BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN

"Planificación y seguimiento de trayectoria de un robot cooperativo (CoBot)"

T E S I S

Presentada para obtener el título de:
Maestro en Ciencias de la Electrónica Opción en Automatización

Presenta:

Lic. en Ing. Brandon Toxqui Gómez

Directores:

Dra. Amparo Palomino Merino (MCEA-FCE-BUAP)

Dr. César Martínez Torres (EDEI-UDLAP)

Puebla, México **Diciembre 2022**

* Becario CONACYT

Índice general

Resumen

La planificación de trayectoria puede plantearse de dos maneras, la primera es cuando el camino geométrico debe planificarse en un entorno estático y conocido, con obstáculos bien modelados, a esto se le conoce como trayectoria off-line, y la segunda de manera mas general se define para un sistema robótico que se encuentre en un entorno dinámico y desconocido a lo que se conoce como trayectoria online. El presente trabajo consiste en planificar y controlar la trayectoria de un robot cooperativo (CoBot) de manera off-line, con este objetivo se da un marco de referencia de los robots cooperativos y el papel que tienen actualmente en la industria, se da una descripción general del robot UR5 que es el robot con el que se va trabajar y se describen algunas aplicaciones en donde ha sido implementado tanto en el ámbito de la investigación como en el ámbito industrial. Posteriormente se estudian los algoritmos de planificación de trayectoria más utilizados, así como los controladores de trayectoria que puedan servir de base para implementar las trayectorias off-line. Posteriormente se desarrollan el modelo cinemático y dinámico del manipulador UR5, después se realiza una implementación en MATLAB de un sistema de control en lazo cerrado de los modelos del robot con diferentes estructuras de control como: el controlador Par Calculado, PD+ y RRMC. Finalmente se presenta el desarrollo de un nuevo esquema de regulación propuesto para robots manipuladores, el cual también es usado para realizar el control de posición del CoBot UR5.

Introducción

Una de las cuestiones de mayor estudio que engloban a la robótica y sus aplicaciones es la de la planificación de ruta [6] y planificación de trayectoria [14], en el caso de la primera se trata de generar una ruta o camino geométrico para el robot sin incluir una ley de tiempo, mientras que la segunda dota de una ley de tiempo a la ruta geométrica que ha sido generada por la primera. En concreto el objetivo de ambas es llevar al robot desde una configuración inicial hasta una configuración final sin tener colisiones con obstáculos que se encuentren en el camino o incluso evitar el choque con los eslabones del propio robot.
La planificación de trayectoria en su modo más simple, es cuando el camino geométrico debe planificarse en un entorno estático y conocido, a esto se le conoce como trayectoria off-line, de manera general se puede definir para cualquier sistema robótico que se encuentre en un entorno dinámico y desconocido, lo que se conoce como trayectoria online. Generalmente se suelen clasificar a los algoritmos de planificación de ruta conforme a la metodología que usan para generar el camino geométrico [6]:

- Técnicas de hoja de ruta o mapa de caminos [15].
- Algoritmos de descomposición celular [1].
- Métodos de potencial artificial [1].

En cambio los algoritmos para la planificación de trayectorias suelen ser clasificados por la función que optimizan:

- Tiempo de ejecución mínimo [16].
- Energía mínima [11].
- Jerk mínimo [6]

Cobots

Dentro de la industria automatizada, uno de los principales pilares que tiene es la robótica, de hecho podemos decir que se ha vuelto indispensable el uso de robots dentro de los procesos industriales, su uso ha hecho que se mejoren varios aspectos dentro de las industrias como: la disminución de los tiempos de producción, la calidad de la

manufactura y la disminución de riesgos para los humanos al poder realizar tareas en ambientes extremos que resultarían perjudiciales para los trabajadores [5].
Un eje principal de la industria 4.0 es la robótica cooperativa, la cual consiste en diseñar y programar robots no solamente para que trabajen en zonas aisladas bajo restricciones de seguridad, sino para que los robots trabajen "hombro a hombro" con los operadores.
Los robots colaborativos(Cobots) son una categoría de robots diseñados para trabajar en conjunto con los humanos. Combinando los puntos fuertes del robot como precisión y fuerza con la destreza y capacidad de resolución de problemas del ser humano, es posible realizar tareas que no pueden ser completamente automatizadas y mejorar la calidad de producción y las condiciones de trabajo para los empleados [9].

Entre las principales características que tienen los Cobots encontramos que [17]:

- Son fáciles de programar: no se necesita tener un conocimiento profundo de programación para poder configurar y manejar este tipo de robots.
- Su instalación es rápida: los elementos necesarios para la puesta en marcha de un Cobot están diseñados de tal forma que el montaje sea una tarea sencilla y que se pueda realizar en algunas horas.
- Son flexibles: el ajuste para que puedan cambiar su diseño de producción se realiza de manera sencilla con componentes que ya se encuentran diseñados, además son ligeros y pequeños, por lo que son fáciles de mover y de reasignarles tareas.
- Son seguros: su sistema de diseño esta basado en la eliminación y disminución de riesgos y debido a las tareas colaborativas que tienen que desempeñar tienen limitaciones de potencia y fuerza.

Descripción general del Cobot

El robot UR5 es un robot colaborativo industrial de peso ligero de la marca Universal Robots, en la figura1 (Manual UR5, 2015) podemos visualizar la estructura general del robot. Este robot se usa generalmente para automatizar tareas como "pick and place", paletizado, pruebas y selección. Debido a su tamaño es posible reubicarlo fácilmente ya que no se requiere mas infraestructura para hacerlo funcionar más que su caja de control y su consola de programación.

El movimiento del robot se puede programar mediante dos maneras, la primera, a través de una interfaz gráfica de usuario (IGU) llamada PolyScope que nos permite manejar el brazo robótico,configurar la caja de control y escribir y ejecutar los programas realizados, esta herramienta es muy intuitiva y permite a cualquier persona sin experiencia aprender fácilmente como generar los movimientos para el robot.

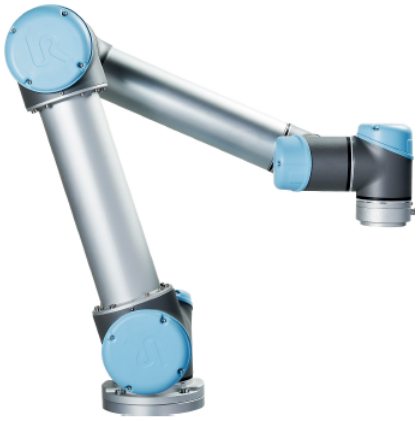

Figura 1: Robot UR5 [12]

La segunda opción es a través de su consola de programación llamada URSCRIPT, la cual viene incluida cuando se adquiere el robot. El ambiente de esta consola es muy parecido a PolyScope por lo que su manejo también es sencillo de aprender. Las principales características técnicas del robot UR5 son las siguientes [12]:

- Tiene 6 grados de libertad.
- Todas sus articulaciones son de tipo rotacional, las cuales son base(A), hombro(B), codo(C) y 3 muñecas(D, E, F), en la figura 2 se pueden visualizar.
- Sus articulaciones pueden girar 360°, la velocidad a la que pueden trabajar es de hasta 180°/s.
- La velocidad lineal máxima de la herramienta es de 1m/s.
- Su radio de trabajo de 850mm/33.5 in.
- La caja de control cuenta con puertos de entradas y salidas que nos permiten conectar sensores para diferentes aplicaciones así como señales para activar algún actuador.
- El material del cual se encuentra hecho es de Aluminio y Polipropileno.
- El rango de operación de temperatura va de 0° a 50°.
- La alimentación del robot es de 100-240 VAC a frecuencias de 50-60Hz.

Algunas aplicaciones con el UR5

Debido al nuevo concepto de robótica colaborativa con el que trabaja el UR5, las posibilidades de aplicación que se han hecho en la industria son muy variadas, sin

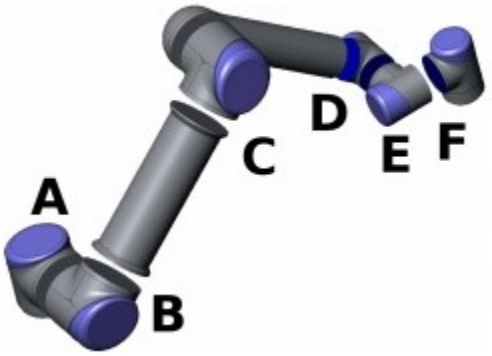

Figura 2: Articulaciones en el UR5 [12]

embargo debido a que una de las principales características de estos robots es su seguridad hemos encontrado que se están haciendo estudios de factibilidad para que estos robots puedan ayudar en tareas como la rehabilitación física de personas que han sufrido lesiones del sistema musculo-esquelético, un ejemplo lo encontramos en [10].
Una aplicación industrial para este robot se ha desarrollado en [11] donde se desarrolla un controlador basado en la cinemática del robot, para pintura en aerosol autónoma. En este trabajo a través de la observación de pintado por spray, se concluye que un pequeño ángulo entre la dirección de pulverización y la superficie normal no afecta la calidad del trabajo de pintura. Es decir que no necesariamente la válvula de salida del spray tiene que ser normal a la superficie a pintar. Este pequeño cambio logra que se reduzca el consumo de energía.
También encontramos aplicaciones del UR5 en el campo de la soldadura en [13] se optimiza el tiempo de la trayectoria articular en un entorno donde el robot realiza soldadura por fricción-agitación en ingles Friction Stir Welding (FSW), un tipo de soldadura usada para materiales blandos en la que una herramienta giratoria con un perno que perfora la unión de las piezas a soldar, eleva la temperatura por la fricción generada y después se hace un recorrido a lo largo de la unión de los materiales a unir lo que hace que el material caliente vaya de una cara a la otra y que al enfriarse realize la soldadura.
En la actualidad este robot ha sido implementado en empresas de toda índole,en su sitio de internet [17] se describen varios casos donde este robot ha tenido éxito al integrarse a las etapas de producción, por ejemplo se utilizó robots UR5 en Cascinia Italia para la preparación de los paquetes de huevos que resulta ser una tarea repetitiva y agotadora para los trabajadores. También la empresa Nordic Sugar utiliza al UR5 para automatizar el análisis de las muestras de remolacha de azúcar de sus materias primas, aquí los robots escanean códigos de barras y recogen los contenedores con el azúcar para el análisis de las escalas de filtración y los devuelven a la linea de producción. En SFGE (Scott Fetzer Electrical Group) se utiliza este robot par recoger una pieza eléctrica, la coloca sobre un soporte, toma un corta cables para cortar el exceso de cable de la pieza y luego coloca la pieza para que otro robot la ponga en

una cinta transportadora. En estos tipos de aplicaciones que se acaban de mencionar el UR5 destaca por su movilidad ya que en algunas de estas aplicaciones el robot esta montado en una mesa móvil y es llevado a diferentes espacios a desempeñar las tareas, pero también destaca por su seguridad ya que en todas estas tareas se encuentran operadores realizando tareas en conjunto con el robot o muy cerca de el.

Justificación

Teniendo en cuenta la definición de la robótica cooperativa emerge un nuevo problema que tiene que ver con la seguridad de los operadores que trabajan con Cobots, ya que estos podrían causar daños severos ante una colisión con el operador que lo maneja o inclusive sabemos que se puede producir un choque con el mismo robot. Por tales motivos es que en [6] se expone que la planificación de ruta y la planificación de trayectoria representan una cuestión crucial en la robótica y en general en el campo de la automatización, también se explica que la tendencia de los robots y las máquinas automáticas es funcionar a una velocidad cada vez más alta con el fin de lograr tiempos de producción mas cortos, el hecho de trabajar con una velocidad muy alta puede obstaculizar la precisión y repetitividad del robot ya que se requieren prestaciones extremas de los actuadores y del sistema de control.
Debido a las problemáticas anteriores es que es importante estudiar la planificación de trayectorias, con el objetivo de poder generar trayectorias que busquen obtener la mayor velocidad de movimiento, pero sin exponer al humano y a los sistemas que componen al robot, es decir evitar aceleraciones excesivas de los actuadores y vibraciones de la estructura mecánica, este tipo de trayectoria es definida como suave [6], recordemos que la suavidad de una función es una propiedad que se mide por el numero de derivadas continuas que tiene en algún dominio.
Como se viene mencionando puesto que en los procesos de producción el tiempo mínimo de los procesos [14] es uno de los principales aspectos a tener en cuenta se debe garantizar que la optimización de la trayectoria del robot este enfocada a establecer un tiempo de ejecución mínimo para su recorrido.

Objetivos

Objetivo general

Implementar un controlador para seguir trayectorias generadas offline en un brazo robótico UR5.

Objetivos particulares

- Estudiar los diferentes métodos de planeación de trayectorias para brazos robóticos industriales.
- Estudiar los controladores para seguimiento de trayectorias aplicados a brazos robóticos industriales
- Estudiar el modelo cinemático y dinámico del robot UR5.
- Seleccionar el método de planeación de trayectoria y el controlador adecuado para la aplicación.
- Simular los métodos seleccionados.
- Realizar una simulación de la planificación de trayectoria y el controlador utilizados mediante MATLAB.
- Publicación de resultados.

