las del extremo. Este modelo queda descrito mediante un modelo diferencial utilizando la matriz Jacobiana.

Para poder desarrollar la cinemática es preciso, en primera instancia, definir la geometría del robot. Para ello, Denavit y Hartenberg propusieron un método sistemático en el que se describen todos los elementos del robot sobre un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea que describe la relación espacial entre dos elementos rígidos adyacentes.

2.2.1. Algoritmo de Denavit-Hartenberg

Denavit-Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas (S_i) ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

Según la representación D-H, escogiendo adecuadamente los sistemas de coordenadas asociados para cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón. De este modo será posible pasar de un eslabón al siguiente mediante cuatro transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón. Las transformaciones constan de dos traslaciones y dos rotaciones efectuadas del siguiente modo:

1. Rotación sobre el eje z_{i-1} un ángulo θ_i.
2. Traslación a lo largo de z_{i-1} una distancia d_i.
3. Traslación a lo largo de x_i una distancia a_i.
4. Rotación sobre el eje x_i un ángulo α_i.

Los cuatro parámetros de D-H (θ_i, a_i, d_i, α_i) dependen exclusivamente de las características geométricas de cada eslabón y de las articulaciones que unen cada uno con el siguiente. Estos representan:

- θ_i: Es el ángulo que forman los ejes x_{i-1} y x_i medido en un plano perpendicular al eje z_i, utilizando la regla de la mano derecha. En una articulación móvil será un parámetro variable.

- a_i: Es la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas se define como la distancia más corta entre los ejes z_{i-1} y z_i.

- d_i: Es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas $(i-1)$-ésimo hasta la intersección del eje z_{i-1} con el eje x_i. En articulaciones prismáticas será un parámetro variable.

- α_i: Es el ángulo de separación del eje z_{i-1} y el eje z_i, medido en un plano perpendicular al eje x_i utilizando la regla de la mano derecha.

Para calcular los parámetros D-H de forma sistemática existe un listado con 16 pasos que se han de seguir en orden. De este modo se obtendrá una tabla con la definición geométrica de un robot. Los pasos son los siguientes:

- **DH1**. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (ultimo eslabón móvil). Se numerara como eslabón 0 a la base fija del robot.

- **DH2**. Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.

- **DH3**. Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

- **DH4**. Para i de 0 a $n-1$, situar el eje z_i, sobre el eje de la articulación $i+1$.

- **DH5**. Situar el origen del sistema de la base (S_0) en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situaran dé modo que formen un sistema dextrógiro con Z0.

- **DH6**.Para i de 1 a $n-1$, situar el sistema (S_i) (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría (S_i) en el punto de corte. Si fuesen paralelos (S_i) se situaría en la articulación $i+1$.

- **DH7**. Situar x_i en la línea normal común a z_{i-1} y z_i.

- **DH8**. Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.

- **DH9**. Situar el sistema (S_n) en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.

- **DH10**. Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.

- **DH11**. Obtener d_i como la distancia, medida a lo largo de z_{i-1}, que habría que desplazar (S_{i-1}) para que x_i y x_{i-1} quedasen alineados.

- **DH12**. Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo (S_{i-1}) para que su origen coincidiese con (S_i).

- **DH13**. Obtener a_i como el ángulo que habría que girar entorno a x_i (que ahora coincidiría con x_{i-1}), para que el nuevo (S_{i-1}) coincidiese totalmente con (S_i).
- **DH14**. Obtener las matrices de transformación ${}^{i-1}A_i$.
- **DH15**. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^0A_i,\ 1^A_2 ...\ {}^{n-1}A_n$.
- **DH16**. La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido ala base en función de las n coordenadas articulares.

Como ejemplo, una vez realizados todos los pasos sobre el brazo de la figura 2, queda como resultado la tabla 1.

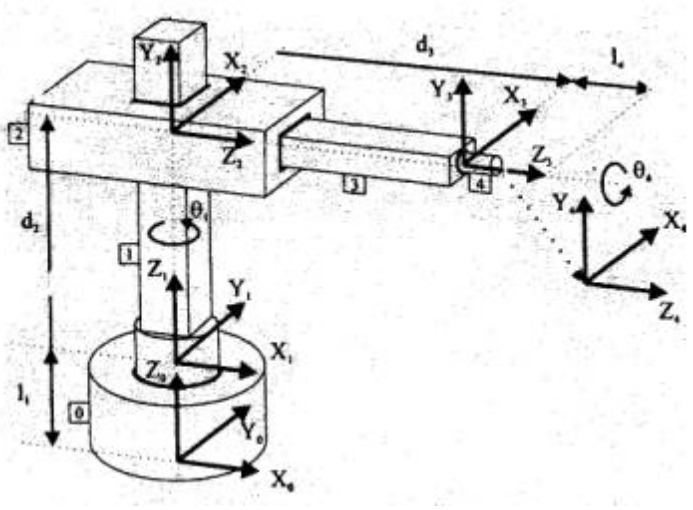


Figura 2: Ejemplo de D-H sobre brazo robótico.

Articulación	θ	d	a	α
1	q_1	l_1	0	0
2	$\frac{\pi}{2}$	d_2	0	$\frac{\pi}{2}$
3	0	d_3	0	0
4	q_4	l_4	0	0

Tabla 1: Parámetros D-H del robot de la figura 2.

2.2.2. Cinemática directa

La resolución de la cinemática directa se fundamenta principalmente en el uso de álgebra vectorial y matricial para la descripción de un objeto en el espacio tridimensional respecto a un eje de referencia fijo. Un robot puede ser considerado una cadena cinemática formada por eslabones unidos mediante articulaciones, por lo que usualmente el sistema de referencia fijo se sitúa en la base, dejando el resto de eslabones localizados respecto a dicha base. Dicho esto, se puede decir que el

problema de la cinemática directa se puede resumir a encontrar una matriz de transformación T, en función de las coordenadas articulares, que relacione la posición y la orientación del extremo del robot respecto a un sistema de referencia fijo.

Generalmente un robot de n grados de libertad está formado por n eslabones unidos por n articulaciones, de forma que cada articulación-eslabón constituye un grado de libertad. A cada eslabón se le puede asociar un sistema de referencia propio y, utilizando las transformaciones homogéneas, se pueden representar las rotaciones y traslaciones relativas entre los distintos eslabones que forman el robot. Generalmente, la matriz de transformación homogénea que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se suele denominar matriz ${}^{i-1}A_i$. De este modo, ${}^{0}A_1$ describe la posición y orientación del sistema de referencia propio del primer eslabón con respecto al sistema de referencia de la base, y de este modo sucesivamente hasta completar el robot.

