Diseño, Simulación y Análisis de un Robot Serial de Seis Grados de Libertad para Aplicaciones de Manipulación

Juan Ignacio Quiroga - Legajo: 13889
Juan Marcos De Miguel - Legajo: 13673
Matías Armani - Legajo: 13664

Facultad de Ingeniería - Universidad Nacional de Cuyo
Profesora responsable: Carolina Díaz Baca
Jefe de Trabajos Prácticos: Eric Sánchez Ferreyra
Año de Cursado: 2024

Resumen

Este documento describe el desarrollo y análisis de un robot serial de seis grados de libertad, diseñado para aplicaciones de manipulación en un entorno estructurado. Se presentan los aspectos teóricos y prácticos relacionados con la parametrización de la estructura, los problemas de cinemática directa e inversa, y la planificación de trayectorias. Se incluye una validación de los modelos mediante simulaciones realizadas en MATLAB, junto con la descripción de sensores, actuadores y las limitaciones del sistema.

Índice

1. Introducción

La robótica ha crecido significativamente en los últimos años, especialmente en industrias que demandan alta precisión y flexibilidad. Este proyecto, desarrollado para la materia Robótica I, tiene como objetivo diseñar, simular y analizar un robot serial de seis grados de libertad enfocado en la aplicación del servido automático de helado en potes de telgopor.

El robot, compuesto por seis articulaciones, ha sido diseñado para extraer helado de diferentes vasquetas y servirlo en potes de 1 kg, 1/2 kg y 1/4 kg. El proyecto abarca el desarrollo de modelos de cinemática directa e inversa, planificación de trayectorias optimizadas, e integración de sensores y actuadores para garantizar el control preciso del sistema.

Las simulaciones, llevadas a cabo en MATLAB y el Robotics Toolbox de Peter Corke, buscan validar el comportamiento del robot y enfrentar los desafíos propios de su implementación física en un entorno de servicio de alimentos.

2. Presentación Técnica del Robot y la Tarea

En esta sección se presenta el robot en términos de sus especificaciones técnicas, espacio de trabajo y parametrización. El robot desarrollado es un manipulador serial de seis grados de libertad, específicamente diseñado y optimizado para el servido automático de helado en entornos estructurados. A continuación, se describen las características técnicas más relevantes del sistema.

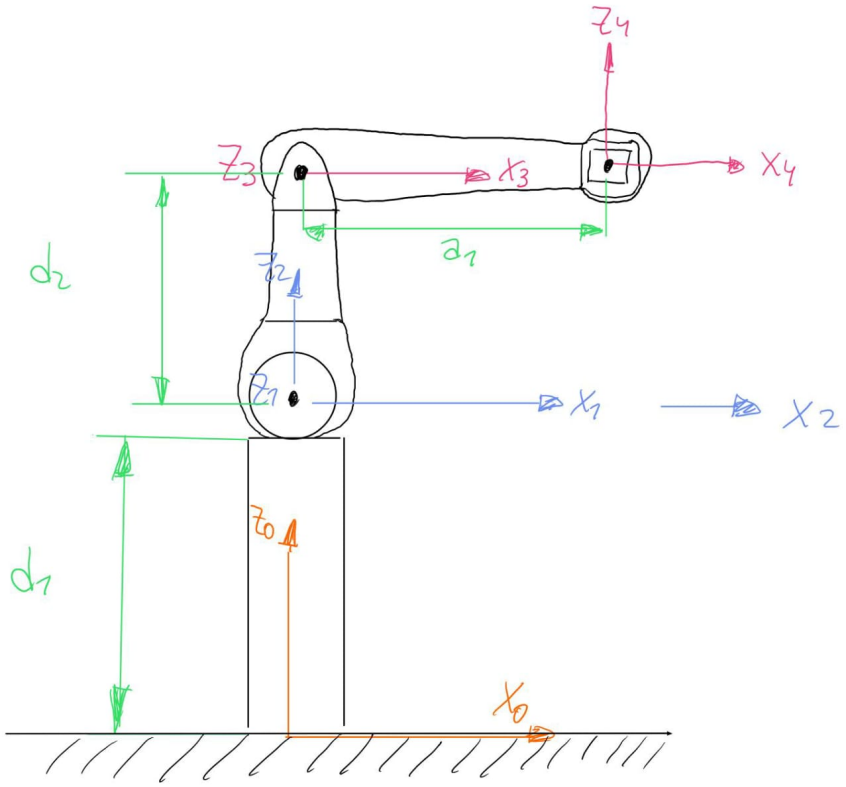

Figura 1: Representación esquemática del robot.

3. Presentación Técnica del Robot y la Tarea

En esta sección se presenta el robot en términos de sus especificaciones técnicas, espacio de trabajo y parametrización. El robot desarrollado es un manipulador serial de seis grados de libertad, diseñado y optimizado específicamente para el servido automático de helado en entornos estructurados. A continuación, se describen las características técnicas más relevantes del sistema.

3.1. Esquema del Robot y Espacio de Trabajo

El robot ha sido modelado utilizando parámetros físicos como longitudes de eslabones y ángulos articulares, optimizando su eficiencia para la manipulación de helado. El diseño del robot sigue la convención de Denavit-Hartenberg (DH) para la parametrización de las articulaciones. Los parámetros d se utilizan en los eslabones que rotan alrededor de su propio eje longitudinal (base, brazo y cuchara), mientras que los parámetros a se aplican en eslabones que rotan alrededor de ejes perpendiculares (como el antebrazo). En las articulaciones 2 y 5 no hay parámetros a ni d, ya que los sistemas coordenados tienen origen coincidente.

Los ejes x de los distintos sistemas están orientados hacia la derecha, mientras que los ejes z se alternan entre apuntar hacia arriba y hacia afuera del plano, lo cual se refleja en la matriz DH, donde los valores de α alternan entre $\pi/2$ y $-\pi/2$, dependiendo de la orientación necesaria para cada transformación entre eslabones.

Este patrón permite describir las transformaciones homogéneas necesarias para cada eslabón del robot, facilitando el análisis cinemático y la simulación. La correcta elección de los parámetros DH asegura que el efector final pueda ejecutar movimientos precisos, evitando colisiones y garantizando la eficiencia operativa.

La matriz DH del robot es la siguiente:

$$\mathrm{dh} = \begin{bmatrix} 0 & d_1 & 0 & \pi/2 & 0 \\ 0 & 0 & 0 & -\pi/2 & 0 \\ 0 & d_3 & 0 & \pi/2 & 0 \\ 0 & 0 & a_4 & -\pi/2 & 0 \\ 0 & 0 & 0 & \pi/2 & 0 \\ 0 & d_6 & 0 & 0 & 0 \end{bmatrix}$$

Donde:

- $d_1 = 0{,}150\,\mathrm{m}$
- $d_3 = 0{,}500\,\mathrm{m}$
- $a_4 = 0{,}500\,\mathrm{m}$
- $d_6 = 0{,}150\,\mathrm{m}$

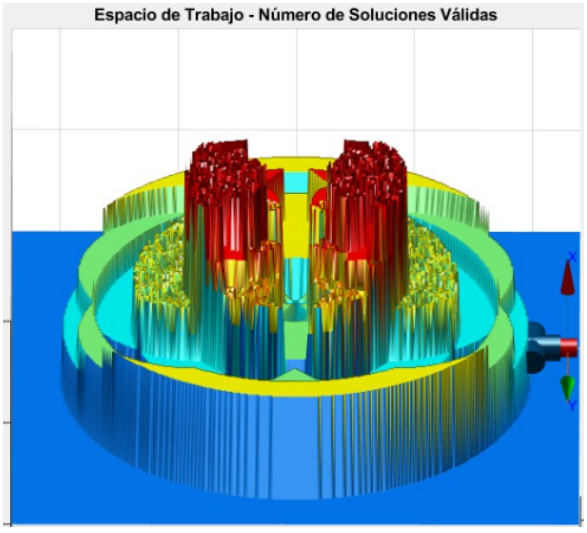

Figura 2: Representación del espacio de trabajo del robot en el plano XY.

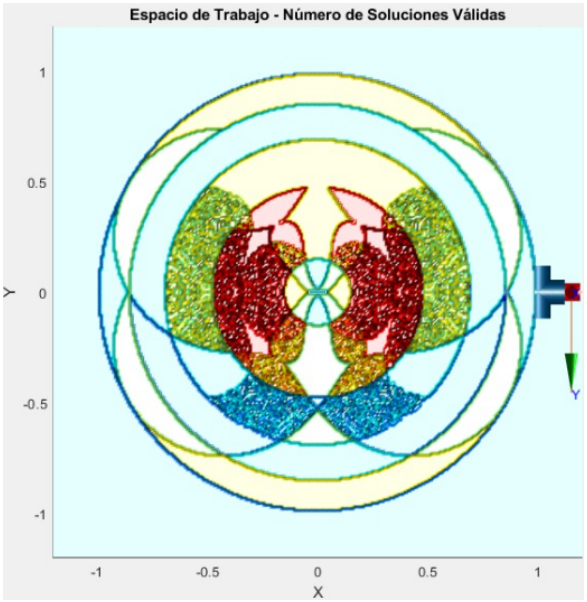

Figura 3: Mapa de calor del espacio de trabajo del robot. Vista superior.

Rojo indica 8 soluciones posibles, azul indica 0 soluciones.