Capítulo 1

Estado del Arte

1.1. Algoritmos de planificación de trayectoria

La planificación óptima de la trayectoria consiste en generar una trayectoria para un robot desde una configuración inicial a una configuración final que satisface ciertos objetivos como lo son: tiempo de ejecución mínimo, jerk continuo y menor consumo de energía. En los sistemas de producción modernos el criterio mas ampliamente aplicado es el de tiempo mínimo en el que el manipulador viaja a través de la ruta especificada lo mas rápido posible [14].
En este trabajo se pretende desarrollar la planificación y seguimiento de trayectoria de un Cobot de 6 grados de libertad, mediante un algoritmo que nos permita generar la trayectoria de forma off-line pero que también pueda ser funcionable en trayectorias on-line, esto con la finalidad de que se puedan hacer futuros trabajos. Se pretende entender las ventajas y desventajas que conlleva trabajar con un algoritmo u otro. Una vez seleccionado el algoritmo de trabajo se buscara realizar una implementación en el robot físico si las condiciones lo permiten, en caso de no ser posible se implementara una simulación que nos permita evaluar el desempeño del algoritmo. Los algoritmos de planificación de trayectoria que se pretenden analizar son los siguientes:

1.1.1. Mapa de caminos u hoja de ruta

Lo que este enfoque busca es mapear todas las posibles conexiones que hay en el espacio de configuraciones, a través de curvas unidimensionales, este mapeo se conoce como mapa de caminos. Una vez obtenido el conjunto de rutas, la planificación consiste en vincular una configuración inicial y final del robot a través de algún camino factible establecido en la hoja de ruta. Dado este concepto es natural pensar que pueda existir mas de un camino que vincule las configuraciones inicial y final del robot, por lo que comúnmente se suele asociar una gráfica a esta técnica con el objetivo de optimizar algún parámetro que en la mayoría de los casos suele ser encontrar el camino más corto entre el estado inicial y final del robot [15]. La mayoría de los algoritmos de planificación de movimiento se basan en estructuras de datos llamados

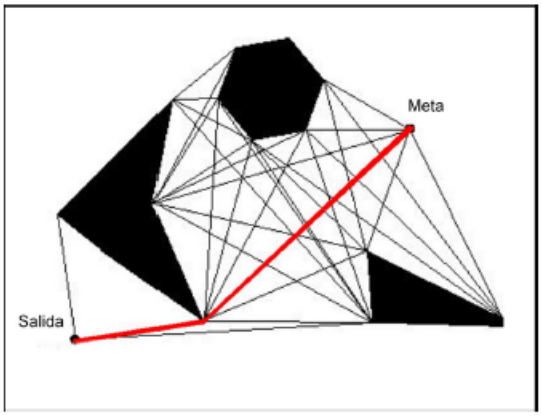

Figura 1.1: Gráfico de visibilidad [15]

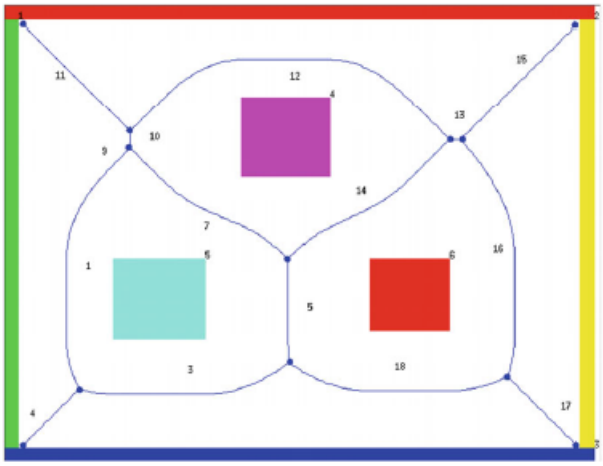

Figura 1.2: Diagrama de Voronoi [15]

gráficos de visibilidad u hoja de ruta. En esta se representan las posibles posiciones del robot en un espacio dado, los vértices de la hoja representan posibles posiciones y los bordes corresponden a la visibilidad directa entre nodos [4].
En la imagen 1.1 (Carbone and Gomez, 2015) se observa el gráfico de visibilidad introducido por Lozano-Perez el cual nos permite visualizar y generar un camino que pasa lo mas cerca posible de los vértices del obstáculo, por otra parte en la imagen 1.2 se visualiza al diagrama de Voronoi que contrario al gráfico de visibilidad nos permiten hallar un camino que se encuentra en la distancia máxima de los obstáculos [6].
En [15] se define un algoritmo de mapa de caminos semántico (SRM) para trayectorias online de la siguiente manera:
En un principio solo se tiene el nodo de inicio:

$$N = \{n_{start}\}, E = \emptyset \tag{1.1}$$

Los puntos de muestra dentro del alcance del sensor se utilizan como nodos candidatos

para la extensión del gráfico. Estos puntos se muestrean aleatoriamente. A un punto aleatorio $n_r \in N_{rand}$ se le adjunta una etiqueta semántica Ψ_r que representa el tipo de entorno detectado por el canalizador de mapeo semántico, que se encarga de encontrar los nodos $\hat{n}$ en el gráfico existente $\Re$ según:

$$\hat{n} = argmin \ \|n_i - n_r\|, \ N \sqsubset \Re \tag{1.2}$$

Donde N es el conjunto de nodos en el gráfico, donde la conexión de $n_i \in N$ a n_r no esta libre de colisiones. Si $\frac{N}{N'} = \emptyset$ significa que n_r no es un punto aleatorio factible porque no existe. Entonces el punto de muestra se descarta y otro punto de muestra en N_{rand} se tendrá en cuenta. Cuando n_r es factible se agrega al gráfico existente junto con el borde libre de colisiones que conecta n_r con su nodo vecino factible mas cercano $\hat{n}$ este proceso completa una iteración de la extensión del gráfico. Además para simplificar el gráfico y asegurar una distribución uniforme de los nodos generados, se realiza la siguiente acción [15]:
Para un nodo candidato n_r, si $\|n_r - n_j\| < \delta, \exists \ n_j \in N$, donde δ esta preestablecido según al tamaño del entorno entonces el nuevo punto n_r también sera descartado. A medida que el robot se mueve, se muestran continuamente nuevos puntos de muestra agregados a N_{rand} proporcionando mas nodos para la extensión del gráfico.
Este algoritmo es muy eficaz en escenarios con pocos obstáculos pero puede llegar a ser muy lento si el número de obstáculos aumenta, esto debido a que el coste computacional aumenta conforme aumentan los nodos que se tienen en el gráfico. Hay que tener en cuenta que para aplicar este método en trayectorias off-line se deben tener modelados los obstáculos previamente.

1.1.2. Algoritmo de descomposición celular

Consiste en dividir el espacio de configuraciones libre, en varias regiones llamadas celdas, una vez hecho esto si dos configuraciones se encuentran en una misma celda la ruta es fácil de generar, en caso de que no sea así se procede a generar un gráfico de conectividad que represente la relación entre celdas adyacentes. Los nodos de dicho gráfico representan a las celdas, así, dos nodos pueden unirse mediante una línea si y solo si las celdas son adyacentes, con esta idea el problema de planificación de camino se vuelve un problema de búsqueda de gráfico [6].
Existen dos subtipos principales:

Descomposición exacta

Se utiliza este término cuando la descomposición genera un conjunto de celdas cuya unión resulta idéntica al espacio de configuraciones libres de colisión. Es decir las celdas se definen de tal manera que se adapten a los obstáculos. De esta forma todo el espacio de configuración libre de colisión queda fragmentado en celdas trapezoidales o triangulares en su totalidad, ver imagen 1.3. A partir de aquí se genera el grafo de conectividad y se halla la secuencia de celdas en donde ha de estar contenida la

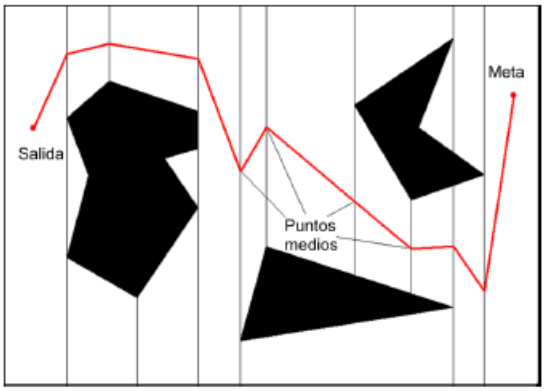

Figura 1.3: Descomposición en celdas exacta [16]

trayectoria. Para esta, basta con elegir los puntos medios de los segmentos adyacentes como puntos de paso. Trazando segmentos rectos entre los mismos se obtiene la solución [1].

Descomposición aproximada

En este tipo de descomposición se utilizan celdas de diseño predefinido, habitualmente rectangulares. Al descomponer todo el escenario en dichas celdas, algunas invadirán la región de los obstáculos ya sea parte de ellas o toda la celda. Estas celdas serán descartadas y las restantes serán las que conformen el grafo de adyacencia. Existen dos métodos muy conocidos: método de división en cuadrantes "quadtree" y el método de frente de ondas conocido también como "fast marching method" [1]. A continuación se describirá el método "quadtree"que ejemplifica de manera sencilla la forma en la que trabaja este algoritmo.
Este método parte de un escenario que se encuentra contenido en un rectángulo. A continuación se divide el escenario en rectángulos semejantes mediante el procedimiento de trazar mediatrices en la base y altura del rectángulo original. Así se obtienen cuatro rectángulos, estos se clasifican según su área, si esta contenida en los obstáculos (se les llamara "llenos") si están en el espacio de configuraciones libre C_{free} (serán "vacios") y si están en ambos serán "mixtos". Estos últimos volverán a sufrir el proceso de división utilizando el mismo sistema. De nuevo se agregaran las etiquetas a los rectángulos dependiendo en que área estén y serán los mixtos a los únicos que se les aplicara el proceso de fragmentación. En cada una de las etapas de división se actualiza una estructura de tipo árbol de grado 4. Es decir en cada nodo del árbol descienden otros cuatro. El nodo raíz representa al rectángulo original. Así, los únicos nodos que tienen descendientes son los mixtos, en la imagen 1.4 se puede visualizar este procedimiento.
Una de las principales ventajas de la descomposición celular es que podemos llevar el problema de movimiento del robot a un problema gráfico al igual que en la técnica de mapa de caminos, sin embargo los espacios de memoria y los recursos computacionales

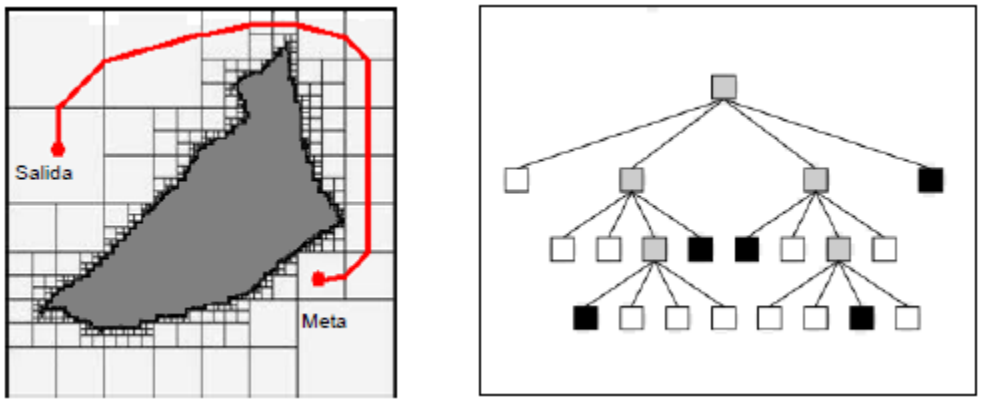

Figura 1.4: División en cuadrantes "Quadtree" [16]

requeridos por esta técnica aumentan considerablemente dependiendo del numero de obstáculos que se tengan y del nivel de discretización que se use para las celdas.

1.1.3. Método de potencial artificial

La idea consiste en considerar al robot como un punto en movimiento en el espacio de configuración, este punto se encuentra sujeto a un campo potencial atractivo generado por el objetivo y un campo potencial repulsivo generado por los obstáculos [6]. En la imagen 1.5 se puede visualizar al robot como un punto sometido a campos potenciales.