Aunque para la descripción de la relación que existe entre dos elementos contiguos puede hacerse siguiendo cualquier sistema de referencia ligado a cada elemento, la forma estándar utilizada consiste en el uso del algoritmo de Denavit-Hartenberg, descrito en la sección 2.2.1.

De este modo, dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. Por tanto se obtiene la ecuación descrita en la ecuación 1.

$$ {}^{i-1}A_i = T(z,\theta_i)T(0,0,d_i)T(a_i,0,0)T(x,\alpha_i) \tag{1} $$

Desarrollando la ecuación 1 obtenemos la ecuación 2.

$$ {}^{i-1}A_i = \begin{vmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = $$

$$ \begin{vmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i C\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2} $$

donde θ_i, a_i, d_i, α_i son parámetros D-H del eslabón i. De este modo es suficiente con identificar dichos parámetros y así obtener la matriz A que vaya relacionando cada uno de los eslabones del robot.

2.2.3. Cinemática inversa

El objetivo de la cinemática inversa consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot q=[q_1, q_2, ..., q_n] para que su extremo

se posicione en una determinada localización espacial con una orientación definida.

Mientras que la cinemática directa se puede abordar de forma sistemática utilizando matrices de transformación homogénea de forma independiente a la configuración del robot, no ocurre lo mismo con la cinemática inversa que si depende fuertemente de la configuración del robot.

Aunque se han planteado diversos métodos para resolver de forma genérica la cinemática inversa de un robot, estos métodos son iterativos por lo que la velocidad de convergencia, o incluso la posibilidad de encontrar una solución posible, no está garantizada. Este problema se ve acentuando a medida que aumentan los grados de libertad de un robot, ya que la cinemática inversa pasa de tener una solución posible a varias.

Por ello la solución más óptima consiste en encontrar una solución matemática que defina a tu robot y permita encontrar de forma directa la cinemática inversa del robot. Este sistema trae consigo varias ventajas, principalmente la velocidad de cómputo, imprescindible en sistemas que necesitan solución en tiempo real y la posibilidad de imponer reglas para obtener la solución que mejor se adapte a la trayectoria de todas las posibles. Dependiendo de la dificultad del problema se puede resolver mediante métodos geométricos o a partir de la matriz de transformación homogénea.

No obstante, como se ha comentado anteriormente, es necesario encontrar una forma matemática de expresar la cinemática inversa del robot, y eso no siempre se puede lograr. Este proceso es dependiente de la configuración del robot y el número de grados de libertad del mismo, por lo que en determinados casos esta solución no es posible. Para ello se utiliza la matriz Jacobiana descrita en la sección 2.2.4.

2.2.4. Matriz Jacobiana

Como se ha visto en las secciones anteriores, el modelado cinemático de un robot busca las relaciones entre las variables articulares y la posición y orientación del extremo del robot. Pese a que en este modelo no se tiene en cuenta el par de fuerzas que actúa sobre el robot, sí que de debe ser capaz de conocer su relación de velocidades, en forma de las derivadas de posición, orientación y variables articulares, para realizar la trayectoria buscada.

Para estos fines se utiliza la matriz Jacobiana, encargada de relacionar las velocidades articulares del robot y las de posición y orientación del mismo, por tanto, permite conocer las velocidades del extremo del robot conociendo las de cada articulación. Por tanto, la matriz Jacobiana inversa permite conocer las velocidades articulares necesarias para lograr las deseadas en el extremo del robot.

El método más directo para la obtención de la relación entre velocidades articulares y del extremo del robot consiste en derivar las ecuaciones correspondientes al modelo dinemático directo. Así, se parte de la ecuación 3 y se deriva tal y como se muestra en 4.

$$\begin{aligned} x = f_x(q_1, ..., q_n) \quad y = f_y(q_1, ..., q_n) \quad z = f_z(q_1, ..., q_n) \\ \alpha = f_\alpha(q_1, ..., q_n) \quad \beta = f_\beta(q_1, ..., q_n) \quad \gamma = f_\gamma(q_1, ..., q_n) \end{aligned} \tag{3}$$

$$\begin{aligned} \dot{x} = \sum_1^n \tfrac{\delta f_x}{\delta q_i}\dot{q}_i \quad \dot{y} = \sum_1^n \tfrac{\delta f_y}{\delta q_i}\dot{q}_i \quad \dot{z} = \sum_1^n \tfrac{\delta f_z}{\delta q_i}\dot{q}_i \\ \dot{\alpha} = \sum_1^n \tfrac{\delta f_\alpha}{\delta q_i}\dot{q}_i \quad \dot{\beta} = \sum_1^n \tfrac{\delta f_\beta}{\delta q_i}\dot{q}_i \quad \dot{\gamma} = \sum_1^n \tfrac{\delta f_\gamma}{\delta q_i}\dot{q}_i \end{aligned} \tag{4}$$

Si las ecuaciones 3 y 4 se expresan en forma matricial queda la ecuación 5

$$\begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{vmatrix} = J \begin{vmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{vmatrix} \tag{5}$$

donde J es la matriz Jacobiana definida como 6.

$$J = \begin{vmatrix} \frac{\delta f_x}{\delta q_1} & \cdots & \frac{\delta f_x}{\delta q_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_\gamma}{\delta q_1} & \cdots & \frac{\delta f_\gamma}{\delta q_n} \end{vmatrix} \tag{6}$$

Dado que el valor numérico de los valores que componen J variará en función de los valores instantáneos de las coordenadas articulares θ_i, el valor de la Jacobiana variará en cada punto del espacio.

De igual modo que la Jacobiana proporciona las velocidades en el extremo del robot conociendo las velocidades articulares, utilizando la matriz Jacobiana inversa se pueden obtener las velocidades articulares necesarias para lograr unas velocidades determinadas en el extremo, tal y como plantea la ecuación 7.

$$\begin{vmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{vmatrix} = J^{-1} \begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{vmatrix} \tag{7}$$

La matriz Jacobiana inversa puede obtenerse de dos métodos diferentes, el primero consistente en realizar la inversa de la matriz J original, y la segunda de una forma análoga a la Jacobiana directa. Si se opta por invertir la matriz Jacobiana, en caso de que ésta sea cuadrada es tan sencillo como calcular la matriz inversa. En caso contrario sería necesario utilizar técnicas para cálculo de matrices pseudoinversas, siendo el método de Moore-Penrose el más habitual para este tipo de soluciones. Finalmente, si se opta por calcular la matriz J^{-1} directamente, el resultado quedará de la forma que describe la ecuación 8.

$$J^{-1} = \begin{vmatrix} \frac{\delta f_1}{\delta x} & \cdots & \frac{\delta f_n}{\delta \gamma} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_n}{\delta x} & \cdots & \frac{\delta f_n}{\delta \gamma} \end{vmatrix} \tag{8}$$

Del mismo modo que utilizando la matriz Jacobiana inversa se puede obtener la relación de velocidades, también es posible obtener la relación entre rotaciones articulares y posición y orientación, tal y como se describe en la ecuación 9. De este modo queda resuelto el problema de la cinemática inversa de una forma independiente al número de grados de libertad y la dificultad del robot.

$$q^{i+1} = q^i + J^{-1}(r^{i+1} - r^i) \tag{9}$$

dónde

$$r = \begin{vmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{vmatrix} \quad q = \begin{vmatrix} q_1 \\ \vdots \\ \vdots \\ \vdots \\ q_n \end{vmatrix}$$

2.2.5. Robotics Toolbox

Robotics Toolbox es un conjunto de herramientas, o toolbox, desarrolladas por Peter Corke para el software matemático MATLAB. Aunque las herramientas son totalmente libres, se requiere de MATLAB para ejecutarlas, que es propietario.