3.2. Límites Articulares

El robot consta de seis articulaciones rotativas, cada una con un rango específico de movimiento definido para evitar colisiones y garantizar la seguridad operativa. Los límites articulares se han configurado de la siguiente manera:

- Articulación 1: $[-180°, 180°]$
- Articulación 2: $[-60°, 240°]$
- Articulación 3: $[-180°, 180°]$
- Articulación 4: $[-150°, 150°]$
- Articulación 5: $[-150°, 150°]$
- Articulación 6: $[-150°, 150°]$

3.3. Configuración del Robot

El robot ha sido creado y configurado en MATLAB usando Robotics Toolbox con los siguientes parámetros:

```
R = SerialLink(dh,'name','Robot Heladero','plotopt',{'scale',0.8,
'jointdiam',1.4,'jointcolor',[0 0.3 0.5],'name','tilesize',0.2,
'tile1color',[1 1 1],'noname','lightpos',[0 0 10]});
```

Offsets articulares:

$$R.offset = [0, -\pi/2, \pi/2, \pi/2, \pi/2, 0]$$

Esta configuración asegura que el robot esté correctamente ajustado para sus tareas dentro del sistema, permitiendo una operación precisa y controlada.

4. Cinemática Directa

4.1. Descripción del Problema

El problema de la cinemática directa consiste en determinar la posición y orientación del efector final del robot dado un conjunto de valores articulares $q = [q_1, q_2, \ldots, q_6]$. Esto se logra aplicando una serie de transformaciones homogéneas que describen la posición de cada eslabón en relación con el anterior, basándonos en los parámetros DH previamente definidos.

4.2. Método de Solución

Para resolver el problema de la cinemática directa, se utilizaron transformaciones homogéneas derivadas de los parámetros Denavit-Hartenberg (DH). La posición y orientación del efector final están dadas por la multiplicación secuencial de las matrices de transformación homogénea de cada articulación:

$$T = T_1 \cdot T_2 \cdot T_3 \cdot T_4 \cdot T_5 \cdot T_6$$

donde cada T_i se define mediante:

$$T_i = \text{Rot}_z(\theta_i) \cdot \text{Trans}_z(d_i) \cdot \text{Trans}_x(a_i) \cdot \text{Rot}_x(\alpha_i)$$

donde $\theta_i, d_i, a_i, \alpha_i$ son los parámetros DH de la articulación i.

4.3. Validación de la Solución

Para la validación se calcularon las matrices de transformación homogénea manualmente, comparándolas luego con los resultados de la función ‘fkine‘ del toolbox, la cual calcula la cinemática directa del robot.

El siguiente código ejemplifica el proceso de cálculo:

```
function listMatrix = matricesCinematicaDirecta(q, dh)
    n = length(q)+1;
    prod = eye(4);
    listMatrix = zeros(4, 4, n);
    for i = (1:n-1)
        listMatrix(:,:,i) = trotz(q(i)) * ...
        transl(dh(i,3),0,dh(i,2)) * trotx(dh(i,4));
        prod = prod * listMatrix(:,:,i);
    end
    listMatrix(:,:,n) = prod;
end
```

La función ‘matricesCinematicaDirecta‘ genera las matrices de transformación para cada eslabón y el efector final. Los resultados fueron comparados con ‘fkine‘ del toolbox, mostrando concordancia y validando la precisión del modelo implementado.

5. Cinemática Inversa

5.1. Descripción del Problema

La cinemática inversa es el proceso de determinar los valores articulares $q = [q_1, q_2, \ldots, q_6]$ que posicionan el efector final en una ubicación y orientación deseadas. Este problema es más complejo que la cinemática directa, ya que puede tener múltiples soluciones o incluso no tener ninguna solución en ciertos casos.

5.2. Método de solución

5.2.1. Método geométrico

Para la elaboración de la función de cinemática inversa que permitiera calcular las coordenadas articulares a partir de la matriz de transformación homogénea del sistema de coordenadas del efector final, se llevó a cabo el siguiente desarrollo:

Posición de la muñeca: Se calcula la posición de la muñeca retrocediendo desde el efector final una distancia d_6 en la dirección del versor Z:

$$p_{\text{muñeca}} = p_{\text{efector}} - d_6 \cdot Z_{\text{efector}}$$

Intersección de esferas: Se generan dos esferas: una centrada en el hombro del robot, con radio igual a la longitud del brazo (d_3), y otra centrada en la muñeca, con radio igual al eslabón del antebrazo (a_4). La intersección de estas esferas genera una circunferencia en el espacio, esta circunferencia posee todos los puntos en los cuales se debe posicionar el codo del brazo robótico para colocar la muñeca en el punto calculado en el primer paso. Si bien los infinitos puntos de la circunferencia satisfacen esta condición, lo que se busca es llevar el efector final a la posición y orientación correcta. Se descubrió que sólo dos puntos de la circunferencia serían los indicados para posicionar el codo, con el objetivo de posicionar y orientar el efector final de forma correcta.

Selección de la posición del codo: Para cualquier posición y orientación del efector final, y debido a las características inherentes de este brazo robótico, los ejes Z_3 y Z_4 siempre son perpendiculares entre sí, esto significa también que la familia de planos generados por dichos vectores son perpendiculares. Llamaremos plano 1 al plano definido por el vector Z_3 y que contiene al punto del codo, y plano 2 al plano definido por el vector Z_4 y que contiene también al punto del codo. Para cualquier punto de la circunferencia producto de la intersección de las dos esferas, se puede generar el vector perpendicular al primer plano a partir del producto vectorial de los vectores $(p_{\text{muñeca}} - p_{\text{hombro}})$ y $(p_{\text{codo}} - p_{\text{hombro}})$, este producto vectorial da como resultado un vector paralelo al vector Z_3. El vector perpendicular que define al segundo plano se puede calcular a partir del producto vectorial de los vectores $(p_{\text{muñeca}} - p_{\text{codo}})$ y Z_{efector}, este producto vectorial da como resultado un vector paralelo al vector Z_4. Aquellos puntos de la circunferencia en los cuales el producto escalar de estos dos vectores es igual a 0, lo que significa que los planos 1 y 2 son perpendiculares, son los puntos donde se debe posicionar el codo para poder llevar el efector final a la posición y orientación correctas.

Cálculo de ángulos articulares: El paso final en el desarrollo de la función de cinemática inversa, una vez que se obtienen los puntos del codo, es calcular todas las coordenadas articulares para satisfacer la posición y orientación del efector final. La coordenada articular q_1 se calcula como el ángulo que forma la proyección del punto del codo en el plano XY del sistema 0 con respecto al vector X_0. q_2 es el ángulo que forma el punto

del codo, visto desde el plano XY del sistema 1, con respecto al vector Y_1. Existen dos combinaciones de q_1 y q_2 que pueden llevar el codo a una determinada posición. Para el cálculo de q_3 y q_4 se sigue un procedimiento similar al desarrollado para calcular q_1 y q_2. La coordenada articular q_3 se calcula como el ángulo que forma la proyección del punto de la muñeca en el plano XY del sistema 2 con respecto al vector X_2, y q_4 es el ángulo que forma el punto de la muñeca, visto desde el plano XY del sistema 3, con respecto al vector X_3. La coordenada articular q_5 se calcula como el ángulo que forma el punto del efector final en el plano XY del sistema 4 en relación al vector $-Y_4$. Finalmente, q_6 es el ángulo que forma el versor X del efector final en relación al vector X_5.

Soluciones posibles: En general, se obtienen hasta 8 soluciones para el vector de ángulos articulares (q): para cada punto del codo, se tienen 2 combinaciones de q_1 y q_2. Además, para cada una de las dos soluciones que posicionan el codo en un mismo punto, se tienen 2 combinaciones de q_3 y q_4 para posicionar la muñeca en el punto calculado en el primer paso. Al tener 2 puntos del codo que dan lugar a 4 soluciones diferentes cada uno, se tiene un total de 8 soluciones.

Optimización: Finalmente, se selecciona la mejor solución en función de la proximidad a una configuración inicial (q_0), minimizando la norma de la diferencia entre las configuraciones y evitando movimientos del codo innecesarios en el medio de una trayectoria.