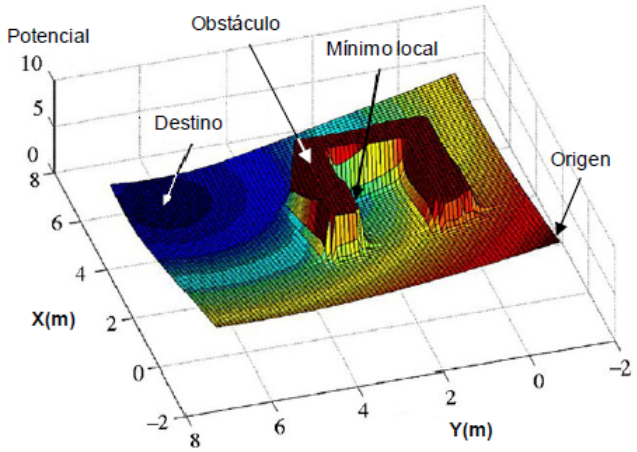

Figura 1.5: Campo de potenciales creado por un entorno dado [22]

La definición del campo potencial total U en [1] se define de la siguiente manera:

$$U(q) = U_{atr}(q) + U_{rep}(q) \tag{1.3}$$

donde U_{atr} es el campo que genera la atracción del destino y U_{rep} es el que representa el campo de repulsión en torno a los obstáculos.
El campo de potenciales responsable de la fuerza de atracción suele definirse como una función parabólica [1]:

$$U_{atr} = k\rho^2_{goal}(q) \tag{1.4}$$

Donde k es un factor de escala, q_{goal} es la configuración destino y $\rho_{goal}(q)$ es la distancia euclídea $\|q - q_{goal}\|$.
La fuerza de atracción derivada de este campo de potencial será:

$$\vec{F}_{atr} = -\vec{\nabla}U_{atr}(q) = -2k \cdot (q - q_{goal}) \tag{1.5}$$

Como se observa la fuerza de atracción consiste en un vector cuyo módulo es proporcional a la distancia entre el punto considerado y el objetivo, y cuya dirección apunta siempre hacia la configuración final. El campo de repulsión suele definirse como la suma de los campos generados por cada obstáculo:

$$U_{rep}(q) = \sum_{k=1}^{r} U_{CB_k}(q) \tag{1.6}$$

U_{CB_k} representa a los campos de cada obstáculo. Un ejemplo de campo potencial de repulsión asociado a un obstáculo podría ser el siguiente [1]:

$$U_{CB_k}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_k}(q) - \frac{1}{\rho_0}\right)^2 & si \quad \rho_k(q) \leq \rho_0 \\ 0 & si \quad \rho_k(q) > \rho_0 \end{cases} \tag{1.7}$$

Donde η es un factor de escala positivo, $\rho_k(q)$ resulta ser la distancia de q al obstáculo-C identificado con el valor de k. Por ejemplo:

$$\rho_k(q) = \min_{q' \in CB_k} \|q - q'\| \tag{1.8}$$

Por ultimo ρ_0 es una constante positiva denominada "distancia de influencia del obstáculo". Así el campo de potencial repulsivo asociado al obstáculo correspondiente es siempre positivo o nulo, y tiende a infinito conforme se acerca al obstáculo-C y se hace nulo más allá de ρ_0. La fuerza de repulsión asociada a cada obstáculo resulta pues [1]:

$$\vec{F}_{CB_k}(q) = \vec{\nabla}U_{CB_k}(q) = \begin{cases} \eta\left(\frac{1}{\rho_k}(q) - \frac{1}{\rho_0}\right)\frac{1}{\rho^2_k(q)}\vec{\nabla}\rho_k(q) & si \quad \rho_k(q) \leq \rho_0 \\ 0 & si \quad \rho_k(q) > \rho_0 \end{cases} \tag{1.9}$$

El conjunto de todas estas fuerzas de repulsión sería:

$$\vec{F}_{rep}(q) = \sum_{k=1}^{r} \vec{F}_{CB_k}(q) \tag{1.10}$$

Y por lo tanto el vehiculo se movera en función de la siguiente fuerza resultante:

$$\vec{F}_{res} = \vec{F}_{rep} + \vec{F}_{atr} \tag{1.11}$$

Algunos inconvenientes con este algoritmo es que presenta es que cuando el sistema alcanza un mínimo local la resultante de todas las fuerzas es nula sin embargo la configuración no es la que se desea alcanzar. Para resolver esto se recurre a métodos de generación aleatoria que permitan abandonar el mínimo para otra vez volver a aplicar el método del gradiente. Un inconveniente más se presenta al momento de generar los caminos aleatorios ya que son sencillos de calcular pero generan una densidad de probabilidad no uniforme [1]. Otra de las soluciones es utilizar funciones que no tengan mínimos locales. Hay que decir que este algoritmo se puede implementar en trayectorias on-line.

Posteriormente una ves seleccionado el método de planificación de trayectoria se buscara optimizar la trayectoria mas corta con el objetivo de establecer un tiempo de ejecución mínimo, para ello se piensa utilizar el ya conocido algoritmo de Dijkstra que nos permite obtener el camino mas corto en un nodo inicial que se conecta a otros nodos mediante aristas que definen lo longitud entre nodos adyacentes. Sin embargo hoy en día con ayuda de las computadoras es posible optimizar distintos tipos de funciones a través de algoritmos que simulan el entorno natural donde vivimos, un par de ejemplos los encontramos a continuación.

1.1.4. Algoritmo genético

El algoritmo genético es un algoritmo bioinspirado y se basa en la selección y mecanismo genético natural del mundo biológico, utilizando el principio de supervivencia del mas apto es capaz de producir una solución óptima aproximada, es un algoritmo de búsqueda aleatorio de computo científico. En [8] se utiliza una curva polinomial para conectar los puntos de ruta adyacentes, de modo que la trayectoria conjunta es suave, una ves realizado esto se establece la función fitness y las restricciones para implementar el algoritmo. Su diagrama de flujo de funcionamiento se muestra en la figura 1.1.4.
Aquí bajo la premisa de que las 6 articulaciones deben satisfacer sus restricciones cinemáticas es decir el limite superior de velocidad, la aceleración y el jerk no se deben exceder. El nodo de la serie temporal con el tiempo total mas corto debe ser obtenido como objetivo. Esto es un problema de optimización con restricción no lineal. La función objetivo de la optimización es:

$$f(x) = min \sum_{i=1}^{n-1} X_i \tag{1.12}$$

Donde, x_i es el intervalo de tiempo entre dos puntos de ruta adyacentes y $x_i = t_i{+}1{+}t_i$, $f(x)$ es el tiempo total para pasar a través de los puntos de ruta.

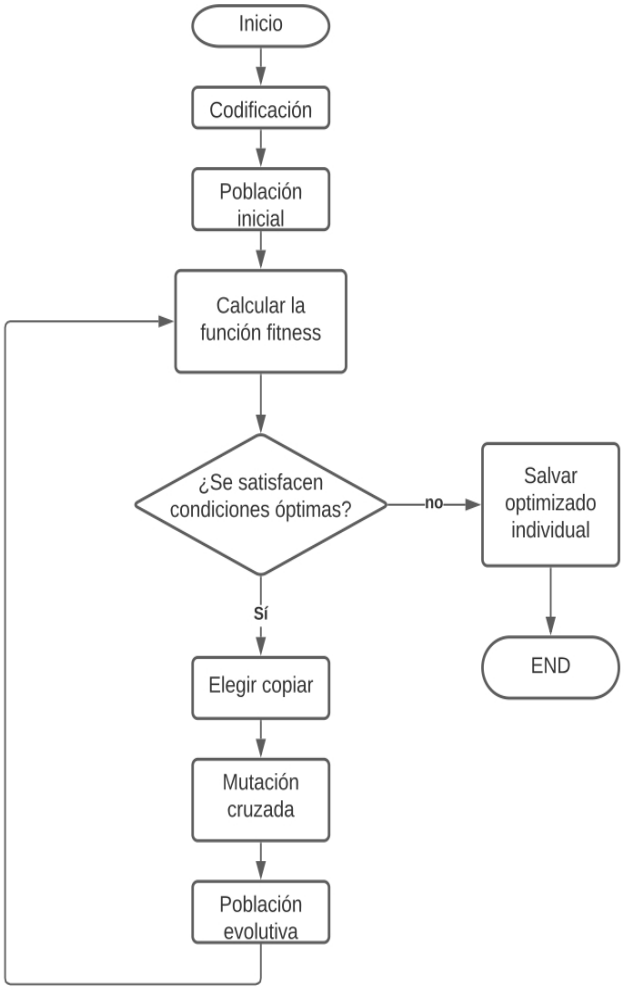

Figura 1.6: Principio de trabajo del algoritmo genético

1.1.5. Algoritmo basado en enjambre de pollos

En [14] se utiliza un nuevo algoritmo bioinspirado llamado "optimización de enjambres de pollo" con las siglas (CSO) que se aplica para optimizar la trayectoria de los manipuladores robóticos. La trayectoria que se describe se construye mediante B-splines de tercer orden cumpliendo las restricciones cinemáticas. Además en esta implementación se utiliza el tiempo mínimo de viaje como función objetivo para obtener la trayectoria óptima.

Definición del CSO

Un enjambre de pollos se divide en varios grupos según la aptitud de los pollos. Cada grupo comprende un gallo dominante, varias gallinas y pollitos. Los gallos RN tienen

los mejores valores de aptitud física y cada uno es designado como un gallo cabeza de un grupo, los pollitos CN son los que tienen peores valores de aptitud física. El resto de los pollos HN actúan como gallinas, de los cuales los pollos MN que alimentan a los pollitos son madres gallina. Cada gallina elige un grupo al azar para unirse y cada pollito elige una gallina al azar para seguir y forrajear. Los gallos, gallinas y pollitos tienen diferentes habilidades de búsqueda y por lo tanto actualizan sus posiciones conforme a su respectivas leyes de forrajeo.

Ley de forrajeo de los gallos

Los gallos tienen mas poder y pueden dominar a los débiles por lo que tienen prioridad para el acceso a los alimentos. La actualización de su posición esta influenciada por otro gallo seleccionado al azar. Su ley de alimentación x_i se puede formular de la siguiente manera:

$$x_{i,j}^{t+1} = x_{i,j}^{t} * \left(1 + G(0, \sigma^2)\right) \tag{1.13}$$

$$\sigma^2 = \begin{cases} 1 & si \quad f_i \leq f_k, \\ exp\left(\frac{f_k - f_i}{|f_i| + \epsilon}\right) & de\ otra\ manera, \end{cases} \quad k \in [1, N], k \neq i \tag{1.14}$$

Donde $G(0, \sigma^2)$ es una distribución normal con media 0 y desviación estándar σ, ϵ es una constante mínima para evitar la división por cero, k es un índice de gallo que es seleccionado al azar y no igual a i. f representa la aptitud del gallo correspondiente.

Ley de alimentación de las gallinas

Las gallinas tienen la característica de alimentación más compleja en el enjambre de pollos. Su capacidad de forrajeo es menor a la de los gallos. Por lo general se alimentan siguiendo a los gallos del grupo al que pertenecen y roban comida encontrada por otros pollos al azar. Su actualización de posición se ve afectada por otros gallos y gallinas. La ley de alimentación de la gallina x_i puede formularse de la siguiente manera:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + c_1 * rand(0,1) * (x_{r1,j}^{t} - x_{i,j}^{t}) + c_2 * rand(0,1) * (x_{r2,j}^{t} - x_{i,j}^{t}) \tag{1.15}$$

$$c_1 = \frac{f_i - f_{r1}}{|f_i| + \epsilon} \tag{1.16}$$

$$c_2 = exp(f_{r2} - f_i) \tag{1.17}$$

Donde $rand(0,1)$ es un numero aleatorio en el rango [0,1], x_{r1} es el gallo de el ith grupo, c_1 el factor de influencia debido a x_{r1}, x_{r2} es otro pollo (gallo o gallina) que es seleccionado aleatoriamente y no es x_{r1}, c_2 es el factor de influencia de x_{r2}, f_{r1} y f_{r2} son los valores de aptitud de x_{r1} y x_{r2} respectivamente.

Ley de alimentación de los pollitos

Los pollitos tienen la peor capacidad de forrajeo. Ellos forrajean alrededor de sus madres gallinas. En consecuencia se inclinan a explotar el espacio local. Su actualización de posición se ve afectada por sus respectivas madres gallinas. La ley de forrajeo de los pollitos se puede formular de la siguiente manera:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + F * (x_{m,j}^{t} - x_{i,j}^{t}) \tag{1.18}$$

Donde x_m es la mama gallina x_i, el parámetro F se adopta para representar la siguiente situación de $x_i's$ respecto a su gallina madre y se obtiene en el rango de [0,2] aleatoriamente.