La toolbox proporciona funciones para operaciones de manipulación y conversión entre tipos de datos, como vectores, transformaciones homogéneas, ángulos roll, pith y yaw, ángulos de Euler u operaciones con quaterniones.

La toolbox además contiene herramientas necesarias para analizar la cinemática, la dinámica y la generación de trayectorias de brazos robóticos. Estas herramientas contienen funciones para la definición del robot, cálculos cinemáticos, dinámicos o visualización de movimientos.

También incluye herramientas para trabajar con robots terrestres,incluyendo planificación de trayectorias con métodos como RRT, localización utilizando filtros de Kalman o de partículas, generación de mapas o localización simultánea a la generación del mapa. Finalmente se han integrado herramientas para trabajar con vehículos aéreos de cuatro rotores.

3. Implementación del sistema

En esta sección se detallarán los elementos necesarios para poder modelar el brazo robótico Manfred V3, desarrollado en la Universidad Carlos III de Madrid, en la herramienta Robotics Toolbox. En la figura 3 se muestra un diagrama de flujo con el funcionamiento del sistema.

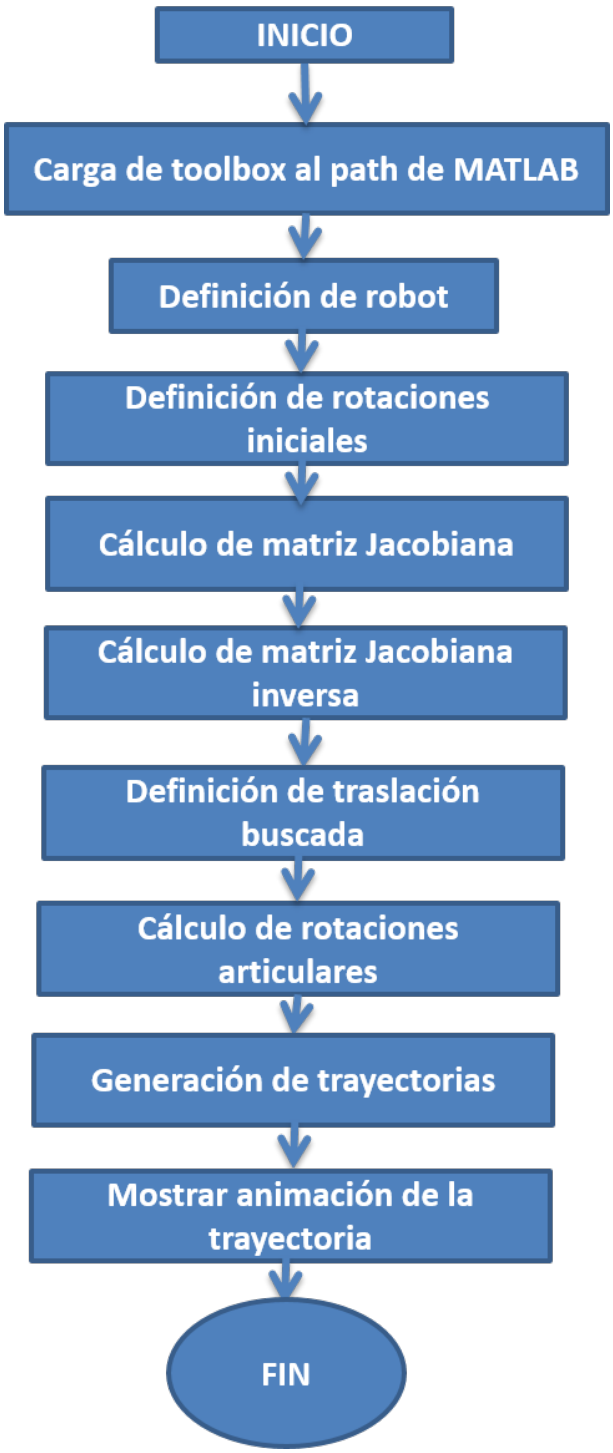


Figura 3: Diagrama de flujo de la implementación.

3.1. Aplicación desarrollada

La implementación en la Robotics Toolbox del problema de cinemática sigue un desarrollo muy próximo al teórico explicado en los fundamentos teóricos. Para ello es preciso modelar el robot según los parámetros Denavit-Hartenberg expuestos en la sección 3.2 y a continuación desarrollar la cinemática.

Debido al número de grados de libertad del robot Manfred, 7, la cinemática inversa calculada directamente no es posible al no encontrar ninguna solución adecuada en el límite de iteraciones que la propia toolbox plantea. Por ese motivo se trabajará con cinemática inversa utilizando la matriz Jacobiana inversa descrita en la sección 2.2.4.

Para ello, la toolbox permite calcular la Jacobiana en el instante en que se tiene una rotación articular concreta, que se corresponderá con el punto de inicio del robot, ya que como queda reflejado en la sección 2.2.4 la matriz Jacobiana depende de la rotación angular en cada instante. Una vez calculada la matriz Jacobiana se invierte para obtener la matriz JAcobiana inversa. Dado que la matriz no será cuadrada al tener 7 grados de libertad, sino que tendrá unas dimensiones de 6x7, la matriz no es invertible y por tanto habrá que invertirla usando el método de la pseudoinversa de Moore-Penrose.

A continuación se calcularán las rotaciones articulares para llegar de un punto concreto a otro indicado, utilizando la fórmula 9. Finalmente se calcula la trayectoria para conectar ambos puntos utilizando las rotaciones articulares de inicio y fin y se muestra una animación con el resultado.

Es necesario puntualizar que en caso de querer realizar una traslación y/o rotación grande es recomendable discretizar el movimiento en varios puntos más cercanos entre sí para así obtener unos mejores resultados al calcular la cinemática inversa.

A continuación se van a explicar las funciones utilizadas para una traslación pequeña, dividiendo el código en distintas funcionalidades.