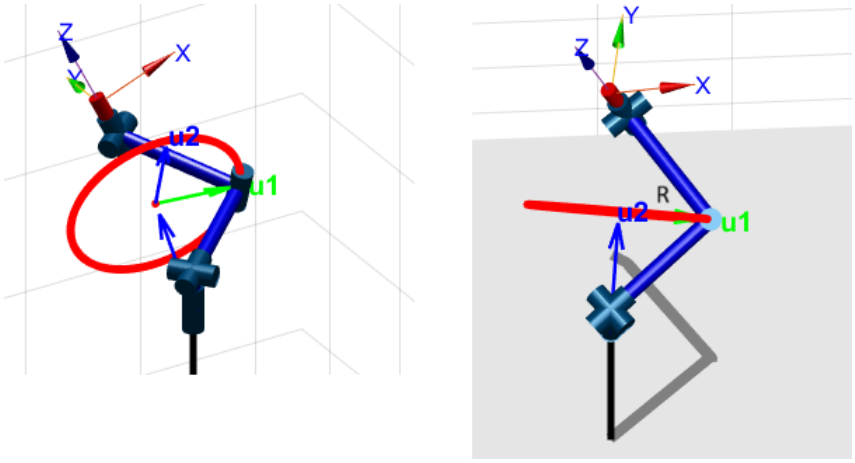

Figura 4: Circunferencia 3D y parametrización de la misma

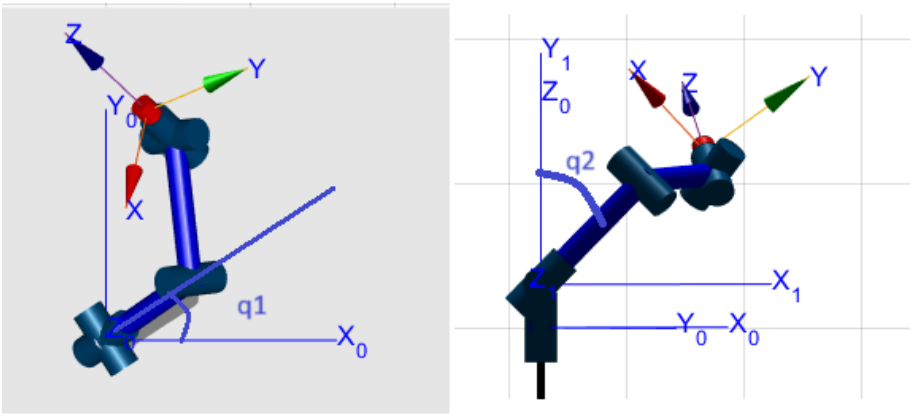

Figura 5: Cálculo de q1 y q2

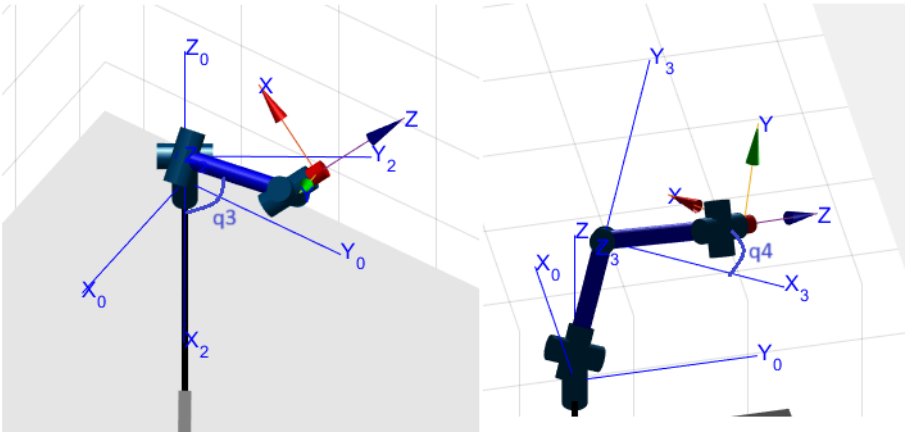

Figura 6: Cálculo de q3 y q4

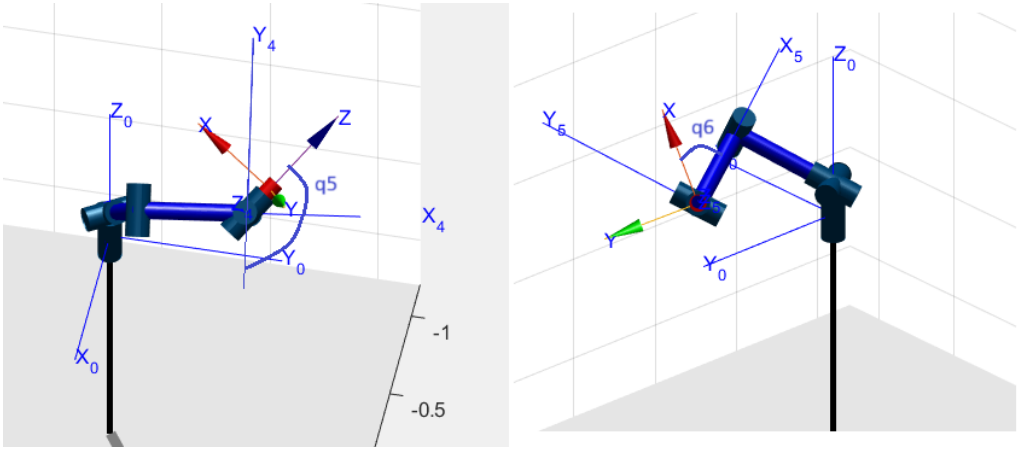

Figura 7: Cálculo de q5 y q6

5.2.2. Método a partir de función ikine del Toolbox

La segunda función desarrollada de cinemática inversa es la implementada en la función cinematicaInversaIkine. El mismo se basa en el método nativo del toolbox de robotics de Peter Corke, pero permite encontrar las 8 soluciones.

El código comienza igual que el anterior, aplicando las transformaciones que eliminan la matriz de tool y de base, y luego restando la distancia d1 en la dirección del versor z, de esta forma se queda ubicado en el punto del hombro.

Una vez ubicado en el hombro, se sabe que, por las características del robot, el único q que determina la distancia entre el hombro y la muñeca es q4.

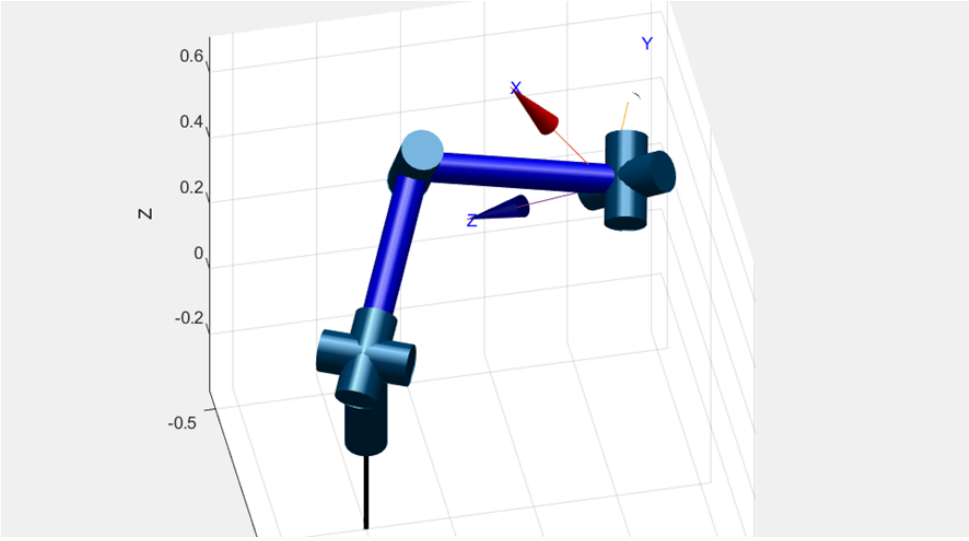

Figura 8: Zoom del codo del robot, dónde se aprecia que q4 es el único que varía la distancia entre el hombro y la muñeca

Se calcula entonces q4 usando el teorema del coseno, con la distancia entre el hombro y la muñeca y los dos eslabones del brazo. Si bien q4 no es el ángulo entre los mismos, se puede derivar fácilmente de ese.

La implementación quedando:

```
d = norm(p_muneca_hombro);
alfa = acos(((R.links(3).d)^2+(R.links(4).a)^2-d^2)  ...
/(2*(R.links(3).d)*(R.links(4).a)));
beta = pi - alfa;
% Valores conocidos de q4
q4_values = [pi/2+beta, pi/2-beta];
if q4_values(1) > pi
    q4_values(1) = q4_values(1) - 2*pi;
end
```

Estos dos valores de q4 se usan desde el principio como semilla para el metodo R.ikine, mientras que los demás q asignan aleatoriamente, planteando que cada vector q generado tenga al menos una de sus componentes en un cuadrante distinto de su intervalo [-pi,pi] de valores posibles. Es decir que nunca se repita una situacion en la que, por ejemplo, todos los q se encuentren en el primer cuadrante [0, pi/2], con el objetivo de hacer más rapida la búsqueda del ikine.

Para q5, tambien solo existen dos valores posibles, y uno puede ser calculado facilmente a partir del otro, por lo que en cuanto el ikine encuentra una solucion, dejan de ser aleatorios los valores de q5 de la semilla y pasan a alternar entre los dos posibles. En la imagen siguiente se ve como estos tambien se alternan con los de q4 a lo largo de las soluciones.

59.2380	66.8720	43.1422	60.0000	20.0000	41.8305
-120.7620	-66.8720	-136.8578	60.0000	20.0000	41.8305
30.0000	50.0000	-110.0000	60.0000	160.0000	-170.0000
-150.0000	-50.0000	70.0000	60.0000	160.0000	-170.0000
30.0000	50.0000	70.0000	120.0000	20.0000	10.0000
-150.0000	-50.0000	-110.0000	120.0000	20.0000	10.0000
59.2380	66.8720	-136.8578	120.0000	160.0000	-138.1695
-120.7620	-66.8720	43.1422	120.0000	160.0000	-138.1695

Figura 9: 8 soluciones de la cinemática inversa del robot, donde se aprecian las simetrías que existen entre ellas, marcados en rojo y azul los "grupos"de soluciones

El comportamiento de todos los q (exepto q4, que puede ser calculado directamente), fue concluido a partir de la observacion de las 8 soluciones obtenidas por ikine, el cual inicialmente fue puesto a trabajar hasta que encontraba 8 soluciones de manera aleatoria (lo cual podía tardar más de 20 segundos).

Una vez que se tuvieron soluciones, fue posible observar las reflexiones, o si los ángulos de las mismas eran complementarios o suplementarios.

La observación más importante realizada fue que existian dos grupos de 4 soluciones, en los cuales todas las soluciones del grupo pueden ser derivados de cualquier solucion que forme parte del grupo. Dichos grupos se encuentran enmarcados en azul y rojo, respectivamente. La condición que permite confirmar que se han hallado por lo menos una solucion perteneciente a cada grupo es que q6 sean diferentes (ya que cada valor de q6 se repite dos veces), y que la suma de sus valores absolutos sea diferente a pi.

Cada vez que se encuentra una solucion se verifica esta condición, y apenas la misma es satisfecha se deja de buscar soluciones numericamente con ikine y se pasa a completar los valores analiticamente, a partir de la informacion ya obtenida.