1.2. Controladores de trayectoria

El problema de control de movimiento es uno de los temas más importantes en robótica. Recientemente ha recibido la atención de la comunidad científica y como resultado se han reportado en la literatura diversos controladores; entre los que han mostrado tener mejor desempeño se encuentran: control par calculado, control PD, PD_+, PD con precompensación y PD con compensación [2].
Los algoritmos de control de trayectoria incluyen la dinámica completa del robot manipulador en la estructura matemática del controlador, es decir se basan en el modelo dinámico del robot. La exactitud, desempeño y robustez de estos controladores dependen del grado de precisión con que se conozcan los parámetros dinámicos que describen el modelo. El control de trayectoria puede plantearse en los siguientes términos [2]: Considérese el modelo dinámico del robot manipulador de n gdl:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + B\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \tag{1.19}$$

además sean $\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d \in \mathbb{R}^n$ un conjunto de funciones vectoriales acotadas que representan la posición, velocidad y aceleración deseadas, respectivamente.
El problema de control de trayectoria o control de movimiento consiste en determinar una función vectorial $\boldsymbol{\tau}$ de tal forma que las posiciones y velocidades asociadas a las articulaciones del robot sigan con exactitud a las posiciones y velocidades deseadas, respectivamente [2].
Con lo anterior se deduce que el objetivo de control de movimiento consiste en encontrar $\boldsymbol{\tau}$ tal que:

$$\lim_{t\to\infty} \begin{bmatrix} \tilde{\boldsymbol{q}} \\ \dot{\tilde{\boldsymbol{q}}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n} \tag{1.20}$$

donde el error de posición $\tilde{\boldsymbol{q}} = \boldsymbol{q}_d(t) - \boldsymbol{q}(t)$ representa la diferencia entre la posición o trayectoria deseada y la posición actual. El error de velocidad $\dot{\tilde{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}(t)$ representa la diferencia entre la velocidad de movimiento deseada y la velocidad articular actual.

La ecuación en lazo cerrado que determina el problema del control de movimiento queda expresada en variables de estado articulares de la siguiente forma [2]:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} = \begin{bmatrix}\dot{\tilde{\boldsymbol{q}}}\\ M(\boldsymbol{q})^{-1}[\tau(K_p, K_v, \tilde{\boldsymbol{q}}, \dot{\tilde{\boldsymbol{q}}}, M(\boldsymbol{q}), C(\boldsymbol{q}, \dot{\boldsymbol{q}}), B, \boldsymbol{g}(\boldsymbol{q})) - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - B\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})]\end{bmatrix} \tag{1.21}$$

la ecuación resultante en lazo cerrado 1.21 es una ecuación diferencial ordinaria no lineal y no autónoma. La incorporación de la trayectoria $\boldsymbol{q}_d(t)$ es lo que determina que la naturaleza de la ecuación diferencial en lazo cerrado sea no autónoma.

1.2.1. Control PD+

El control PD+ con compensación de gravedad denotado por τ_{pd+} es un algoritmo de control que incluye control proporcional del error de posición, control proporcional del error de velocidad y la dinámica completa del robot. En la estructura de este esquema de control también se involucra la trayectoria de seguimiento, velocidad y aceleración deseada [2]. La expresión del PD+ es:

$$\boldsymbol{\tau}_{pd} = K_p\tilde{\boldsymbol{q}} + K_v\dot{\tilde{\boldsymbol{q}}} + M(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_d + B\dot{\boldsymbol{q}}_d + \boldsymbol{g}(\boldsymbol{q}) \tag{1.22}$$

Como se observa en a ecuación 1.22 el controlador requiere de la dinámica completa del manipulador, lo que significa que todos los parámetros del robot son conocidos. La ecuación en lazo cerrado que involucra el modelo dinámico 1.19 y el esquema de control PD+ expresado en variables de estado para el control de trayectoria es [2]:

$$\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} = \begin{bmatrix}\dot{\tilde{\boldsymbol{q}}}\\ -M^{-1}(\boldsymbol{q})[K_p\tilde{\boldsymbol{q}} + K_v\dot{\tilde{\boldsymbol{q}}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\tilde{\boldsymbol{q}}} + B\dot{\tilde{\boldsymbol{q}}}]\end{bmatrix} \tag{1.23}$$

$$= \begin{bmatrix}\dot{\tilde{\boldsymbol{q}}}\\ -M^{-1}(\boldsymbol{q}_d - \tilde{\boldsymbol{q}})[K_p\tilde{\boldsymbol{q}} + K_v\dot{\tilde{\boldsymbol{q}}} + C(\boldsymbol{q}_d(t) - \tilde{\boldsymbol{q}}(t), \dot{\boldsymbol{q}}_d(t) - \dot{\tilde{\boldsymbol{q}}})\dot{\tilde{\boldsymbol{q}}} + B\dot{\tilde{\boldsymbol{q}}}]\end{bmatrix} \tag{1.24}$$

la ecuación resultante es una ecuación diferencial no lineal de primer orden, no autónoma y cuyo punto de equilibrio es el origen $[\tilde{\boldsymbol{q}}, \dot{\tilde{\boldsymbol{q}}}]^T = \boldsymbol{0} \in \mathbb{R}^{2n}$

1.2.2. Control Par Calculado

Es una estructura de control de trayectoria que emplea la dinámica de compensación en el lazo de retroalimentación para linealizar y desacoplar la dinámica no lineal del robot manipulador, por lo que tiene el atributo de obtener una ecuación en malla cerrada lineal en términos de las variables de estado. El control par calculado tiene la siguiente expresión [2]:

$$\boldsymbol{\tau}_{pc} = M(\boldsymbol{q})[\ddot{\boldsymbol{q}}_d + K_p\tilde{\boldsymbol{q}} + K_v\dot{\tilde{\boldsymbol{q}}}] + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + B\dot{\boldsymbol{q}} \tag{1.25}$$

La ecuación en malla cerrada se obtiene sustituyendo la acción de control $\boldsymbol{\tau}_{pc}$ en la ecuación del modelo dinámico del robot 1.19 y bajo la suposición de que existe una

cancelación exacta del modelo dinámico que interviene en el lazo de realimentación. Entonces el control par calculado hace un desacoplamiento dinámico en todas las articulaciones del robot resultando el siguiente sistema lineal [2]:

$$\ddot{\tilde{\boldsymbol{q}}} + K_v\dot{\tilde{\boldsymbol{q}}} + K_p\tilde{\boldsymbol{q}} = \mathbf{0} \tag{1.26}$$

esta ecuación también puede escribirse en variables de estado, como a continuación [2]:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} = \begin{bmatrix}0 & I\\ -K_p & -K_v\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} \tag{1.27}$$

donde $I \in \mathbb{R}^{n\times n}$ es la matriz identidad. También se puede observar que la ecuación 1.27 es una ecuación diferencial lineal y autónoma, con un único estado de equilibrio en $[\tilde{\boldsymbol{q}}^T\ \dot{\tilde{\boldsymbol{q}}}^T]^T = \mathbf{0} \in \mathbb{R}^{2n}$, este estado de equilibrio tiene estabilidad exponencial si las matrices K_p y K_v son simétricas y definidas positivas.

1.2.3. Control PID

El controlador PID no representa un esquema nuevo de control, mas bien es la versión modificada del control proporcional derivativo PD que pretende subsanar la deficiencia del error en régimen estacionario. Con esta finalidad se incorpora en el algoritmo de control PD la acción de control integral, resultando el siguiente esquema proporcional integral derivativo [2]:

$$\boldsymbol{\tau} = K_p\tilde{\boldsymbol{q}} - K_v\dot{\boldsymbol{q}} + K_i\int_0^t \tilde{\boldsymbol{q}}(\mu)d\mu + \boldsymbol{g}(\boldsymbol{q}) \tag{1.28}$$

donde $K_p, K_v, K_i \in \mathbb{R}^{n\times n}$ son matrices definidas positivas y se les denomina ganancias proporcional, derivativa e integral respectivamente. Todas son matrices definidas positivas.
La acción de control PID introduce una nueva variable de estado que se puede denotar por v, la cual esta relacionada con la derivada temporal del error de posición $\dot{\boldsymbol{v}} = \tilde{\boldsymbol{q}}$. La ecuación en lazo cerrado en términos de las variables de estado $[\boldsymbol{v}, \tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}]^T$ adquiere la siguiente forma:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{v}\\ \dot{\tilde{\boldsymbol{q}}}\\ \dot{\boldsymbol{q}}\end{bmatrix} = \begin{bmatrix}\tilde{\boldsymbol{q}}\\ -\dot{\boldsymbol{q}}\\ M(\boldsymbol{q})^{-1}[K_p\tilde{\boldsymbol{q}} - K_v\dot{\boldsymbol{q}} + K_i\boldsymbol{v} - C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - B\dot{\boldsymbol{q}}]\end{bmatrix} \tag{1.29}$$

La ecuación 1.29 es una ecuación diferencial autónoma cuyo único punto de equilibrio es el origen $[\boldsymbol{v}^T, \tilde{\boldsymbol{q}}^T, \dot{\boldsymbol{q}}^T]^T = \mathbf{0} \in \mathbb{R}^{3n}$.

Una ves propuestos los algoritmos a utilizar se llevara a cabo mediante el software de MATLAB la implementación de los bloques de control y seguimiento de trayectoria para el robot, hecho esto se realizara una simulación en el software Gazebo que nos

permita verificar la funcionalidad de las propuestas hechas. Además este trabajo se implementara físicamente en el robot UR5, posteriormente si se cuenta con tiempo suficiente se implementara una aplicación de bin picking con un sistema kinect con el que también se cuenta. Finalmente hay que agregar que ya se tomo un curso online de Universal Robots con la finalidad de contar con las nociones básicas de como funcionan y como se programan este tipo de robots.

Capítulo 2

Modelado

En el área científica los problemas son formulados y resueltos mediante modelos que representan a los sistemas reales. El problema de planificación de trayectoria para un robot manipulador también se encuentra planteado en términos de modelos. El primero de ellos es llamado cinemática directa que consiste en hallar una función que dados los parámetros geométricos del robot y el valor de sus articulaciones $\boldsymbol{q}$ nos retorne la posición y orientación del extremo final del robot. Consecuentemente surge el planteamiento de la cinemática inversa, que consiste en hallar una función que dada la posición y orientación del extremo final del robot esta nos retorne el valor en el cual se tengan que posicionar las articulaciones del robot para alcanzar tal posición y orientación del extremo final. Los dos modelos mencionados anteriormente como su análisis y nombre lo indica son modelos geométricos es decir no toman en cuenta las causas que originan el movimiento, es por ello que existe el modelo dinámico pues este toma en cuenta toda la interacción mecánica que el robot tiene con el espacio, puesto que dentro de su movimiento existen varios fenómenos interesantes como lo es la inercia y el efecto Coriolis. El modelo dinámico de un manipulador de n grados de libertad es necesario para poder implementar algoritmos de control que nos permitan solucionar problemas como el control de posición o regulación y control de trayectoria. Debido a lo anterior es importante contar con el modelo dinámico numérico del manipulador, es por ello que las siguientes secciones se dedican a obtener el modelo cinemático y dinámico del robot UR5.

2.1. Cinemática Directa

El estudio de la cinemática directa de robots manipuladores proporciona los elementos necesarios para analizar y diseñar el desplazamiento de trayectorias del robot, así como la orientación de la herramienta de trabajo [2]. En otras palabras se puede describir como un análisis únicamente geométrico de la estructura del manipulador que describe su desplazamiento en el espacio. Las herramienta mas utilizada que nos permite obtener el modelo de cinemática de un robot manipulador es el análisis geométrico y básicamente lo que se hace es establecer un sistema de referencia fijo y establecer sistemas de referencia móviles en las articulaciones del robot, luego mediante análisis algebraico-geométrico se establecen las relaciones que existen entre los marcos de referencia móviles y el marco de referencia fijo. A instancia se puede notar que este método crece en dificultad conforme aumenta el número de grados de libertad del robot, sin embargo en 1955 Jaques Denavit y Richard S. Hartenberg en 1955 presentaron un procedimiento para obtener una representación mínima de la orientación y traslación de robots manipuladores a través de 4 parámetros dentro de una transformación homogénea. Esta metodología sera la que se va a utilizar para obtener el modelo cinemático del robot y por lo tanto se explicara mas adelante con detalle.

En cuestión de aplicación práctica la **cinemática directa** es una función vectorial $\boldsymbol{f}_R(l_i, \boldsymbol{q})$ que relaciona las coordenadas articulares $\boldsymbol{q} \in \mathbb{R}^n$ y las propiedades geométricas del sistema mecánico l_i con las coordenadas cartesianas $[x, y, x]^T \in \mathbb{R}^3$ del robot y la orientación $[\theta.\phi, \psi] \in \mathbb{R}^3$ de la herramienta de trabajo colocada en el extremo final [2]. En otras palabras la cinemática directa es un mapeo vectorial $\boldsymbol{f}_R : \mathbb{R}^n \to \mathbb{R}^m$ tal que:

$$\begin{bmatrix} x \\ y \\ z \\ \theta \\ \phi \\ \psi \end{bmatrix} = \boldsymbol{f}_R(l_i, \boldsymbol{q}) \tag{2.1}$$

donde n indica el número de grados de libertad y la dimensión del vector de coordenadas articulares $\boldsymbol{q}$, m es la dimensión conjunta de las coordenadas cartesianas y la orientación de la herramienta de trabajo.

2.1.1. Estructura del robot

Un robot manipulador industrial se encuentra constituido por una serie consecutiva de eslabones y articulaciones que forman una cadena cinemática abierta, a su ves esta conforma la estructura mecánica básica de un robot manipulador industrial.La cadena en cinemática abierta esta formada de la siguiente manera: la primera arti-

culación sirve para formar la base; a continuación siguen conexiones sucesivas entre articulaciones y eslabones, en el extremo final del último eslabón no hay articulación y generalmente se destina a colocar la herramienta de trabajo. La figura muestra esta descripción.