3.1.1. Definición del robot

El primero paso consiste en definir el robot según los parámetros de Denavit-Hartenberg mostrados en la tabla 2. Para ello se utiliza la función *Link* para definir cada eslabón, y la función *SerialLink* para generar el robot.

La función *Link* se define del siguiente modo:

$$\text{L=Link([}\theta\ d\ a\ \alpha\text{]);}$$

De este modo, los eslabones del robot quedarán definidos del siguiente modo:

$$\begin{gathered}\text{L(1)=Link([}\pi/2\ 228\ 0\ \pi/2\text{]);}\\ \text{L(2)=Link([pi/2 0 0 pi/2]);}\\ \text{L(3)=Link([-}\pi/2\ 312.5\ 0\ -\pi/2\text{]);}\\ \text{L(4)=Link([}\pi\ 0\ 0\ -\pi/2\text{]);}\end{gathered}$$

L(5)=Link([0 278 0 π/2]);
L(6)=Link([0 0 0 -π/2]);
L(7)=Link([0 180 0 π/2]);

Una vez definidos los eslabones se ensambla el brazo utilizando la función *SerialLink*, definida de la siguiente forma:

robot=SerialLink(L, 'opción', 'argumento');

De modo que, asociado a nuestro robot, quedaría:

manfred=SerialLink(L, 'name', 'manfredv3');

Con esto queda el robot definido. La figura 4 muestra el resultado del robot con todas las rotaciones articulares con valor 0.

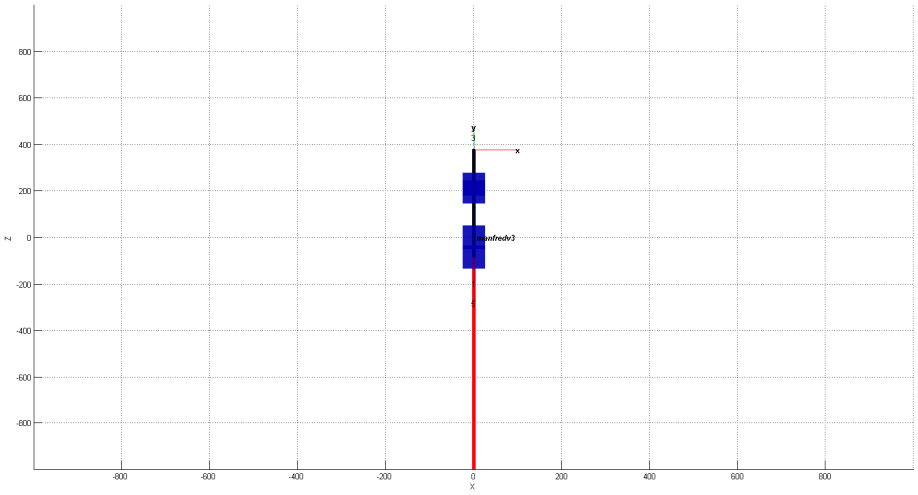


Figura 4: Manfred V3 definido en Robotics Toolbox.

3.1.2. Definición de punto inicial y matriz Jacobiana inversa

Una vez definido el robot en la Robotics Toolbox, comienza el proceso de definición del punto de inicio. Para ello se establecen las rotaciones articulares iniciales como un vector de 7 posiciones, una por grado de libertad. Para este ejemplo se ha usado el siguiente, quedando como resultado la figura 5.

qn = [0, π/2, 0, π, π/2, π/4, 0];

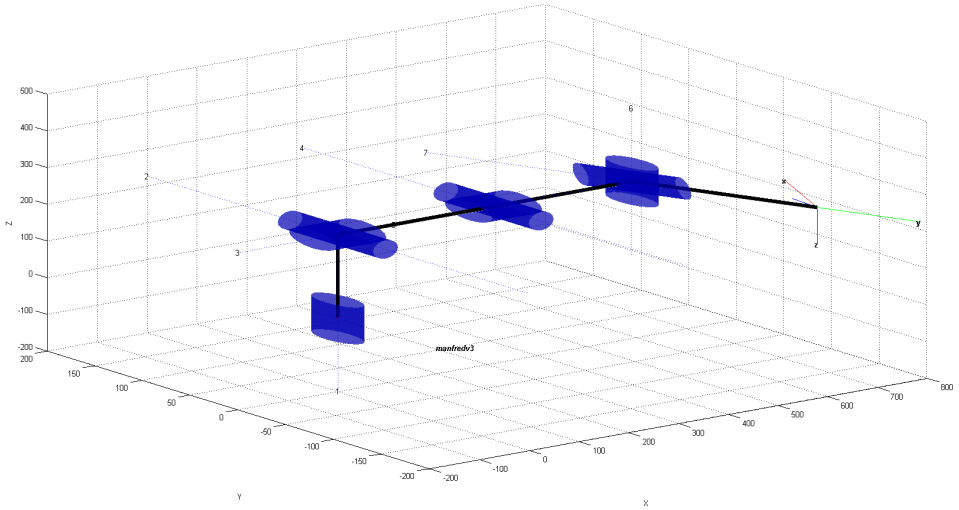


Figura 5: Manfred V3 en posición inicial.

Una vez se ha definido el vector de rotaciones articulares iniciales se puede calcular la Jacobiana y la Jacobiana inversa del siguiente modo

jacobian=manfred.jacob0(qn);
jacobianInv=pinv(jacobian);

3.1.3. Generación de trayectoria y muestra de resultados

Con las matriz Jacobiana inversa calculada en el punto anterior, el siguiente paso consiste en calcular las rotaciones articulares necesarias para alcanzar la posición final, utilizando la fórmula 9. Dicha ecuación, pasada a la nomenclatura utilizada en este ejemplo, quedaría tal que:

qr=qn+jacobianInv*T;

Siendo T la traslación en los tres ejes, tanto longitudinal como angular, buscada. De este modo, T puede ser definido como:

$$\mathrm{T}=(p_1 - p_0);$$

De este modo, el vector T puede ser rellenado directamente con los incrementos buscados, o utilizando la diferencia entre el punto inicial y final. El punto inicial, conociendo las rotaciones articulares qn, se puede obtener de la matriz de transformación homogénea que proporciona la toolbox del siguiente modo:

p_0=manfred.fkine(qn);

Una vez calculadas las rotaciones articulares finales, qr, se puede generar la trayectoria enumerando el número de pasos en los que se busca llegar al punto, n, de la siguiente forma:

n=0:1;
traj1=jtraj(qn,qr,n);

Esta trayectoria puede ser mostrada en forma de animación utilizando la función *plot* integrada en la toolbox, mostrado en la figura 6, de la siguiente forma:

```
manfred.plot(traj1);
```