Esto asegura que el tiempo de ejecución máximo del programa sea el que le toma a ikine encontrar 5 soluciones (puesto que eso asegura 100 % que por lo menos hay una de cada grupo), la gran mayoria de las veces solo requiriendo 2,3 o 4. Si bien no está ni cerca de la velocidad de un metodo 100 % analítico, es mucho más aceptable que lo que demora ikine en encontrar las 8, demorando en promedio cerca de 1,5s en hallar una solucion, pero con mucha varianza ya que esta sujeto al azar de que ikine encuentre las soluciones que necesita.

El resto de las soluciones se encuentran de manera casi inmediata, pudiéndose completar q6 y q2 directamente con 100 % de seguridad. Esto es así ya que, si bien no hay solo dos valores posibles como en el caso de q4 y q5, son 4 que pueden ser extraidos y colocados directamente a partir de las dos soluciones obtenidas, ya que, habiendo obtenido y ordenado q4 y q5, q2 y q6 siempre se pueden calcular y disponer ordenadamente de manera que sean soluciones.

Para el caso de la combinación de q1 y q3 el proceso es un poco más complejo, ya que hay dos opciones. Para resolver esto se establece una solucion tentativa, y se comprueba si la aplicación de la cinemática directa a la misma lleva a un error menor al error numerico de ikine (en el orden de 1e-10). Si esto se cumple se encontró la combinación correcta, sino se asigna la otra, esto para cada una de las 8 que no hayan sido encontradas en el paso de la búsqueda por ikine.

Este metodo fue desarrollado en paralelo con el de la circunferencia, si comparamos los dos vemos que el otro es mucho más rápido, y consistente con esa velocidad, siempre

demora lo mismo, mientras que lo que tarda este esta sujeto al ikine, como ventajas podemos considerar la precisión (similar a la del ikine), y el hecho de que, si bien para desarrollarlo se utilizo el conocimiento de como sacar q4 geométricamente, el desarrollo del método se baso casi enteramente en la observación de resultados y sus simetrías, siendo desarrollado en considerablemente menos tiempo que el otro, el cual tiene una complejidad bastante mayor, y requiere de más conocimiento e intuición acerca del robot

6. Relación de Velocidades y Análisis Jacobiano

6.1. Análisis del Jacobiano

El Jacobiano relaciona las velocidades articulares $\dot{q}$ con las velocidades del efector final $\dot{x}$:

$$\dot{x} = J(q)\dot{q}$$

donde $J(q)$ es el Jacobiano, calculado en función de las coordenadas articulares q. Este análisis permite estudiar cómo se propagan las velocidades desde las articulaciones hasta el efector final, además de identificar posibles singularidades donde el robot pierde grados de libertad.

6.2. Cálculo del Jacobiano

Para calcular el Jacobiano del robot se plantearon tres enfoques distintos, dos de ellos utilizando métodos simbólicos y un tercero basado en la evaluación numérica de configuraciones específicas:

- Los primeros dos métodos intentaron obtener el Jacobiano de manera simbólica y calcular su determinante. Sin embargo, a pesar de dejar el cálculo trabajando durante varias horas, estos métodos no lograron converger a un resultado debido a la complejidad del sistema.
- El tercer enfoque consistió en evaluar numéricamente el determinante del Jacobiano en ciertos puntos seleccionados a partir del análisis geométrico del robot. Este método nos permitió verificar si dichas configuraciones eran efectivamente singulares.

6.3. Identificación de Singularidades

Finalmente, los puntos singulares se determinaron mediante observación y análisis del comportamiento del robot. Estos puntos se verificaron usando el tercer enfoque mencionado anteriormente. Los puntos singulares encontrados son los siguientes:

```
Determinante en singularidad de muñeca (q5 = pi/2): 5.493163e-18
Determinante en singularidad de hombro (q2 = 0): -1.628096e-18
Determinante en singularidad de codo (q4 = pi/2): 7.478259e-18
```

Estos valores cercanos a cero indican que las configuraciones mencionadas corresponden a puntos singulares, lo cual limita los movimientos posibles del robot y debe ser considerado en el diseño de trayectorias.

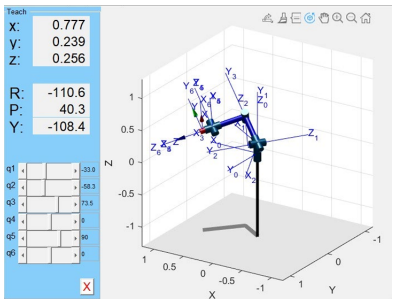

Figura 10: Singularidad de muñeca ($q_5 = \pi/2$)

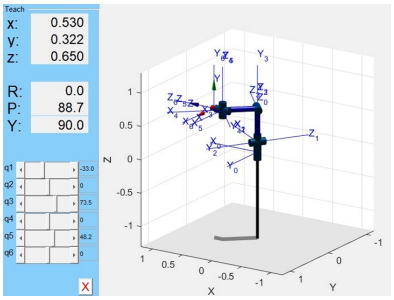

Figura 11: Singularidad de hombro ($q_2 = 0$)

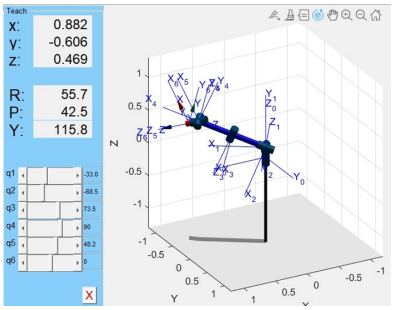

Figura 12: Singularidad de codo ($q_4 = \pi/2$)

7. Planificación y Generación de Trayectorias

7.1. Planificación de Trayectorias

La planificación de trayectorias es esencial en el control de sistemas robóticos, ya que implica el diseño de un camino óptimo entre un punto inicial y un objetivo final, respetando restricciones de velocidad, aceleración y posibles obstáculos. Este proceso puede realizarse tanto en el espacio articular como en el espacio de trabajo, buscando optimizar el rendimiento del robot y garantizar precisión y seguridad en su operación.

En el proyecto del robot servidor de helado, la planificación de trayectorias tiene un enfoque particular, ya que las trayectorias deben ser suaves y visualmente atractivas, evitando movimientos bruscos que podrían afectar la percepción del sistema y la precisión al manipular helado, una sustancia semi-sólida que requiere un control cuidadoso del efector final.

- **Entorno y limitaciones físicas**: Las vitrinas donde se encuentran las bandejas de helado representan un desafío debido a las restricciones de espacio. Las trayectorias deben ser rápidas y precisas para evitar colisiones con las estructuras de la vitrina y otros componentes. Esto exige la utilización de técnicas avanzadas, como la interpolación cúbica o quíntuple, para lograr una transición suave entre puntos y minimizar cambios abruptos en posición, velocidad y aceleración.

- **Interacción con el material**: A diferencia de un robot de pick and place tradicional, el proyecto implica manipular helado, lo cual demanda un control meticuloso del efector final para recoger la cantidad adecuada y servirla de manera limpia y precisa. Esto se logra manteniendo una orientación y fuerza constantes en la cuchara mediante una planificación adecuada en el espacio de trabajo y el control de las fuerzas aplicadas.

- **Consideraciones dinámicas**: La generación de trayectorias debe considerar la dinámica del sistema para evitar sobrecargar los motores y asegurarse de que el robot opere dentro de sus límites (posiciones, velocidades y aceleraciones máximas). Los picos de aceleración excesivos son imposibles de lograr en la práctica, ya que requerirían actuadores desproporcionados para el robot y la tarea.

En resumen, la planificación de trayectorias para un robot que sirve helado requiere un equilibrio entre estética y funcionalidad, precisión en la manipulación del material y

atención a las restricciones físicas del entorno. Implementar trayectorias suaves y eficientes garantiza tanto la operación efectiva del sistema como una experiencia positiva para los usuarios.

7.2. Generación de Trayectorias

La generación de trayectorias en robótica abarca desde la planificación hasta la implementación y el control dinámico del movimiento. Primero, un programa principal define los parámetros iniciales como puntos de paso, tipo de trayectoria, tiempo, velocidad y aceleración. Estos parámetros se envían al generador de trayectorias, que calcula los caminos respetando las limitaciones cinemáticas del robot. Luego, se realiza el muestreo para generar consignas utilizadas en el control del robot, asegurando que los movimientos respeten las restricciones físicas y de tiempo real.

7.2.1. Generación de trayectorias en espacio articular

En el espacio articular, cada articulación del robot se controla de manera independiente, permitiendo una implementación más sencilla y directa, con menor carga computacional. Este enfoque evita singularidades y posiciones incómodas, y es especialmente atractivo en el proyecto del robot servidor de helado por su estética, ya que tiende a generar trayectorias curvas o circulares más agradables visualmente.

7.2.2. Generación de trayectorias en espacio cartesiano

En el espacio cartesiano, el control se centra en el movimiento del efector final en el espacio de trabajo, definiendo trayectorias específicas en un sistema de coordenadas absoluto. Este enfoque es ideal para aplicaciones que requieren alta precisión, aunque puede implicar cálculos más complejos y presentar desafíos en la cinemática inversa. Las trayectorias en este espacio son más coherentes para el efector final, pero pueden resultar en movimientos más complejos para las articulaciones.

7.2.3. Comparación de espacio articular y cartesiano

La principal diferencia radica en el tipo de control. Mientras que el espacio articular ofrece un control más directo y sencillo, el espacio cartesiano permite definir trayectorias precisas para el efector. La interpolación en el espacio articular garantiza trayectorias suaves para las articulaciones, mientras que en el espacio cartesiano se prioriza la trayectoria del efector, lo que puede llevar a movimientos menos óptimos para las articulaciones.