2.1.2. Nomenclatura para robots industriales

El eslabón

El robot manipulador cuenta con un elemento estructural bien definido al cual se le denomina **eslabón**, enlace o (*link*). Un eslabón es una estructura mecánica rígida que no sufre deformación al interactuar con el medio. Estos enlaces se encuentran conectados a través de articulaciones que son las que definen la forma de movimiento del eslabón. En concreto se cuenta con dos tipos de articulaciones: rotacional y prismática, una corresponde al movimiento rotatorio y la otra al movimiento lineal o prismático.

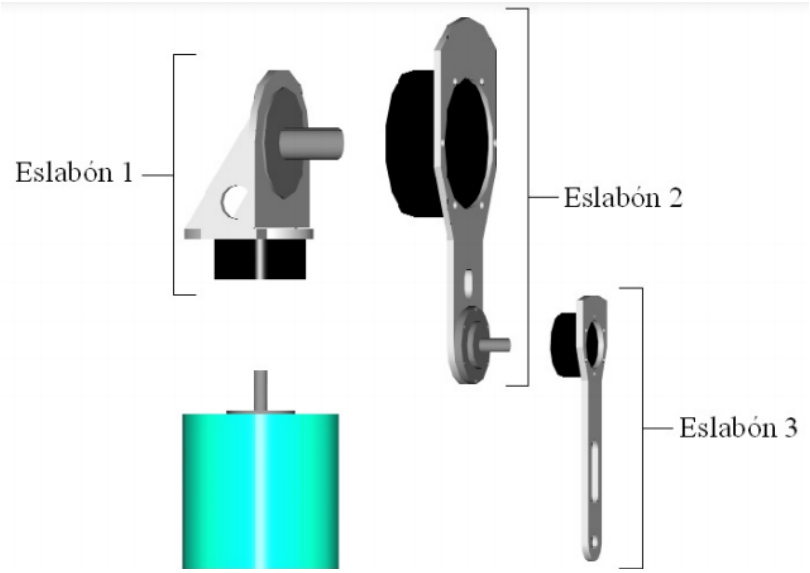

Figura 2.1: Morfología de un eslabón [2]

Espacio de trabajo (workspace)

El espacio de trabajo de un robot manipulador es el espacio o lugar en el que el robot puede realizar todos sus movimientos, este se encuentra determinado por la geometría del robot y el tipo de articulaciones que tenga (rotacionales o prismáticas). Otra forma de describir el *workspace* es como el conjunto de todas las configuraciones (*poses*) que el robot pueda adquirir sin colisionar con su entorno o con alguna parte de el mismo, esto debido a su geometría [24]. En la figura 2.2 podemos apreciar el espacio de trabajo de un robot.

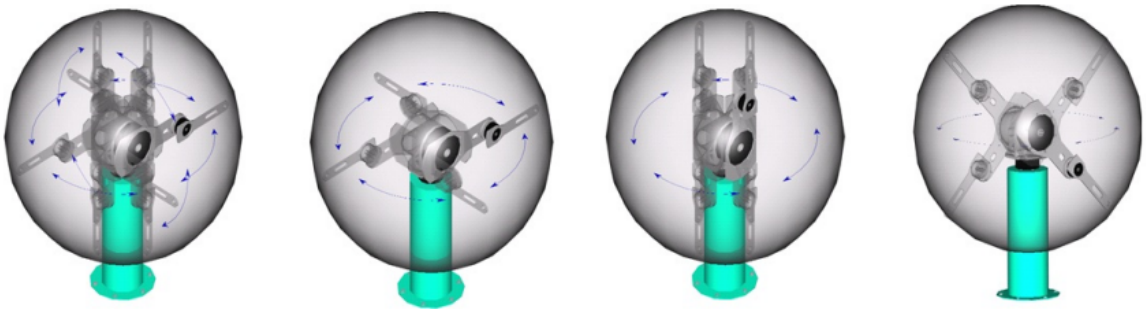

Figura 2.2: Espacio de trabajo o *workspace* [2]

Extremo final

El extremo final, en ingles *end-effector* es la parte final del último eslabón, su posición corresponde a un punto en el espacio representada por $[x, y, z]^T$ y su orientación se denota a través de los ángulos de Euler.

Figura 2.3: Extremo final en un robot FANUC [2]

2.1.3. Convención Denavit-Hartenberg

El algoritmo Denavit-Hartenberg es una herramienta ampliamente conocida en el área de ingeniería, ya que ofrece un procedimiento sencillo para obtener el modelo cinemático de un robot de n grados de libertad. El resultado de procesar el algoritmo Denavit Hartenberg da como resultado obtener la posición y orientación del extremo final en términos de transformaciones homogéneas [24]. El algoritmo Denavit-Hartenberg consiste en determinar una tabla de parámetros que están relacionados con las articulaciones y eslabones del robot descritos en marcos de referencia fijos respecto a un

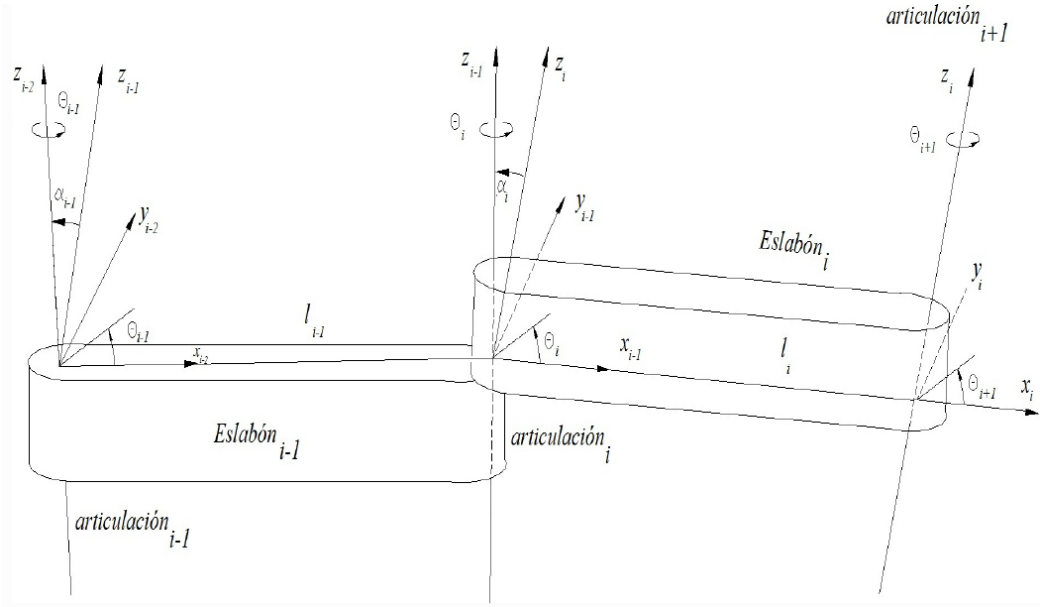

Figura 2.4: Convención Denavit-Hartenberg [2]

marco de referencia fijo. Este algoritmo se detalla en [2], y a continuación se describe una síntesis del procedimiento.

Algoritmo Denavit-Hartenberg

Los parámetros Denavit-Hartenberg se denotan como:

- θ_i para el tipo rotacional.
- Prismática o lineal por d_i; sin embargo este parámetro d_i también hace el papel de representar el ancho del servomotor de la articulación rotacional más el espesor de la placa metálica del eslabón, en este caso se denota por el símbolo β_i.
- La longitud del eslabón se representa con l_i.
- El ángulo de separación entre los ejes z_i y z_{i-1} se denota con α_i.
- El ángulo θ_i es el ángulo entre los ejes x_{i-1} y x_y medido alrededor del eje z_{i-1}.
- d_i es la distancia del origen del sistema de referencia $i-1$ a la intersección del eje x_i con el eje z_{i-1}. Su medición se realiza a lo largo del eje z_{i-1}.
- El parámetro l_i se define como la distancia a lo largo del eje x_i desde el origen del sistema de referencia coordenado $i-1$ hasta la intersección del eje z_{i-1} con el eje x_i.

- El ángulo entre los ejes z_i y z_{i-1} se denota por α_i, su medición es respecto a un plano normal a x_i. Una medición de ángulo positivo para α_i se toma en dirección del eje z_{i-1} hacia z_i.

La representación DH es a través del producto de cuatro transformaciones básicas:

$$\begin{aligned} H_{i-1}^{i} &= H_{R_{z_{i-1}}}(\theta_i)H_{T_{z_{i-1}}}(d_i;\beta_i)H_{T_{x_{i-1}}}(l_i)H_{R_{x_{i-1}}}(\alpha_i) \\ &= \begin{bmatrix} cos(\theta_i) & -sen(\theta_i) & 0 & 0 \\ sen(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i(\beta_i) \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \begin{bmatrix} 1 & 0 & 0 & l_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sen(\alpha_i) & 0 \\ 0 & sen(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} cos(\theta_i) & -sen(\theta_i)cos(\theta_i) & sen(\theta_i)sen(\alpha_i) & l_i cos(\theta_i) \\ sen(\theta_i) & cos(\theta_i)cos(\alpha_i) - cos(\theta_i)sen(\alpha_i) & l_i sen(\theta_i) & \\ 0 & sen(\alpha_i) & cos(\alpha_i) & d_i(\beta_i) \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.2}$$

Donde i representa al sistema de referencia actual e $i-1$ al sistema de referencia anterior. La matriz de transformación homogénea total se obtiene como $H_0^n = H_0^1 H_1^2 ... H_{n-2}^{n-1} H_{n-1}^n$. La cinemática directa es la forma general de transformaciones homogéneas que concatena los sistemas de referencia cartesianos asociados a los eslabones del robot, todos relativos al sistema de referencia fijo Σ_0 [22].

La matriz de transformación homogénea H_{i-1}^i

La notación más común para representar la transformación de traslación y rotación en forma compacta se conoce como **transformación homogénea** [23]. Por ejemplo, para representar el caso de traslación y rotación del sistema $\Sigma_1(x_1, y_1, z_1)$ con respecto al sistema $\Sigma_0(x_0, y_0, z_0)$ tenemos:

$$\boldsymbol{p}_0 = \boldsymbol{d}_0^1 + R_0^1 \boldsymbol{p}_1 \tag{2.3}$$

La matriz de transformación homogénea seria la siguiente:

$$H_0^1 = \begin{bmatrix} R_0^1 & \boldsymbol{d}_0^1 \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.4}$$

donde $R_0^1 \in SO(3)$ y $\boldsymbol{d}_0^1 \in \mathbb{R}^3$. Para propósitos de acoplamiento en dimensiones, el vector $\mathbf{0}^t$ y el número 1 aparecen en el último renglón. La representación inversa que relaciona el punto $\boldsymbol{p}_1$ en función del punto $\boldsymbol{P}_0$ adquiere la siguiente forma:

$$\boldsymbol{p}_1 = -R_0^{1^T}\boldsymbol{d}_0^1 + R_0^{1^T}\boldsymbol{p}_0 \tag{2.5}$$

la transformación homogénea inversa está determinada por [2]:

$$H_0^{1^{-1}} = \begin{bmatrix} R_0^{1^T} & -R_0^{1^T}\boldsymbol{d}_0^1 \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.6}$$

Las matrices de transformación descritas anteriormente nos permiten modelar la orientación de la herramienta de trabajo colocada en el extremo final del robot, y junto con las transformaciones homogéneas dentro de una sola matriz incluye la orientación y posición de la herramienta de trabajo, esto a su ves proporciona la estructura del modelo cinemático del robot.

2.1.4. Modelo cinemático del UR5

Las ecuaciones de cinemática directa (posición y orientación) del extremo final del manipulador UR5 se obtienen mediante la metodología Denavit-Hartenberg, el desarrollo se encuentra basado en [17], [18] y [19] donde se escogen los marcos de referencia dados por la figura 2.5 para el robot manipulador.