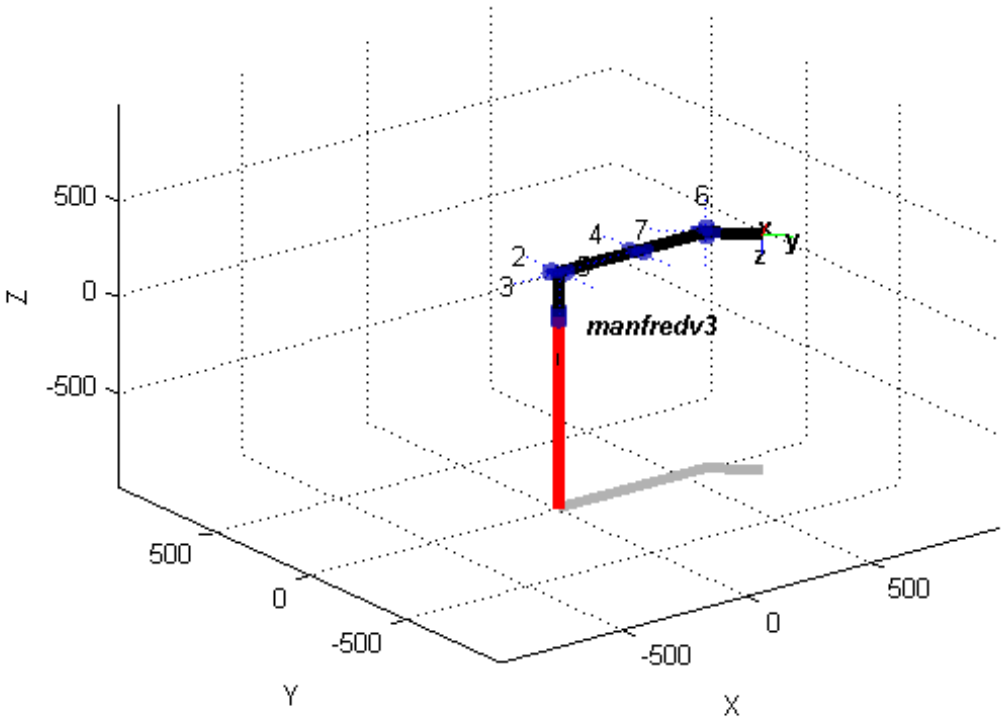
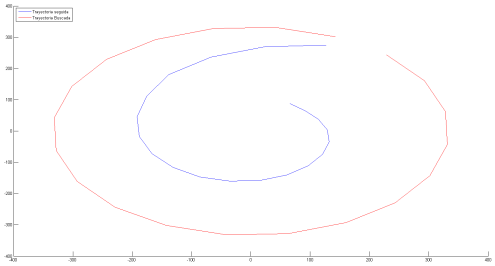


Figura 6: Manfred V3 realizando la ruta.

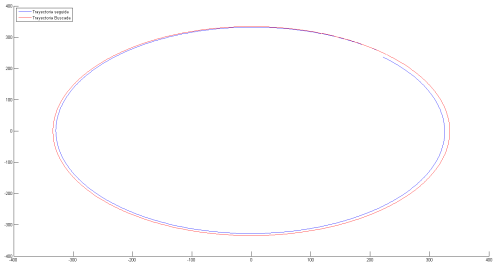
3.1.4. Consideraciones en la implementación

Como se comentó en la sección 2.2.4, el valor de la matriz Jacobiana varía en función de la posición del robot. Por ese motivo, es complejo calcular la cinemática inversa para llegar a un punto lejano al original. Para solucionar dicho problema, la distancia es discretizada en puntos intermedios de menor distancia que, finalmente, acaben llevando al objetivo.

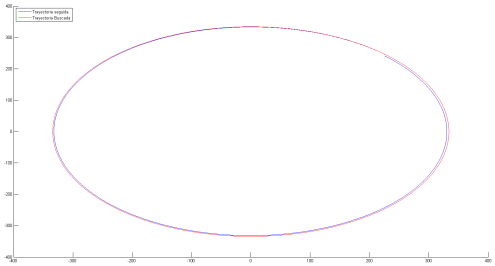
Como ejemplo se utilizará una trayectoria circular. En la figura 7 se mostrarán los resultados en función del número de puntos intermedios. En la figura 7a se ha discretizado con 20 puntos. Mientras, en la figura 7b se han utilizado 800, en la 7c, 2000 y finalmente en la figura 7d se han utilizado 6000. Como se puede apreciar, a mayor número de puntos intermedios se obtiene una trayectoria con mayor exactitud a la requerida.



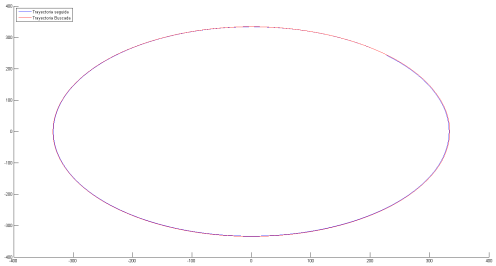
(a) Trayectoria con 20 puntos.



(b) Trayectoria con 800 puntos.



(c) Trayectoria con 2000 puntos.



(d) Trayectoria con 6000 puntos.

Figura 7: Trayectorias circulares.

3.2. Parámetros Denavit-Hartenberg

El elemento imprescindible para modelar al brazo se corresponde a los parámetros Denavit-Hartenberg que definen el robot. Estos datos han sido proporcionados por los responsables del diseño del brazo, y están definidos en la tabla 2. El diseño del robot se representa en la figura 8.

Articulación	θ	d	a	α
1	$\frac{\pi}{2} + q_1$	228	0	$\frac{\pi}{2}$
2	$\frac{\pi}{2} + q_2$	0	0	$\frac{\pi}{2}$
3	$-\frac{\pi}{2} + q_3$	312,5	0	$-\frac{\pi}{2}$
4	$\pi + q_4$	0	0	$-\frac{\pi}{2}$
5	q_5	278	0	$\frac{\pi}{2}$
6	q_6	0	0	$-\frac{\pi}{2}$
7	q_7	180	0	$\frac{\pi}{2}$

Tabla 2: Parámetros D-H del robot Manfred V3.

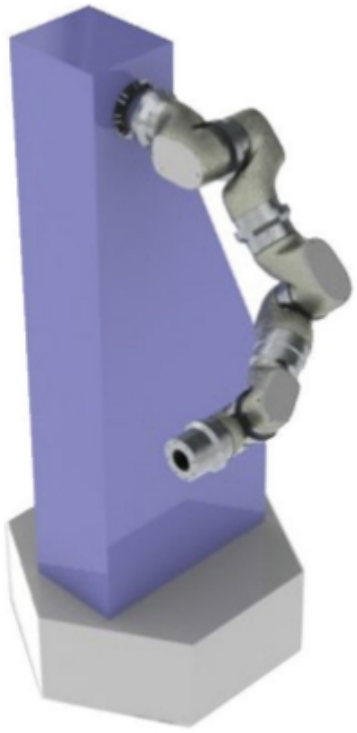


Figura 8: Diseño de Manfred V3.