7.2.4. Métodos utilizados en MATLAB y Robotics Toolbox

Para la generación de trayectorias, se utilizaron los siguientes métodos:

- `jtraj`: Utilizado para la interpolación en el espacio articular, calcula trayectorias suaves entre dos puntos dados para cada articulación.
- `mstraj`: Aproximación en espacio articular con múltiples puntos de paso, permitiendo diseñar trayectorias que no requieren detener la velocidad en cada cambio de dirección, favoreciendo movimientos más fluidos.

- `ctraj`: En el espacio cartesiano, permite la interpolación de trayectorias definidas por matrices de transformación homogénea, garantizando precisión en posición y orientación.

7.3. Implementación de las Trayectorias

Para implementar los diferentes enfoques de interpolación previamente descritos, se utilizaron dos scripts en MATLAB: uno para la generación de trayectorias y otro para la simulación de estas.

El script de generación combina la funcionalidad del método `R.teach()` del toolbox con un botón adicional que permite capturar y almacenar el vector de posiciones articulares y la matriz de transformación homogénea del efector final en un archivo de texto. Este proceso fue útil para registrar cada punto crítico de la trayectoria, permitiendo luego decidir si se debía interpolar en el espacio articular o cartesiano según conveniencia.

El script de simulación lee los archivos de texto, almacena las posiciones en variables y permite trabajar en el espacio cartesiano o articular, interpolando las diferentes partes de las trayectorias de manera flexible.

A continuación, se muestran diversas trayectorias generadas a lo largo del proyecto, organizadas cronológicamente y por complejidad. Estas trayectorias reflejan la evolución y mejora en el control de los criterios definidos para la generación, utilizando métodos como `mstraj` para garantizar una posición y orientación adecuadas del efector final.

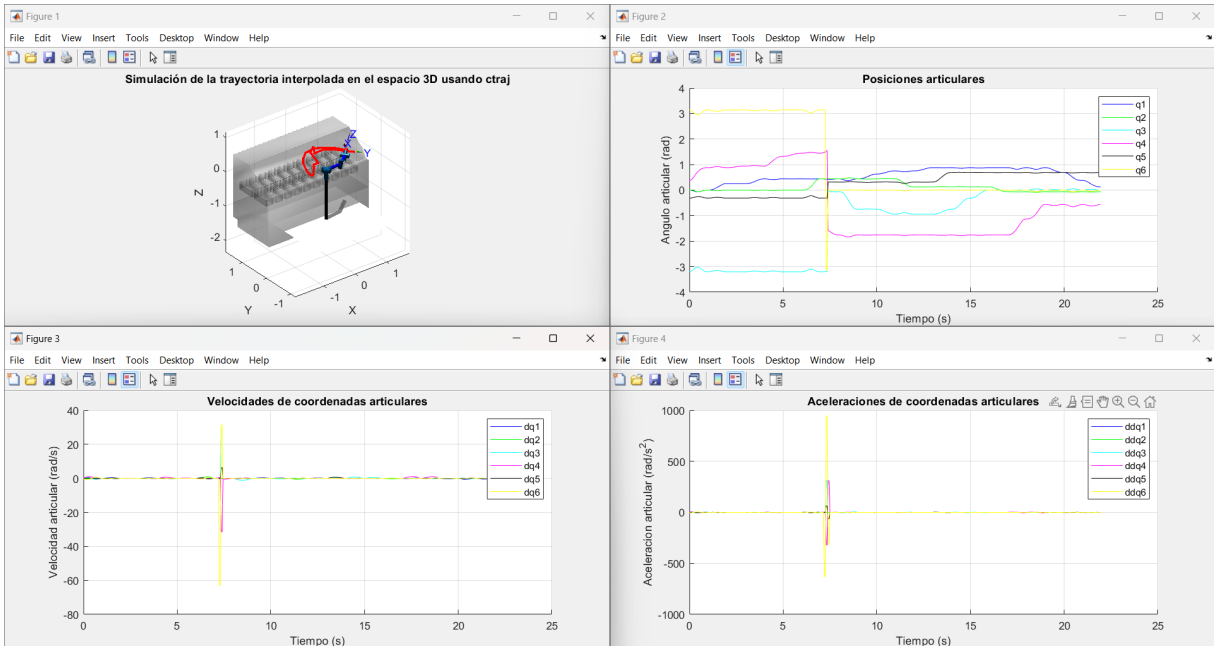

Figura 13: Trayectoria 1, se observa un salto dado por un movimiento brusco a la hora de generar la trayectoria, dando lugar a valores elevados de velocidad y aceleración

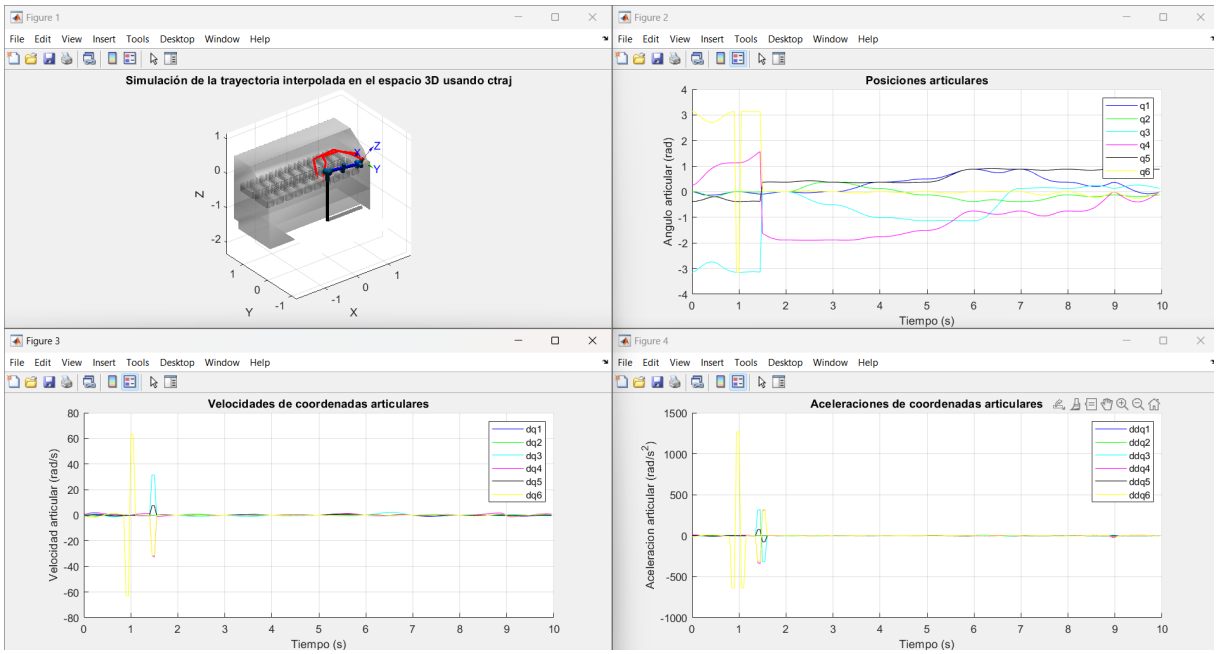

Figura 14: Trayectoria 2, en esta ocasión el salto se debe al trabajo muy cerca del limite articular de q6, lo que lleva a ese salto que tiene que dar de -pi a pi