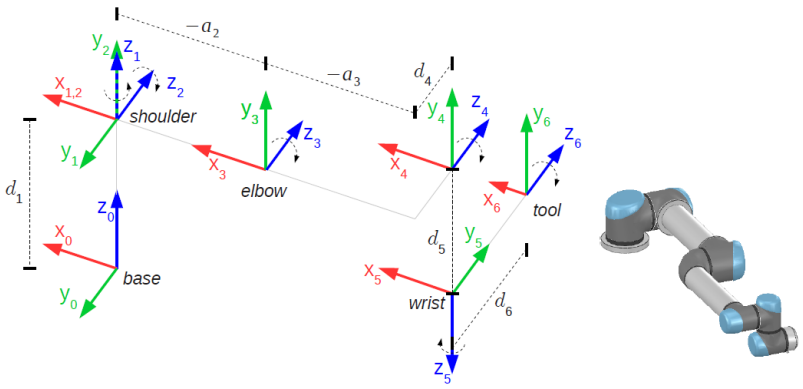

Figura 2.5: Marcos de referencia utilizados para describir al UR5 [18]

La tabla de parámetros Denavit-Hartenberg asociada a la descripción de los marcos de referencia de la figura 2.5 es la siguiente:
Procesando la tabla de parámetros Denavit-Hartenberg haciendo uso de la matriz de transformación homogénea descrita por la ecuación 2.2. Teniendo en cuenta que el manipulador es una cadena cinemática, se deben hallar las relaciones de marco actual Σ_i y marco anterior Σ_{i-1} a lo largo de la cadena cinemática hasta llegar al marco de referencia del extremo final Σ_n. La cinemática del robot manipulador es encontrar la matriz de transformación total H_n^0 que resulta de la concatenación de matrices de transformación homogénea H_{i-1}^i que hay entre los sistemas de referencia del robot:

$$H_{i-1}^i = H_{R_{z_{i-1}}}(\theta_i)H_{T_{z_{i-1}}}(d_i;\beta_i)H_{T_{x_{i-1}}}(l_i)H_{R_{x_{i-1}}}(\alpha_i) \tag{2.7}$$

Eslabón	α_i	a_i	d_i	θ_i
1	$\frac{\pi}{2}$	0	d_1	q_1
2	0	a_2	0	q_2
3	0	a_3	0	q_3
4	$\frac{\pi}{2}$	0	d_4	q_4
5	$-\frac{\pi}{2}$	0	d_5	q_5
6	0	0	d_6	q_6

Cuadro 2.1: Parámetros Denavit-Hartenberg del robot UR5

resolviendo para los sistemas de referencia del robot:

$$H_0^1 = \begin{bmatrix} cos(q_1) & 0 & sen(q_1) & 0 \\ sen(q_1) & 0 & -cos(q_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_1^2 = \begin{bmatrix} cos(q_2) & -sen(q_2) & 0 & a_2cos(q_2) \\ sin(q_2) & cos(q_2) & 0 & a_2sen(q_2) \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^3 = \begin{bmatrix} cos(q_3) & -sen(q_3) & 0 & a_3cos(q_3) \\ sen(q_3) & cos(q_3) & 0 & a_3sin(q_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_3^4 = \begin{bmatrix} cos(q_4) & 0 & sen(q_4) & 0 \\ sen(q_4) & 0 & -cos(q_4) & 0 \\ 0 & 1 & 0 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_4^5 = \begin{bmatrix} cos(q_5) & 0 & -sen(q_5) & 0 \\ sen(q_5) & 0 & cos(q_5) & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_5^6 = \begin{bmatrix} cos(q_6) & -sen(q_6) & 0 & 0 \\ sen(q_6) & cos(q_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de transformación total queda determinada por:

$$H_0^6 = H_0^1 H_1^2 H_2^3 H_3^4 H_4^5 H_5^6 \tag{2.8}$$

Debido al número de grados de libertad del robot (6), realizar el procedimiento manual para obtener el calculo de la matriz de transformación total resulta un poco

complicado, por este motivo se implemento en MATLAB un programa que realiza el calculo simbólico de esta matriz, la cual usando la siguiente notación:

$$H_0^6 = \begin{bmatrix} \hat{x}_{x0}^6 & \hat{y}_{x0}^6 & \hat{z}_{x0}^6 & p_{0x}^6 \\ \hat{x}_{y0}^6 & \hat{y}_{y0}^6 & \hat{z}_{y0}^6 & p_{0y}^6 \\ \hat{x}_{z0}^6 & \hat{y}_{z0}^6 & \hat{z}_{z0}^6 & p_{0z}^6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

donde $[\hat{x}_{x0}^6, \hat{x}_{y0}^6, \hat{x}_{z0}^6]^T$ corresponde a la orientación o proyección del eje x_6 sobre Σ_0, es decir sobre cada uno de los ejes del marco de referencia fijo $[x0, y0, z0]^T$, del mismo modo los vectores $[\hat{y}_{x0}^6, \hat{y}_{y0}^6, \hat{y}_{z0}^6]^T$ y $[\hat{z}_{x0}^6, \hat{z}_{y0}^6, \hat{z}_{z0}^6]^T$ corresponden a la orientación que tienen los marcos y_6 y z_6 con respecto a los marcos del origen. De manera particular el vector $[p_{0x}^6, p_{0y}^6, p_{0z}^6]^T$ corresponde a la posición cartesiana del extremo final del robot. Finalmente ay que tener en cuenta que el vector $\mathbf{0}^T$ y el número 1 se usan con fines de acoplamiento en el álgebra, no esta de más decir que estos elementos corresponden a la perspectiva y escala respectivamente, estos elementos tienen uso en otras áreas de la robótica. Los elementos de la matriz de transformación homogénea H_0^6 son los siguientes:

$$\hat{x}_{x0}^6 = c_6(s_1 s_5 + c_{234} c_1 c_5) - s_{234} c_1 s_6$$
$$\hat{x}_{y0}^6 = -c_6(c_1 s_5 + c_{234} c_5 s_1) - s_{234} s_1 s_6$$
$$\hat{x}_{z0}^6 = c_{234} s_6 + s_{234} c_5 c_6$$

$$\hat{y}_{x0}^6 = -s_6(s_1 s_5 + c_{234} c_1 c_5) - s_{234} c_1 c_6$$
$$\hat{y}_{y0}^6 = s_6(c_1 s_5 - c_{234} c_5 s_1) - s_{234} c_6 s_1$$
$$\hat{y}_{z0}^6 = c_{234} c_6 - s_{234} c_5 s_6$$

$$\hat{z}_{x0}^6 = c_5 s_1 - c_{234} c_1 s_5$$
$$\hat{z}_{y0}^6 = -c_1 c_5 - c_{234} s_1 s_5$$
$$\hat{z}_{z0}^6 = -s_{234} s_5$$

$$p_{0x}^6 = d_6(c_5 s_1 - c_{234} c_1 s_5) + d_4 s_1 + a_2 c_1 c_2 + d_5 s_{234} c_1 + a_3 c_1 c_2 c_3 - a_3 c_1 s_2 s_3$$
$$p_{0y}^6 = a_2 c_2 s_1 - d_4 c_1 - d_6(c_1 c_5 + c_{234} s_1 s_5) + d_5 s_{234} s_1 + a_3 c_2 c_3 s_1 - a_3 s_1 s_2 s_3$$
$$p_{0z}^6 = d_1 + a_3 s_{23} + a_2 s_2 - d_5 c_{234} - d_6 s_{234} s_5$$

donde $s_1 = sen(q_1), c_1 = cos(q_1)$ y $s_{234} = sen(q_2 + q_3 + q_4), c_{234} = cos(q_2 + q_3 + q_4)$. Esta notación se usa para los demás elementos c_i o s_i.

Modelo cinemático del UR5 opción 2

Debido a que el software con el cual se esta trabajando es MATLAB, se desarrolla una segunda opción de modelo cinemático directo tal que los ejes del marco inercial coincidan con los ejes de las gráficas del MATLAB, ya que como podemos visualizar en la figura 2.5 el eje z y y coinciden con los ejes del MATLAB sin embargo el eje x apunta en la dirección contraria. En resumen en este esquema no se cumple con la regla de la mano derecha. Teniendo esto en cuenta se propone otra tabla de parámetros Denavit - Hartenberg que se muestra en el cuadro considerando los ejes que se visualizan en las figuras 2.6 y 2.7.

Eslabón	α_i	a_i	d_i	θ_i
1	$\frac{\pi}{2}$	0	d_1	q_1
2	0	a_2	0	q_2
3	0	a_3	0	q_3
4	$\frac{\pi}{2}$	0	$-d_4$	q_4
5	$\frac{\pi}{2}$	0	d_5	q_5
6	0	0	d_6	$q_6 + \pi$

Cuadro 2.2: Parámetros Denavit-Hartenberg del robot UR5 - Opción 2

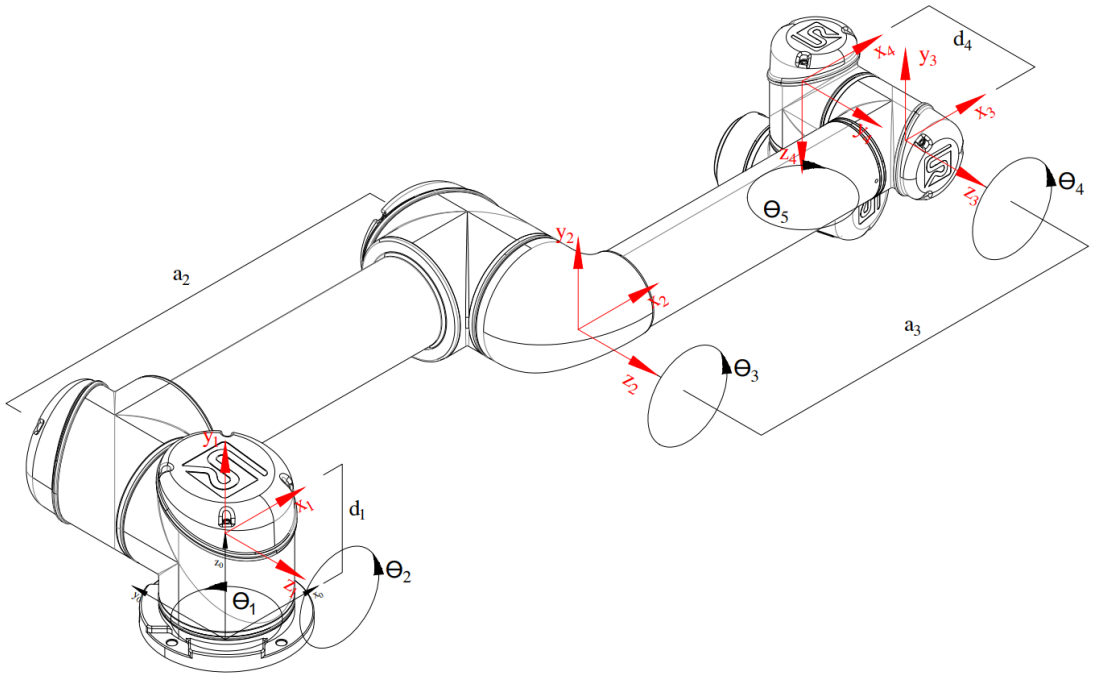

Figura 2.6: Sistemas de referencia en los eslabones 1, 2, 3 y 4

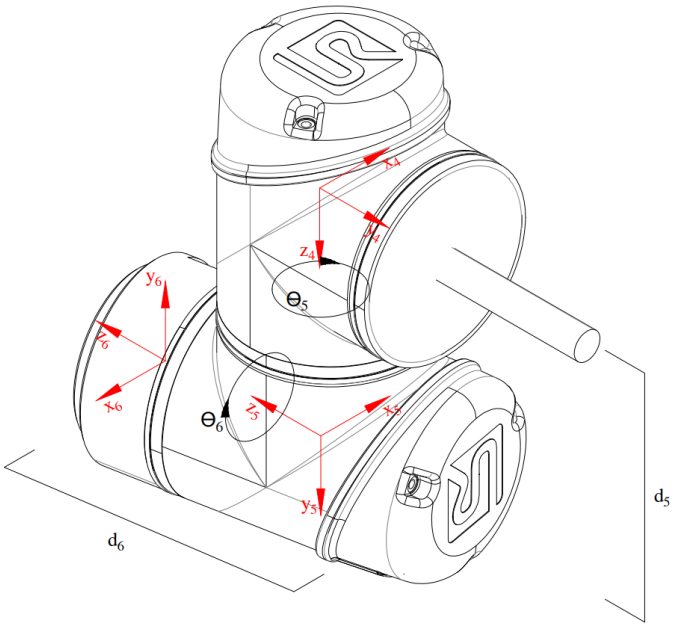

Figura 2.7: Sistemas de referencia para los eslabones 4, 5 y 6

2.2. Cinemática Inversa

Una vez obtenida la cinemática directa se tiene modelada la posición del extremo final del robot en coordenadas cartesianas $[x, y, z]^T$ y la orientación $[\psi, \theta, \phi]^T$, con respecto a un sistema de referencia fijo $\Sigma_0(x_0, y_0, z_0)$, así como los parámetros geométricos l_i, teniendo esto en cuenta surge la siguiente pregunta natural:

¿Pueden obtenerse las coordenadas articulares del robot $\boldsymbol{q}$ para que el extremo final de robot se posicione en las coordenadas cartesianas deseadas con la orientación solicitada?
En robótica la cuestión planteada se conoce como **cinemática inversa** [23] y representa un área de la robótica que tiene mas complejidad que el problema de cinemática directa. De hecho para un robot manipulador siempre es posible encontrar el modelo de cinemática directa, mientras que en la cinemática inversa pueden haber varias soluciones e inclusive no existir una solución analítica [2]; si este es el caso, entonces como posibles formas de solución pueden proponerse métodos numéricos, iterativos, geométricos , redes neuronales o algoritmos heurísticos.
La **cinemática inversa** es un problema no lineal que relaciona las coordenadas articulares $\boldsymbol{q} \in \mathbb{R}^3$ en función de las coordenadas cartesianas y la orientación de la herramienta del extremo final del robot manipulador:

$$\boldsymbol{q} = \boldsymbol{f}_R^{-1}(x, y, z, l_i, \theta, \phi, \psi) \tag{2.10}$$

donde $\boldsymbol{q} = \boldsymbol{f}_R^{-1}(x, y, z, l_i, \theta, \phi, \psi)$ representa una función inversa de la ecuación 2.1. Resuelto el problema de cinemática directa e inversa se tiene cubierto el modelo geométrico del robot y es posible trasladarse del espacio cartesiano que comúnmente entendemos los humanos al espacio articular que maneja el robot y viceversa.