Una vez introducido el robot en la toolbox, como queda el robot con el brazo extendido se define en la figura 9.

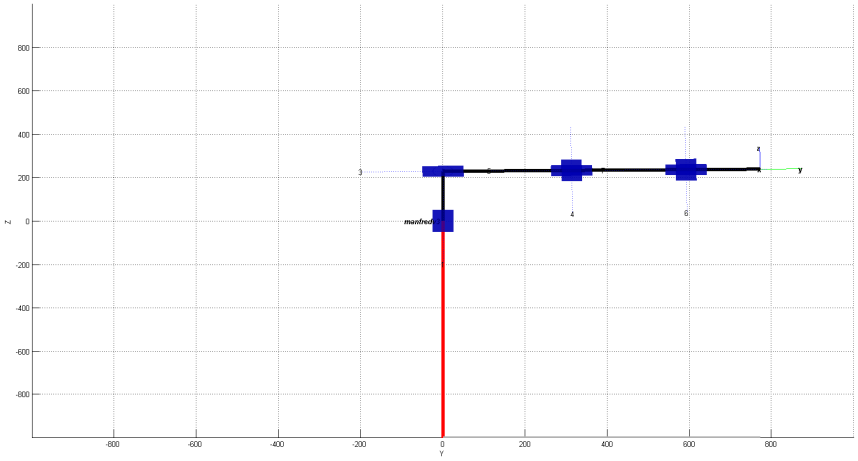


Figura 9: Visualización de Manfred V3 en Robotics Toolbox.

4. Pruebas experimentales

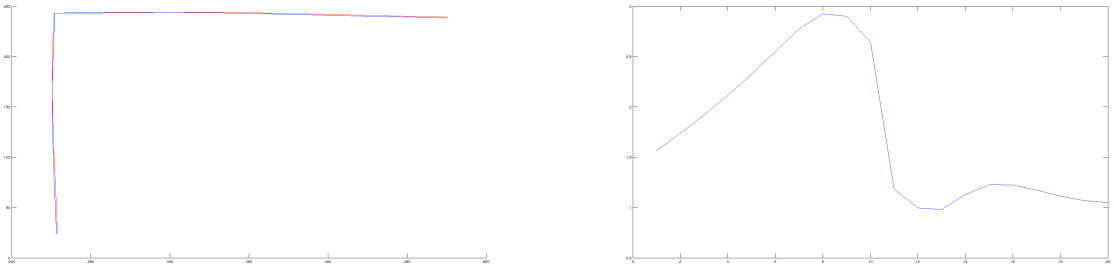
En esta sección se describirán los resultados de la cinemática al aplicar el sistema descrito en la sección 3 sobre tres tipos de trayectorias. La primera será una trayectoria lineal en la que solo se tendrá en cuenta una traslación longitudinal. La segunda realizará una trayectoria circular, añadiendo al desplazamiento longitudinal la rotación la herramienta a la vez que se forma el circulo. Finalmente se realiza una rotación en los tres ejes sobre un mismo punto, sin realizar desplazamiento longitudinal. En todos los casos se mostrarán los resultados y los errores en función del grado de discretización de la trayectoria. El error se calcula según la ecuación 10.

$$error = \sqrt{(p_{1x} - p_{0x})^2 + (p_{1y} - p_{0y})^2 + (p_{1z} - p_{0z})^2} \tag{10}$$

4.1. Trayectoria lineal

En esta primera prueba el robot realizará una trayectoria lineal desde su punto de origen a uno indicado, siguiendo una forma especificada. La prueba consiste en realizar la misma trayectoria discretizada en distinta cantidad de puntos.

En el primer caso se ha discretizado con 20 puntos, y el resultado puede verse en la figura 10a, siendo la línea azul la realizada por el robot y la roja la deseada. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 10b. Como se puede ver el error oscila entre 1 y 3 milímetros, lo cual podría ser suficiente en según que aplicaciones.

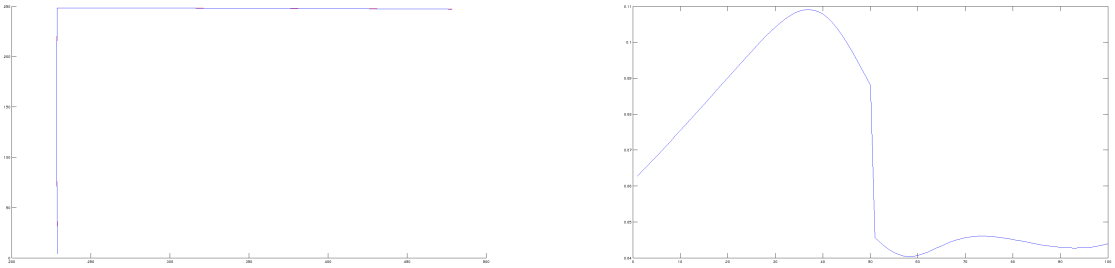


(a) Trayectoria con 20 puntos.

(b) Error en la trayectoria con 20 puntos.

Figura 10: Trayectoria lineal con 20 puntos.

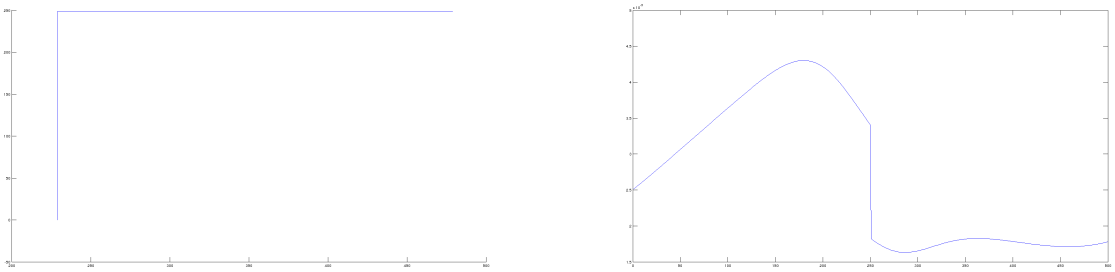
En el segundo caso se ha discretizado con 100 puntos, con un resultado mostrado en la figura 11a, siendo la línea azul la realizada por el robot y la roja la deseada. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 11b. Como se puede ver el error ha disminuido notablemente y oscila entre 0.04 y 0.11 milímetros, obteniendo así una precisión mucho mayor.