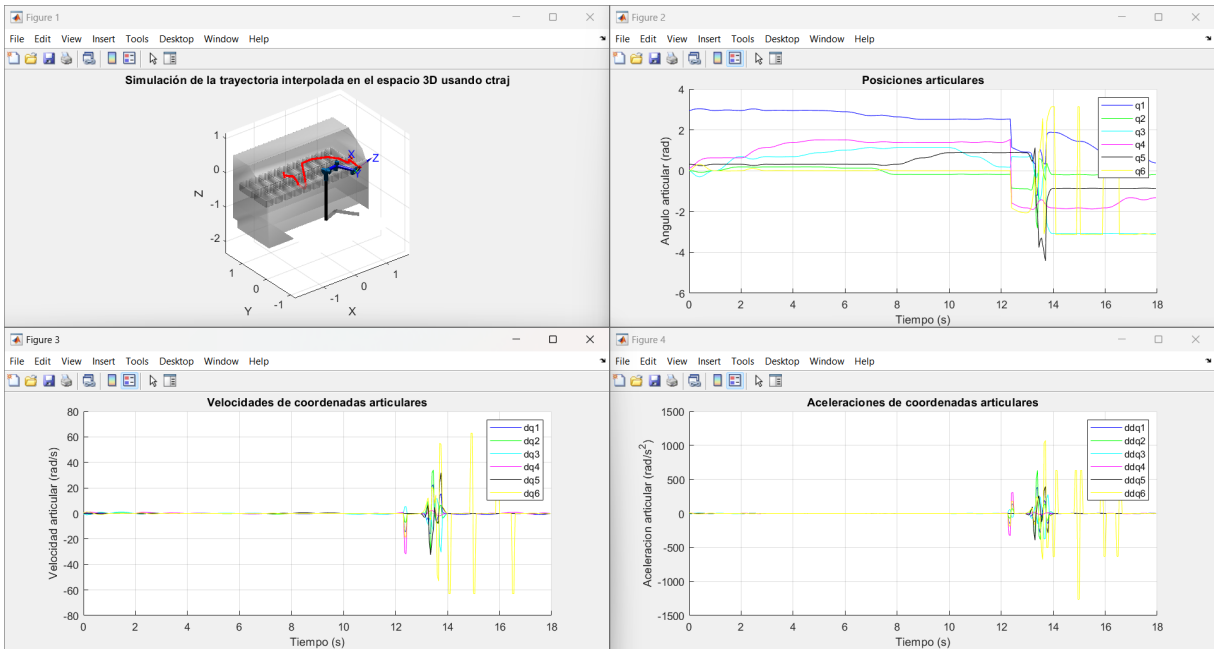

Figura 15: Trayectoria 3, ocurre lo mismo que en Trayectoria 2

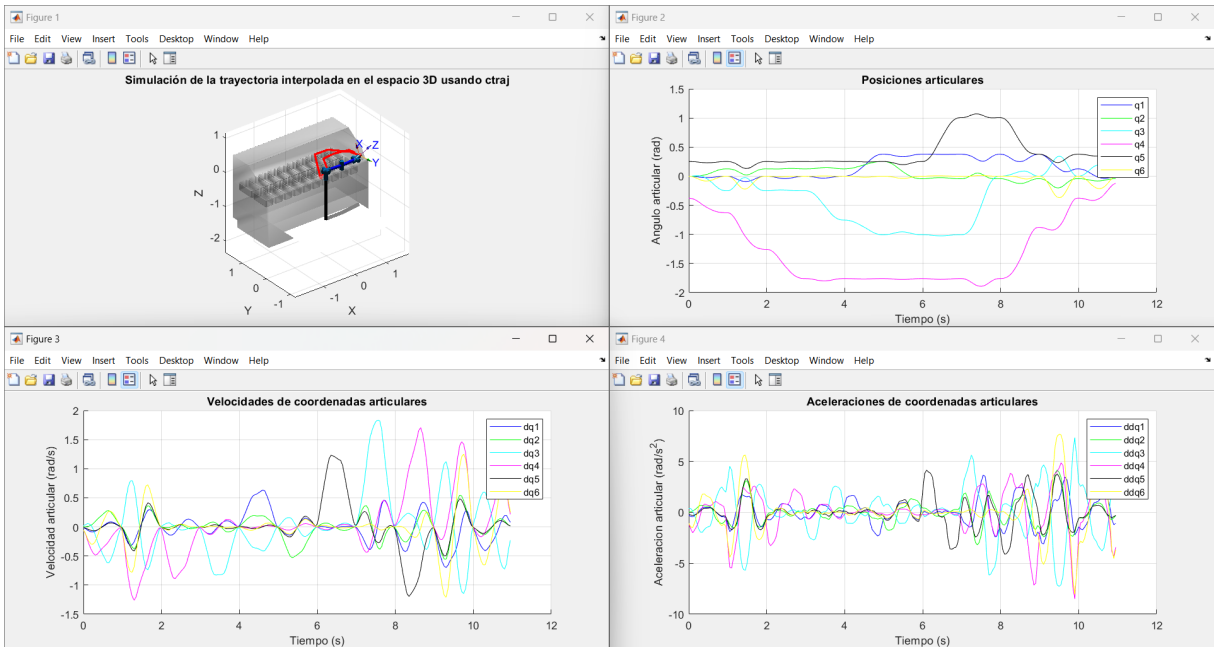

Figura 16: Trayectoria 4, se realizaron arreglos a la función de cinemática inversa, así como cuidados añadidos a la hora de generar las trayectorias, que permitieron evitar saltos articulares y así conseguir una buena trayectoria

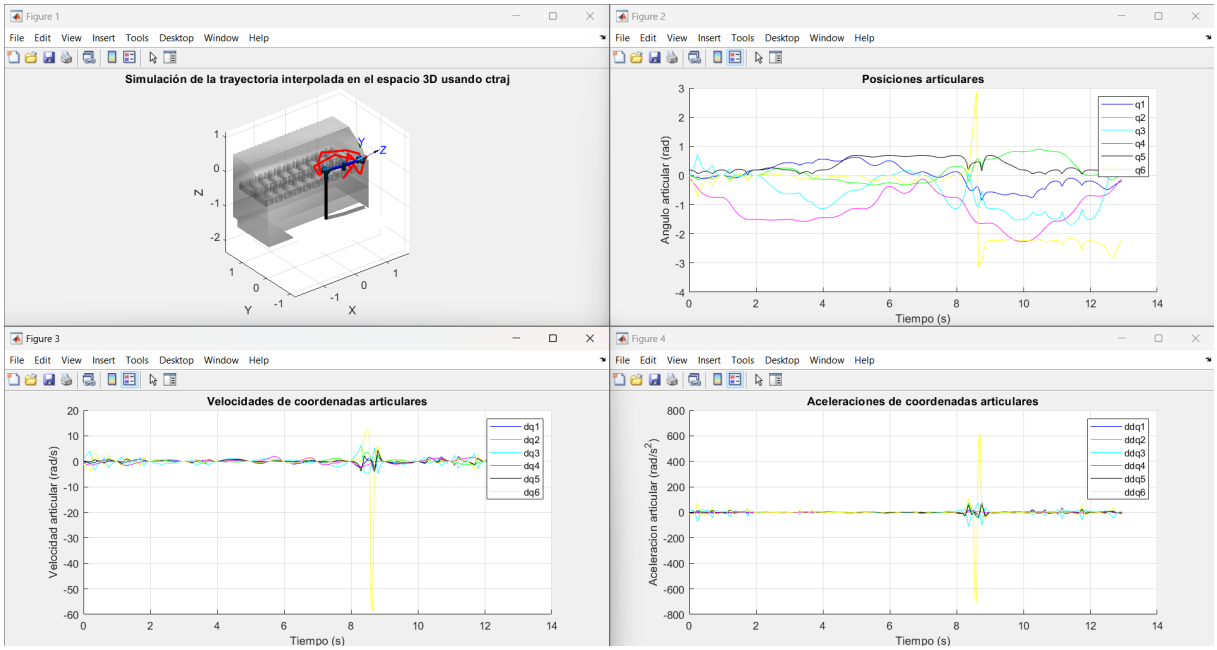

Figura 17: Trayectoria 5, más compleja que las anteriores, se buscó añadir la limpieza de la espátula, esto trajo de nuevo el problema de los saltos en q6.

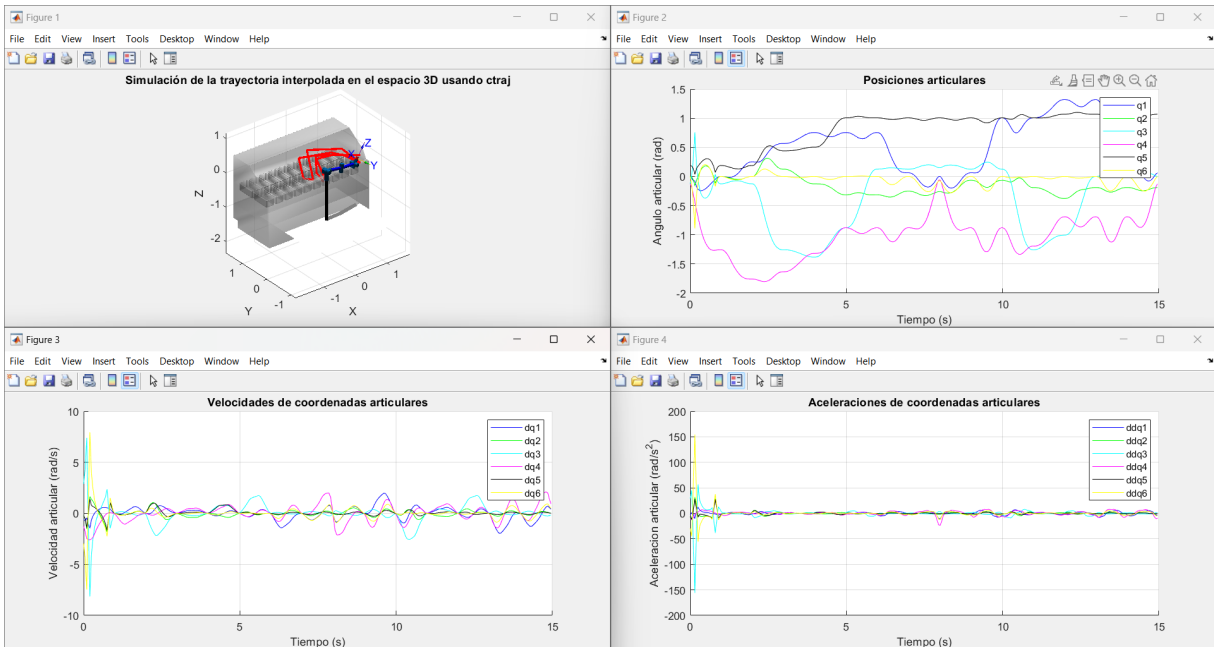

Figura 18: Trayectoria 6, la más compleja hasta el momento con el método de interpolación cartesiana, se añadió la doble bocha y se mantuvieron buenos valores en toda la trayectoria, aunque se presentan picos en el instante inicial

Luego de considerable trabajo de interpolación cartesiana, se optó por probar la realización de la misma tarea, pero esta vez interpolando en coordenadas articulares, se realizaron modificaciones en los scripts que permitieron la implementación de las funciones jtraj y mstraj que permitieron la siguiente trayectoria:

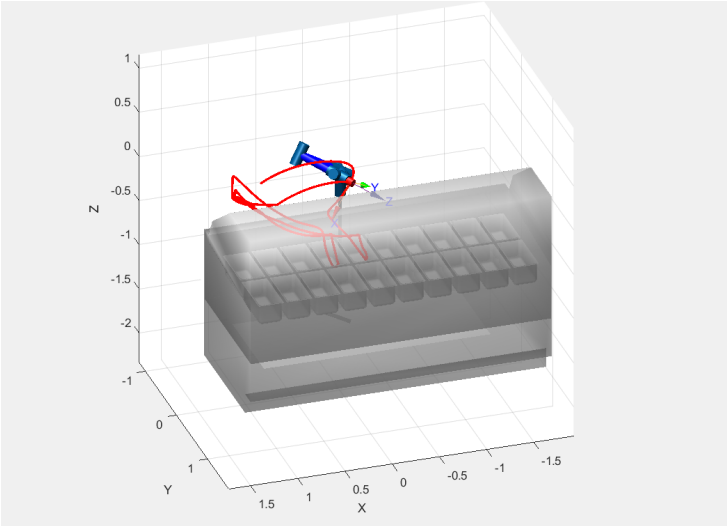

Figura 19: Trayectoria 7, utilizando aproximación o interpolación articular, interpolación en necesidad de mayor precisión, aproximación en los movimientos de traslado

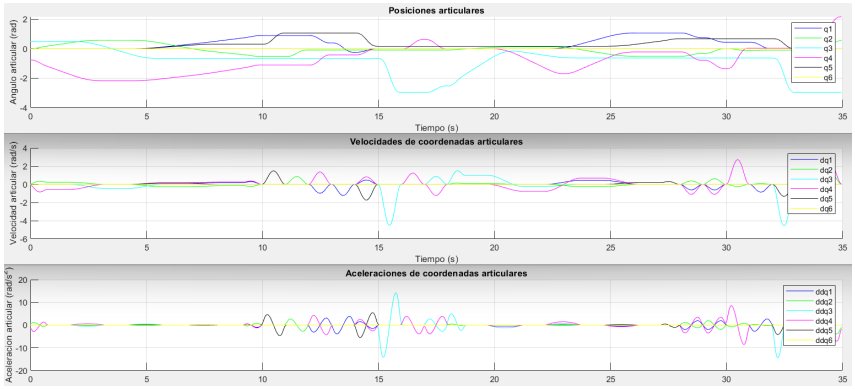

Figura 20: Posición, velocidad y aceleración articular de la trayectoria 7

Esta trayectoria, en comparación de las demás, presentó movimientos más fluidos, describiendo arcos en el aire en lugar de lineas rectas.

Otra ventaja a considerar es la que se puede observar en los perfiles de posición, velocidad y aceleración, en los que no existe ningún tipo de discontinuidad en la posición, por ende no hay picos de velocidad ni aceleración (cosa que se puede notar por la escala de las gráficas)

Además, se puede ver que los momentos en los que fue aproximada en lugar de interpolada la trayectoria, se consiguieron valores muy bajos de aceleración, debido a que no fue necesario que el robot parara y arrancara constantemente, sino que podía doblar en radios como se observa en la figura 10.

Luego de análisis y de devoluciones del equipo docente, donde se resaltó que ni uno ni el otro método (interpolar todo en el espacio cartesiano o aproximar todo en el espacio articular) eran el enfoque óptimo para la tarea, debido a que, si bien las aproximaciones articulares se ven mejor y generan curvas x,v,a más suaves, es necesaria la precisión en el efector final a la hora de extraer el helado de la vasqueta.

Tomando en cuenta todas estas consideraciones, se llegó a la siguiente trayectoria final, que combina tanto interpolación como aproximación articular, como interpolación cartesiana.

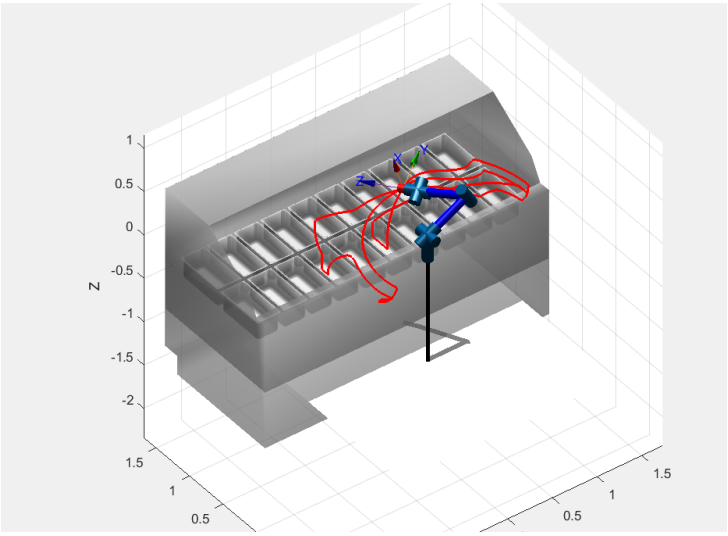

Figura 21: Trayectoria final, se observan las influencias de los distintos tipos de interpolación y aproximación

Como se observa en la figura, se logró implementar una trayectoria compleja que involucra el servido de dos bochas con una parada intermedia para limpiar la cuchara en agua, combinando los tres métodos utilizados: `jtraj`, `mstraj` y `ctraj`.

La trayectoria comienza con una aproximación desde la posición inicial del robot hasta la entrada a la cubetera, donde se inserta la cuchara en el helado. Para asegurar el levantamiento adecuado del helado, se trazaron dos líneas rectas perpendiculares que se recorren lentamente mientras se levanta el helado. No se consideró la variabilidad en la altura o distribución del helado, ya que escapa al alcance del proyecto. Sin embargo, se contempla la posibilidad de ampliar el proyecto en el futuro mediante la incorporación de una cámara con reconocimiento de imágenes, capaz de identificar la ubicación del helado y generar instrucciones para recoger la cantidad deseada, utilizando una referencia (como el punto central de la vasqueta elevado sobre el nivel máximo). Esto podría validarse mediante sensores de peso.

Otra alternativa sería discretizar el volumen de la vasqueta en puntos y sistematizar el levantamiento del helado para que la trayectoria pase siempre por ciertos puntos, los cuales se eliminarían de la matriz discretizada una vez utilizados, hasta que la vasqueta se vacíe. Aunque este enfoque es menos robusto, ya que no contempla la realimentación y podría acumular errores al no ser el helado homogéneo (diferente viscosidad o pedazos sólidos), es más sencillo y económico de implementar.

El resto de los movimientos de la trayectoria siguen la misma lógica: aproximaciones para los traslados e interpolaciones para las tareas que requieren precisión. En el caso del llenado del pote, se utilizó interpolación en el espacio articular debido a la naturaleza circular de los movimientos.

A continuación se presentan las gráficas de posición, velocidad y aceleración.

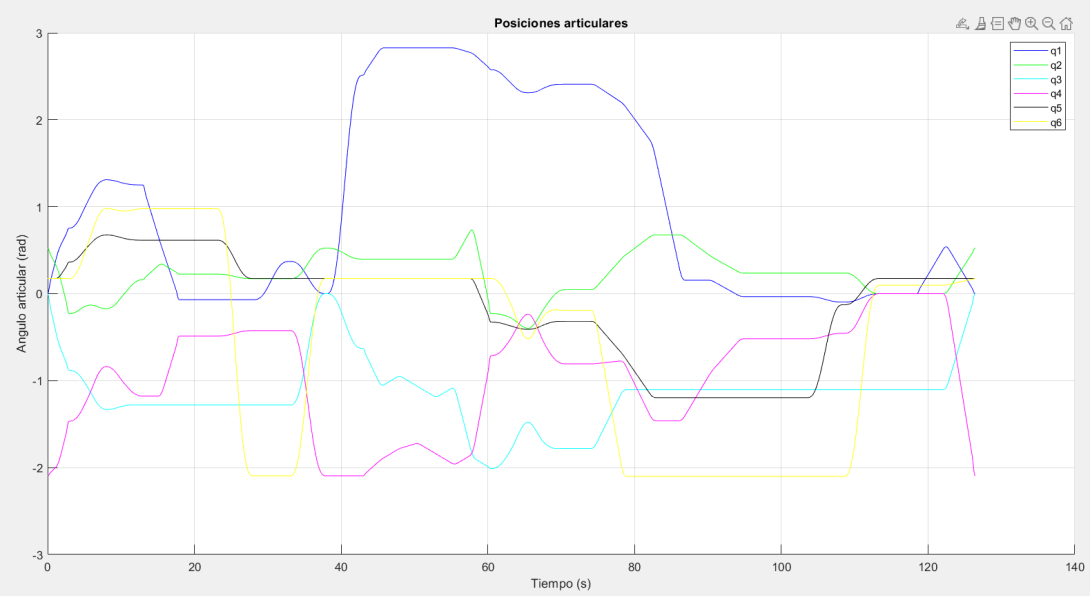

Figura 22: Posiciones articulares trayectoria final

Se observan las posiciones a lo largo del tiempo, donde no se da ningún salto ni discontinuidad entre las mismas.