2.2.1. Cinemática inversa del UR5

La cinemática del robot manipulador UR5 ha sido desarrollado en trabajos como [18] y [19] donde se utiliza un método geométrico y un método analítico-algebraico respectivamente. Ambos tienen ventajas, el método geométrico es de gran ayuda y se complementa cuando se cuenta con el robot físico o un análisis detallado de su geometría. Por otra parte el método analítico se encuentra basado directamente en la cinemática directa del robot reduciéndose su objetivo a buscar despejar las variables articulares de las expresiones obtenidas a partir de la cinemática directa. A continuación se presenta la cinemática inversa del manipulador UR5, tomando de cada método sus ventajas para facilitar la solución de cinemática inversa.

2.2.2. Articulación 1

: El primer paso para encontrar a q_1 es determinar la posición del marco 5 con respecto al marco fijo que se encuentra en la base, es decir hallar p_0^5. Geométricamente p_0^5 se puede encontrar si se realiza una traslación hacia atrás del marco 6 al marco 5 sobre el eje z_6. Entendiendo lo anterior podemos escribir la traslación p_0^5 como:

$$p_0^5 = p_0^6 - d_6 \cdot \hat{z}_0^6 \Leftrightarrow$$

$$p_0^5 = H_0^6 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} \tag{2.11}$$

Una vez localizada la traslación de p_0^5, se considera a la figura 2.8 donde se puede visualizar las relaciones necesarias para calcular a q_1. Mediante inspección visual se pude ver que la rotación q_1 debe ser igual a la diferencia entre las rotaciones del marco 0 a 5 y 1 a 5.

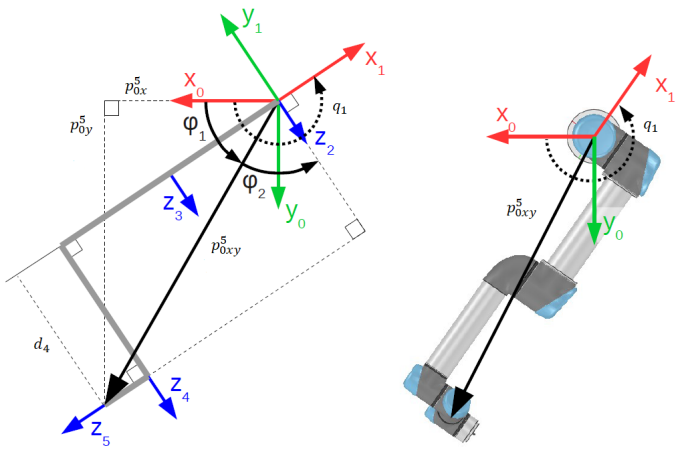

Figura 2.8: Visualización de la pose del robot para hallar q_1 [18]

entonces se deduce que:

$$q_1 = \phi_1 + \left(\phi_2 + \frac{\pi}{2}\right) \tag{2.12}$$

El ángulo q_1 puede ser encontrado observando al triangulo que tiene lados p_{0x}^5 y p_{0y}^5:

$$\phi_1 = atan2(p_{0y}^5, p_{0x}^5) \tag{2.13}$$

Para hallar el ángulo ϕ_2 se analiza al triangulo con ϕ_2 como uno de sus ángulos y con lados $|p_{0xy}^5|$ y d_4:

$$cos(\phi_2) = \frac{d_4}{|p_{0xy}^5|} \Rightarrow$$

$$\phi_2 = \pm acos\left(\frac{d_4}{|p^5_{0xy}|}\right)$$

$$\phi_2 = \pm acos\left(\frac{d_4}{\sqrt{(p^5_{0x})^2 + (p^5_{0y})^2}}\right) \tag{2.14}$$

Finalmente el ángulo q_1 queda determinado con la siguiente expresión:

$$q_1 = \phi_1 + \phi_2 + \frac{\pi}{2}$$

$$q_1 = atan2(p^5_{0y}, p^5_{0x}) \pm acos\left(\frac{d_4}{\sqrt{(p^5_{0x})^2 + (p^5_{0y})^2}}\right) \tag{2.15}$$

como se observa existen dos soluciones para q_1, dependiendo de si se va a trabajar con un hombro izquierdo o derecho. Por otra parte el uso de la función $atan2()$ es esencial para asegurar los signos y el comportamiento adecuados, sobre todo cuando $p^5_{0x} = 0$.

2.2.3. Articulación 5

Dada una configuración particular de q_1 se puede resolver el ángulo q_5. Usando la transformación del marco 1 al marco se forma la siguiente igualdad:

$$\begin{bmatrix} -sen(q_1) & cos(q_1) & 0 \end{bmatrix} \begin{bmatrix} p^6_{0x} \\ p^6_{0y} \\ p^6_{0z} \end{bmatrix} = -d_4 - cos(q_5)d_5$$

de donde podemos despejar a q_5:

$$q_5 = \pm acos\left(\frac{p^6_{0x}sen(q_1) - p^6_{0y}cos(q_1) - d_4}{d_6}\right) \tag{2.16}$$

nuevamente tenemos 2 soluciones para q_5, que corresponden a configuraciones donde la muñeca está "adentro-abajo" o "afuera-arriba".

2.2.4. Articulación 6

Para encontrar a la sexta articulación q_6 se considera el eje coordenado $\hat{y}^1_6$ y se establece la siguiente relación:

$$\begin{bmatrix} -\hat{x}^6_{0x}sen(q_1) + \hat{x}^6_{0y}cos(q_1) \\ -\hat{y}^6_{0x}sen(q_1) + \hat{y}^6_{0y}cos(q_1) \\ -\hat{z}^6_{0x}sen(q_1) + \hat{z}^6_{0y}cos(q_1) \end{bmatrix} \begin{bmatrix} -cos(q_6)cos(q_5) \\ sen(q_6)sen(q_5) \\ -cos(q_5) \end{bmatrix}$$

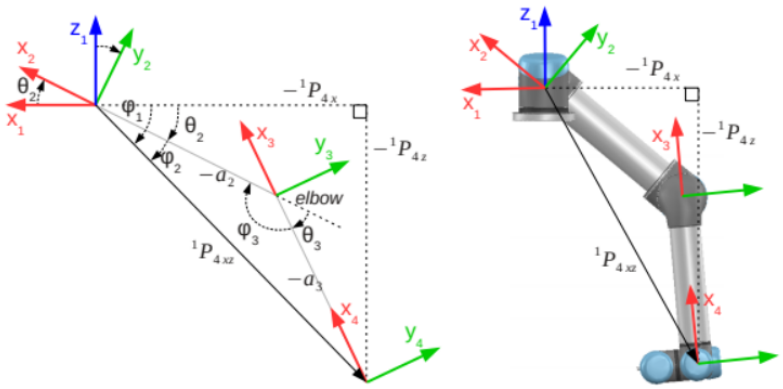

Figura 2.9: Configuración planar 3R formada por las articulaciones 2, 3 y 4. Con $\theta_1 = q_1$ y $\theta_3 = q_3$ (Skovgaard, 2018)

La igualdad presentada, representa una relación de coordenadas esféricas para establecer la dirección del vector unitario $\hat{y}_6^1$ en donde q_6 corresponde al ángulo acimut y el ángulo q_6 corresponde al ángulo polar. De lo anterior se deriva que:

$$q_6 = atan2\left(\frac{-\hat{x}_{6y}^0 \cdot sin(q_1) + \hat{y}_{6y}^0 \cdot cos(q_1)}{sen(q_5)}, \frac{\hat{x}_{6x}^0 \cdot sen(q_1) - \hat{y}_{6x}^0 \cdot cos(q_1)}{sen(q_5)}\right) \tag{2.17}$$

2.2.5. Articulación 3

Para hallar la rotación q_3 hay que tomar en cuenta a la articulación q_2 y q_4 y derivar que estas 3 articulaciones constituyen junto con los eslabones a_2 y a_3 la estructura de un robot planar 3R. Analíticamente esto se demuestra verificando la figura 2.9 donde se establecieron los sistemas de referencia para hallar la tabla D-H. En esta imagen podemos ver que los ejes z_2, z_3, z_4 que corresponden a las articulaciones 2, 3, 4 del robot serán siempre paralelos, de ahí que se considera esta parte del robot como una configuración planar 3R.
Para hallar a la articulación q_3 se analiza la relación del marco 4 con el marco 1 ya que al conocer las articulaciones anteriores, se tiene conocimiento de las matrices de transformación H_0^1, H_4^5 y H_5^6. Observando la figura 2.9 se ilustra que la longitud del vector de traslación $|p_{1xz}^4|$ se encuentra determinada únicamente por q_3 la cual se encuentra conociendo el ángulo ϕ_3 que también se muestra en la imagen y del cual se obtiene la siguiente relación:

$$cos(\phi_3) = \frac{(-a_2)^2 + (-a_3)^2 - |p_{1xz}^4|^2}{2(-a_2)(-a_3)} = \frac{a_2^2 + a_3^2 - |p_{1xz}^4|^2}{2a_2a_3} \tag{2.18}$$

Luego la relación entre $cos(\phi_3)$ y $cos(q_3)$ es:

$$cos(q_3) = cos(\pi - \phi_3) = -cos(\phi_3) \tag{2.19}$$

De la relación de las ecuaciones 2.18 y 2.19 se deriva:

$$cos(q_3) = -\frac{a_2^2 + a_3^2 - |p_{1xz}^4|^2}{2a_2a_3}$$

Finalmente:

$$q_3 = \pm acos\left(\frac{|p_{1xz}^4|^2 - a_2^2 - a_3^2}{2a_2a_3}\right) \tag{2.20}$$

Como se puede ver nuevamente se tienen 2 soluciones que corresponderían a "codo arriba" o "codo abajo". Además se debe tener en cuenta que la función $acos()$ solo tiene soluciones para un dominio $\in [-1, 1]$. Un detalle interesante es que se puede demostrar que esto es equivalente a decir que la longitud $|p_{1xz}^4|$ se encuentra acotada $\in [|a_2 - a_3|, |a_2 + a_3|]$.

2.2.6. Articulación 2

Para hallar esta articulación se inspecciona la figura 2.9, de donde se ve que $q_2 = \phi_1 - \phi_2$. Estos ángulos auxiliares a su vez pueden ser hallados de la siguiente manera:

$$\phi_1 = atan2(-p_{1z}^4, \ -p_{1x}^4) \tag{2.21}$$

$$\frac{sen(\phi_2)}{-a_3} = \frac{sen(\phi_3)}{|p_{1xz}^4|} \Leftrightarrow$$

$$\phi_2 = asin\left(\frac{-a_3 sen(\phi_3)}{|p_{1xz}^4|}\right) \tag{2.22}$$

Sustituyendo en 2.22 q_3 por ϕ_3, esto teniendo en cuenta que $sen(\phi_3) = sen(\pi - q_3) = sen(q_3)$, luego retomando:

$$q_2 = \phi_1 - \phi_2$$

Al ultimo:

$$q_2 = atan2(-p_{1z}^4, \ -p_{1x}^4) - asin\left(\frac{-a_3 sen(q_3)}{|p_{1xz}^4|}\right) \tag{2.23}$$

2.2.7. Articulación 4

El ángulo de la ultima articulación se puede obtener de la matriz de transformación H_3^4 teniendo en cuenta que q_4 corresponde al ángulo entre los ejes x_4 y x_3 medido alrededor del eje eje z_4, finalmente usando la columna de orientaciones del eje x_4 respecto a x_3, $\hat{x}_3^4$:

$$q_4 = atan2(\hat{x}_{3y}^4, \ \hat{x}_{3x}^4) \tag{2.24}$$

2.2.8. Comprobación Cinemática Directa e Inversa

Con el fin de verificar que la cinemática directa e inversa funcionara de manera adecuada una ves implementadas en MATLAB como funciones, se realizo un programa para verificar que dichas funciones fueran capaces de poder seguir una serie de puntos dados. Es decir se plantearon dos trayectorias un en el plano $x-y$ y otra en el espacio tridimensional mediante una función vectorial, posterior a eso se pasaron cada uno de los puntos en coordenadas cartesianas a la cinemática inversa para obtener los valores de posición articular necesarias para alcanzar el punto dado en el espacio. Finalmente se pasan todos los valores de posición articular a la cinemática directa para verificar que efectivamente todos los puntos que se proponen por las trayectorias son alcanzables por el robot. En la figura 2.10 se muestran la trayectoria 3D propuesta, luego en la figura 2.11 se muestran los valores que retorna la cinemática inversa, luego en las figuras 2.12 y 2.13 se muestran los valores que retorna la cinemática directa y la trayectoria propuesta y devuelta respectivamente. Como se puede observar las funciones trabajan adecuadamente. Cabe destacar que los valores que de estas trayectorias en el espacio articular se usaran mas adelante en el control del robot.