(a) Trayectoria con 100 puntos. (b) Error en la trayectoria con 100 puntos.

Figura 11: Trayectoria lineal con 100 puntos.

En este último caso se ha discretizado con 500 puntos, con un resultado mostrado en la figura 12a, siendo la línea azul la realizada por el robot y la roja la deseada. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 12b. En este último caso el error entre 0.0015 y 0.005 milímetros, es decir, una precisión mucho mayor a la del caso de 100 puntos.



(a) Trayectoria con 500 puntos. (b) Error en la trayectoria con 500 puntos.

Figura 12: Trayectoria lineal con 500 puntos.

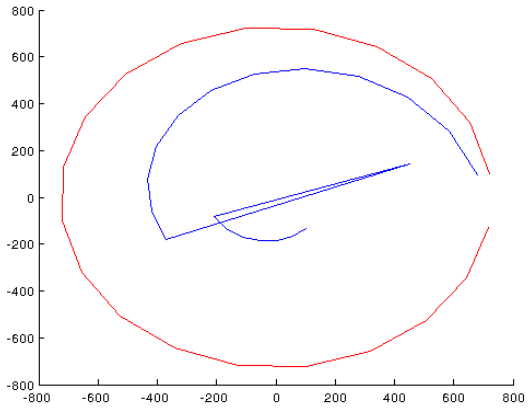
Como conclusión final, queda demostrado que un mayor número de puntos al describir una trayectoria lineal ofrece una precisión mayor a la hora de que el robot reproduzca dicha trayectoria. El pico de mayor aerror además se produce en el momento del cambio de dirección de desplazamiento del robot, generando en los tres casos una curva con forma similar aunque en distinta escala.

4.2. Trayectoria circular

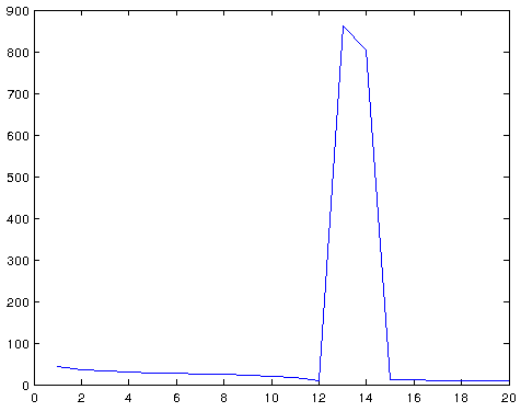
Durante esta segunda prueba el robot realizará una trayectoria circular completa desde su punto de origen a uno indicado, variando no solo la posición geométrica, sino también la orientación de la herramienta a lo largo del eje z. La prueba consiste en realizar la misma trayectoria discretizada en distinta cantidad de puntos.

En la primera iteración se ha discretizado con 20 puntos, y el resultado puede verse en la figura 13a, siendo la línea azul la realizada por el robot y la roja la deseada. Como se puede ver la trayectoria hacia la mitad de la misma se pierde y el resultado queda muy separado al deseado. El error está calculado acorde a la

ecuación 10, y mostrado gráficamente en la figura 13b. Como se puede ver el error oscila entre 0 y casi 900 milímetros, lo cual implica una inexactitud muy grande.



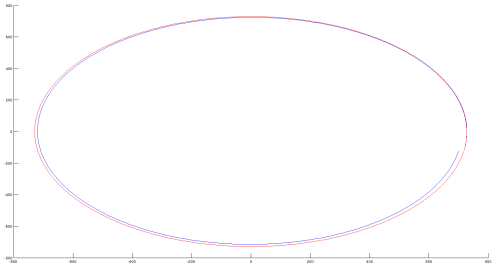
(a) Trayectoria con 20 puntos.



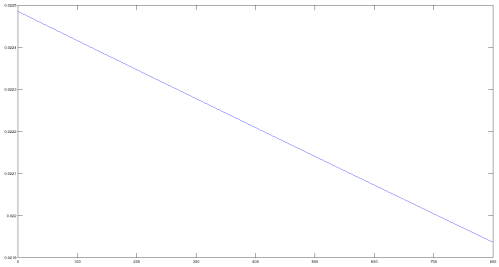
(b) Error en la trayectoria con 20 puntos.

Figura 13: Trayectoria circular con 20 puntos.

Para la segunda iteración se ha discretizado con 800 puntos, una cifra muy superior a la anterior, y el resultado puede verse en la figura 14a, siendo la línea azul la realizada por el robot y la roja la deseada. Como se puede ver la trayectoria en este caso sigue la misma forma que la deseada pese a ligeros desplazamientos. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 14b. Como se puede ver el error oscila entre 0.0219 y 0.0225 milímetros, por lo que la precisión ha aumentado notablemente.



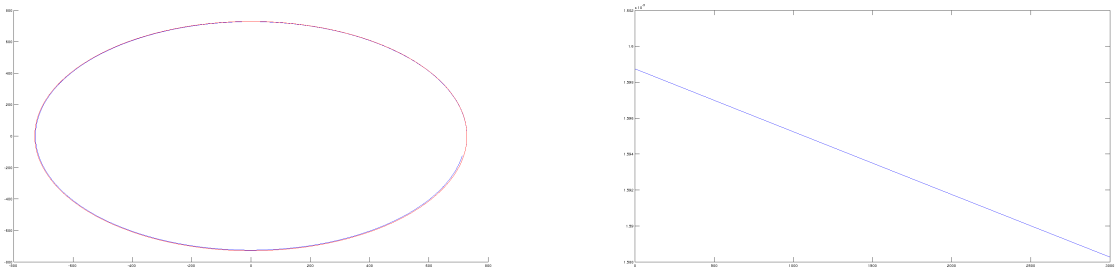
(a) Trayectoria con 800 puntos.



(b) Error en la trayectoria con 800 puntos.

Figura 14: Trayectoria circular con 800 puntos.

Finalmente, para la tercera y última iteración se ha discretizado con 3000 puntos, de nuevo una cifra muy superior a la anterior, y el resultado puede verse en la figura 15a, siendo la línea azul la realizada por el robot y la roja la deseada. Como se puede ver la trayectoria es incluso más cercana a la real que en el caso anterior. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 15b. Como se puede ver el error oscila entre 0.001588 y 0.001599 milímetros, por lo que la precisión ha vuelto a aumentar con respecto a la anterior.



(a) Trayectoria con 3000 puntos. (b) Error en la trayectoria con 3000 puntos.