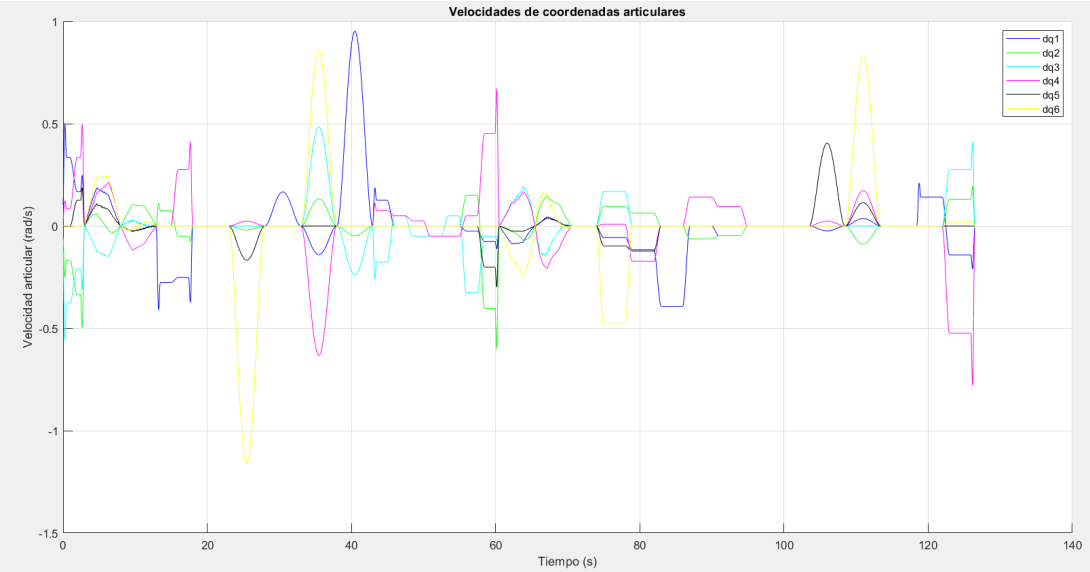

Figura 23: Velocidades articulares trayectoria final

Las velocidades reflejan lo dicho acerca de las posiciones, se puede observar los tramos en donde ha sido interpolado o aproximado, ya que las velocidades vuelven o no a cero, respectivamente.

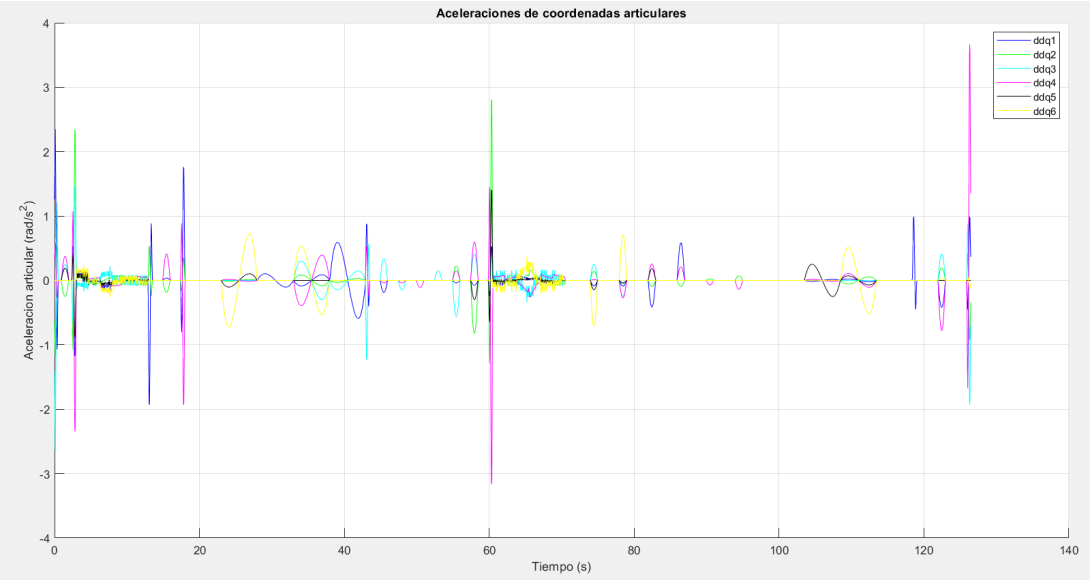

Figura 24: Aceleraciones articulares trayectoria final

Analizando las aceleraciones podemos ver que la escala del gráfico es la menor de las obtenidas hasta el momento, esto es bueno ya que menores aceleraciones requieren tamaño de actuadores menores. Se observan zonas de aceleraciones muy chicas donde se dan las aproximaciones de `mstraj`, y valores mayores en las zonas de interpolación donde se tiene que llevar y sacar del reposo al robot constantemente.

8. Sensores y Actuadores del Sistema

8.1. Sensores Internos

El robot incorporará los siguientes sensores internos para garantizar un funcionamiento seguro y preciso:

- **Encoders rotativos:** Miden la posición angular de cada articulación, proporcionando retroalimentación para el control preciso.
- **Sensores de límite:** Detectan los extremos de los rangos de movimiento, previniendo daños mecánicos.
- **Sensores de corriente:** Monitorizan el consumo de corriente para evitar sobrecargas en los motores.

8.2. Sensores Externos

Para mejorar la operación en un entorno real, se optará por incluir:

- **Cámaras de visión artificial:** Permiten la localización precisa de objetos y el ajuste de trayectorias.
- **Sensores de proximidad:** Detectan obstáculos cercanos para evitar colisiones.
- **Sensores de fuerza/torque:** Miden la fuerza aplicada por el efector, esencial para manipular el helado con precisión.

Para la aplicación en el robot heladero, lo ideal sería la incorporación de una cámara para el reconocimiento de imágenes del helado, tanto en las vasquetas como en el pote.

Un candidato viable sería la Intel RealSense D435, que utiliza tecnología de luz estructurada y un sensor de profundidad, permitiendo capturar datos en 3D. Esta capacidad de medir la profundidad permitiría estimar el volumen de helado restante en la vasqueta, y determinar cuándo es necesario reponer

8.3. Actuadores

Para las articulaciones se seleccionarán los siguientes actuadores:

- **Motores paso a paso:** Para el control preciso de posición en movimientos definidos y suaves.
- **Servomotores:** Para las articulaciones críticas que requieran control preciso de posición, velocidad y torque.
- **Motores de corriente continua (DC):** Para movimientos continuos sin necesidad de alta precisión.
- **Especificaciones:** Los motores deberán cumplir con un torque nominal suficiente, precisión angular de 0,9° por paso, y una velocidad máxima adecuada para evitar sobrecargas y garantizar movimientos suaves.

Para el caso del robot heladero, se elegirían servomotores en las articulaciones más críticas, especialmente en el efector final y en puntos que requieran precisión y control de torque, ya que permiten un manejo preciso de las posiciones angulares, mientras que podrían utilizarse motores paso a paso en articulaciones secundarias donde el control de posición sea necesario pero no crítico, como el movimiento en trayectorias simples de desplazamiento o elevación del brazo.

9. Conclusión

Este proyecto ha permitido aplicar conceptos avanzados de robótica para el diseño, simulación y análisis de un robot serial de seis grados de libertad, enfocado en aplicaciones de manipulación en un entorno estructurado. A través de este trabajo se lograron aportes significativos que sientan una base sólida para futuras investigaciones y desarrollos.

Aportes particulares:

- Se logró implementar y validar los modelos de cinemática directa e inversa con precisión, utilizando simulaciones en MATLAB, lo cual permitió comprender y solucionar los desafíos asociados a la manipulación en espacios restringidos.
- Se generaron trayectorias optimizadas que el robot puede seguir de manera suave y eficiente, minimizando el desgaste mecánico y garantizando la precisión necesaria para la manipulación de helado, lo cual tiene un enfoque práctico en la industria alimentaria.
- Se identificaron y analizaron las singularidades del robot, proponiendo estrategias para evitarlas y mejorar la capacidad de movimiento sin perder eficiencia.
- Se propuso la selección e integración de sensores y actuadores adecuados que garantizan el control preciso del sistema y la seguridad en su operación.

Posibilidad de trabajos futuros:

- Optimización del algoritmo de cinemática inversa para reducir el tiempo de cómputo y mejorar la estabilidad del robot, particularmente en configuraciones cercanas a puntos singulares.
- Integración de sistemas de visión artificial para la percepción del entorno, permitiendo ajustar dinámicamente las trayectorias del robot y mejorar su capacidad de manipulación en entornos cambiantes.
- Extender el análisis al control dinámico del robot, integrando el diseño de controladores que optimicen la respuesta del sistema, considerando la inercia y la interacción con el material manipulado, como el helado.
- Implementar una versión física del robot, validando las simulaciones y optimizando el diseño con base en los resultados obtenidos en un entorno real.

En resumen, este proyecto ofrece un aporte significativo a la robótica aplicada en la industria alimentaria, demostrando cómo el diseño y análisis adecuado de modelos, junto con el uso de simulaciones, puede mejorar la precisión y eficiencia en tareas de manipulación automatizadas. Los trabajos futuros propuestos permitirán llevar este proyecto a un nivel práctico, mejorando aún más sus capacidades y efectividad.

10. Referencias

Referencias

[1] Craig, J. J. (2005). *Introduction to Robotics: Mechanics and Control*. Pearson.

[2] Corke, P. (2011). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer.

[3] MATLAB Documentation. (2024). *Robotics Toolbox Documentation*. Disponible en: `https://www.mathworks.com`

[4] Corke, P. (2024). *Robotics Toolbox for MATLAB*. Disponible en: `https://petercorke.com/toolboxes/robotics-toolbox/`

[5] Zhao, G., Sun, Y., Jiang, D., Liu, X., Tao, B., Jiang, G., Kong, J., Yun, J., Liu, Y., & Li, G. (2022). *A 7DOF redundant robotic arm inverse kinematic solution algorithm based on Bald Eagle swarm intelligent optimization algorithm*. Wuhan University of Science and Technology. Disponible en: `https://doi.org/10.21203/rs.3.rs-2333928/v1`

[6] Qin, L., Wei, X., Lv, L., Han, L., & Fang, G. (2023). *An Analytical Solution for Inverse Kinematics of SSRMS-Type Redundant Manipulators*. Sensors, 23, 5412. Disponible en: `https://doi.org/10.3390/s23125412`