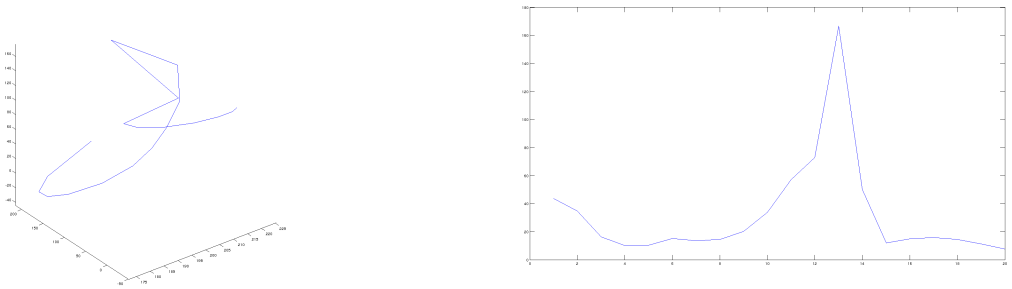
Figura 15: Trayectoria circular con 3000 puntos.

En este caso, debido a la complejidad de la trayectoria buscada y el tamaño de la misma es necesario un mayor número de puntos para lograr una trayectoria suficientemente precisa. El caso de 20 puntos queda completamente descartado ya que no es capaz de realizar una trayectoria aproximada a la deseada, mientras que con 800 y 3000 puntos si lo logra. Entre ambos casos, el factor que marca la elección es la precisión deseada.

4.3. Trayectoria rotacional

En la última prueba realizada, la trayectoria a seguir por el robot consiste en un momento rotacional sobre un mismo punto, variando únicamente la orientación de extremo del mismo. La prueba consiste en realizar la misma trayectoria discretizada en distinta cantidad de puntos.

Durante el primer ensayo se ha discretizado con 20 puntos, y el resultado puede verse en la figura 16a, siendo, como anteriormente, la línea azul la realizada por el robot y la roja la deseada. Como se puede observar el robot si que se mueve del punto fijo, con resplazamientos de entre 150 y 200 milímetros en cada eje. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 16b. Como se puede ver el error oscila entre 10 y 170 milímetros, conllevando una inexactitud muy grande.

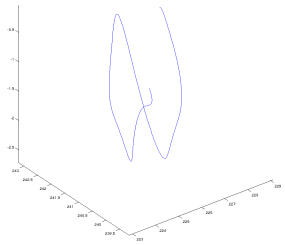


(a) Trayectoria con 20 puntos. (b) Error en la trayectoria con 20 puntos.

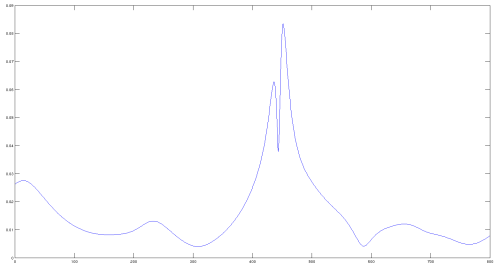
Figura 16: Trayectoria rotacional con 20 puntos.

Durante el segundo ensayo se ha optado por discretizar la trayectoria en 800 puntos, y el resultado puede verse en la figura 17a, siendo, de nuevo, la línea

azul la realizada por el robot y la roja la deseada. Como se puede observar el robot también se mueve con respecto al punto fijo, pero los desplazamientos son mucho menores, de aproximadamente 2 milímetros en el peor de los casos. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 17b. Como se puede ver el error oscila entre 0 y 0.9 milímetros, por lo que la precisión se ha visto aumentada notablemente de nuevo.



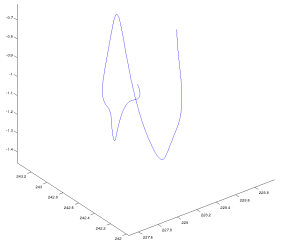
(a) Trayectoria con 800 puntos.



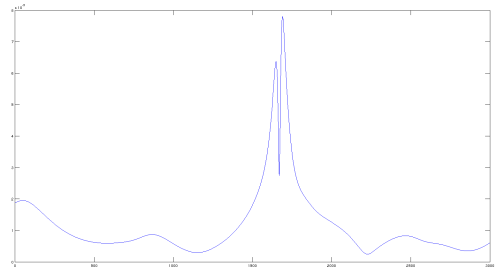
(b) Error en la trayectoria con 800 puntos.

Figura 17: Trayectoria rotacional con 800 puntos.

Finalmente, durante el tercer ensayo se ha optado por discretizar la trayectoria en 3000 puntos, y el resultado puede verse en la figura 18a, siendo, de nuevo, la línea azul la realizada por el robot y la roja la deseada. Como se puede observar el robot se vuelve a mover con respecto al punto fijo, pero los desplazamientos son mucho menores, de aproximadamente 0.6 milímetros en el peor de los casos. El error está calculado acorde a la ecuación 10, y mostrado gráficamente en la figura 18b. Como se puede ver el error oscila entre 0 y 0.0008 milímetros, obteniendo así una precisión mucho mayor.



(a) Trayectoria con 3000 puntos.



(b) Error en la trayectoria con 3000 puntos.

Figura 18: Trayectoria rotacional con 3000 puntos.

De nuevo las conclusiones son similares. En el caso de un discretizado de 20 puntos la trayectoria no proporciona unos resultados aceptables, mientras que la elección de 800 o 3000 puntos depende de la precisión necesaria a la hora de ejecutar la trayectoria. Esta trayectoria ha demostrado ser de una complejidad mayor a la anterior, y en ningún caso ha sido posible lograr que el robot permanezca en el punto de origen como se deseaba.

5. Conclusiones

A lo largo de este trabajo se ha realizado una implementación de la cinemática del robot Manfred V3 de la Universidad de Carlos III de Madrid sobre MATLAB utilizando la Robotics Toolbox. Esta implementación permite realizar simulaciones del comportamiento del robot antes de implementar los algoritmos sobre la plataforma real, permitiendo así evitar en la medida de lo posible errores que diesen lugar a dañar al propio robot.

Durante la implementación de esta cinemática se han ido descubriendo problemas muy habituales en este tipo de aplicaciones, como la imposibilidad de aplicar la cinemática inversa sobre un robot redundante, ya que la existencia de múltiples soluciones complica la capacidad de resolver iterativamente la solución. Por tanto se ha optado por trabajar con la matriz Jacobiana, la cual permite dar una solución a dicho problema cinemático con bastante precisión.

Además de este punto, ha quedado patente lo importante que resulta una buena discretización de la trayectoria. Durante los ensayos realizados se han generado tres trayectorias concretas discretizadas en distintas cantidades de puntos, analizando los resultados de la simulación en cada caso, y los errores de movimiento generados. Como conclusión final a estos ensayos queda que un mayor número de puntos intermedios en la trayectoria aumenta la precisión de la trayectoria realizada por el robot y por tanto los resultados arrojados son mejores.