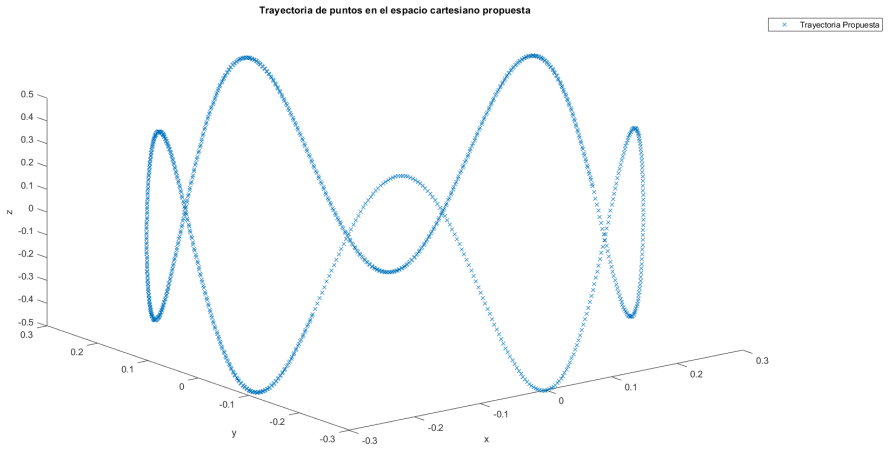

Figura 2.10: Trayectoria de puntos en el espacio propuesta

Por otra parte en las figuras 2.14 y 2.15 se muestra la trayectoria propuesta en el plano $x-y$ y las posiciones articulares que retorna la cinemática inversa respectivamente. Luego en las figura 2.16 podemos ver la trayectoria que se genera de introducir las variables articulares en la cinemática directa del robot y finalmente en la figura 2.17 se muestra la comparativa de ambas trayectorias. De igual forma que en la trayectoria 3D la cinemática directa e inversa funcionan adecuadamente.

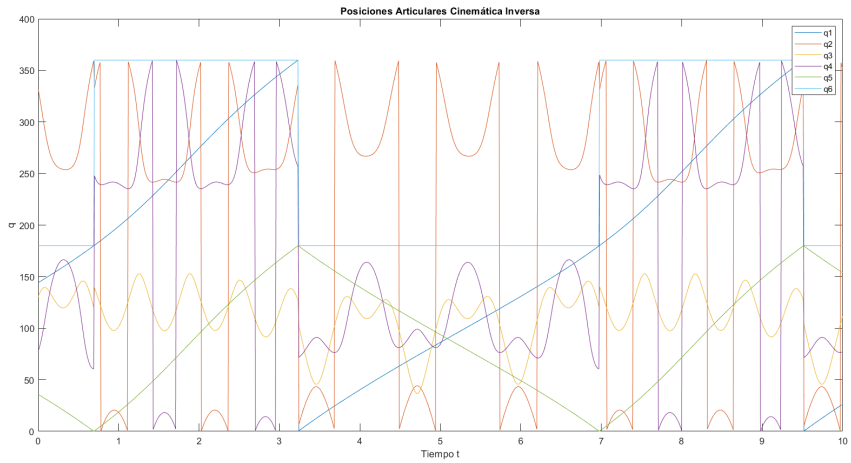

Figura 2.11: Posiciones articulares devueltas por la cinemática inversa

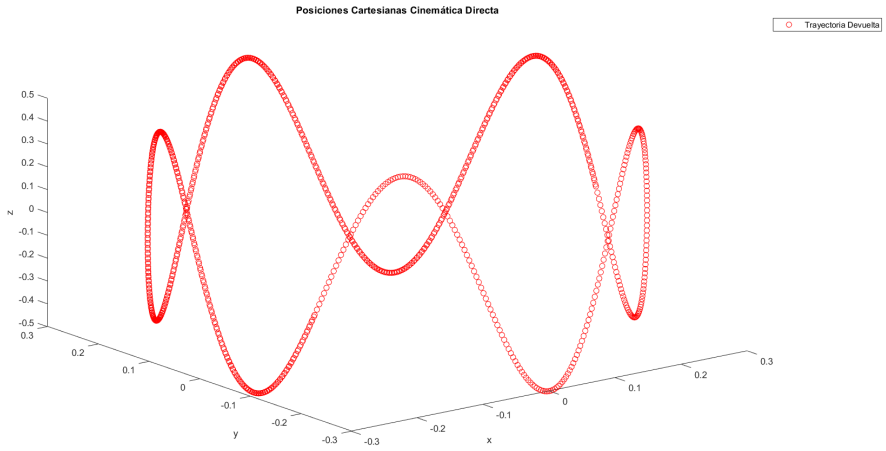

Figura 2.12: Trayectoria devuelta por la cinemática directa

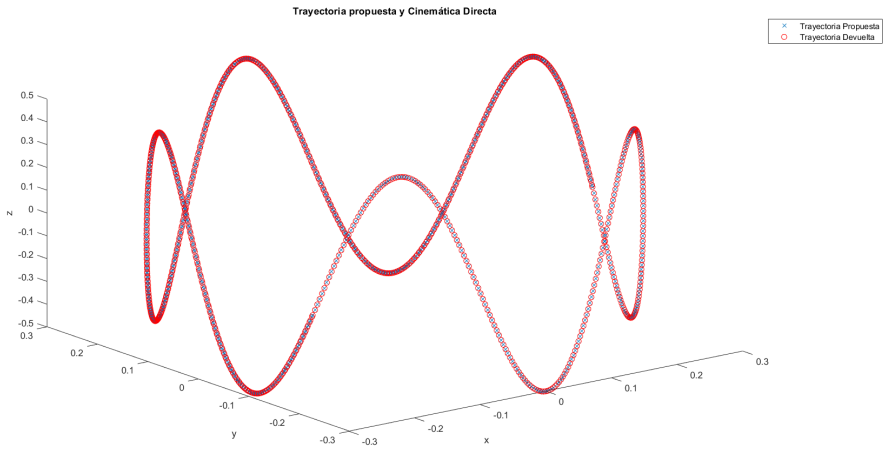

Figura 2.13: Trayectoria propuesta y trayectoria devuelta

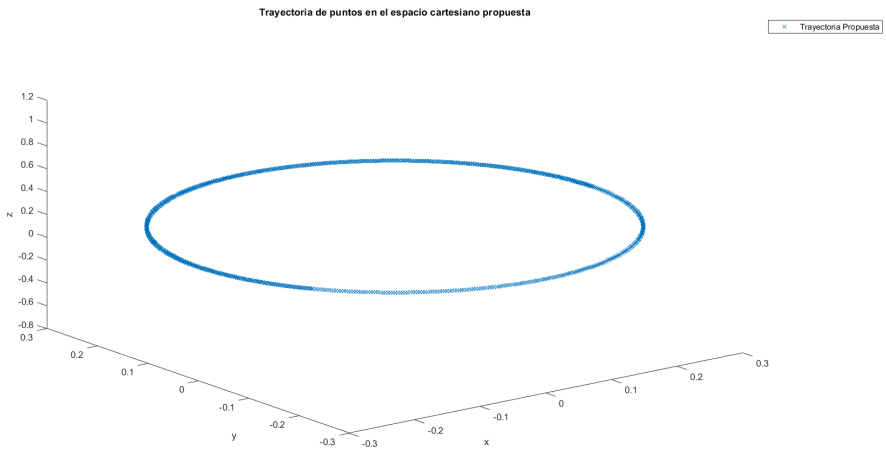

Figura 2.14: Trayectoria de puntos en el plano $x - y$ propuesta

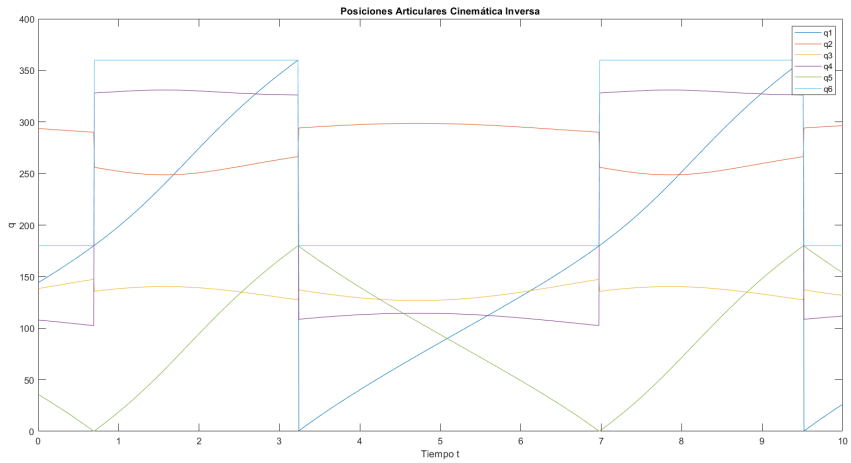

Figura 2.15: Posiciones articulares devueltas por la cinemática inversa plano $x - y$

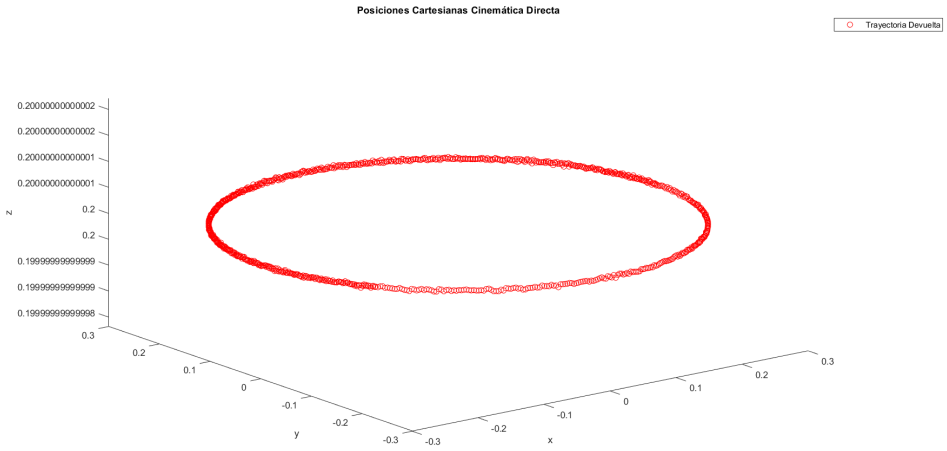

Figura 2.16: Trayectoria devuelta por la cinemática directa plano $x - y$

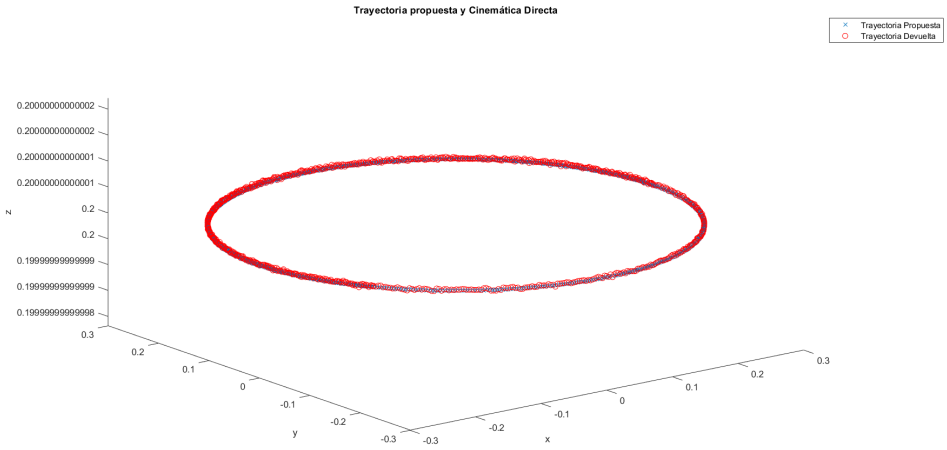

Figura 2.17: Trayectoria propuesta y trayectoria devuelta plano $x - y$

2.3. Jacobiano

Hasta este punto se ha realizado el análisis cinemático del robot manipulador de estudio, como ya se menciono estos análisis nos permiten obtener una relación o mapeo entre las variables de posición de las articulaciones y las variables de posición del efector final. Generalmente la cinemática directa e inversa se utilizan para conocer los valores que deben tener las variables articulares para llegar a un cierto punto en el espacio con el efector final. Sin embargo es claro ver que en la mayoría de tareas que realizan los robots manipuladores no solo se requiere alcanzar una posición final sino que se requiere llegar a esa posición con cierta velocidad en el efector final por lo cual los aspectos relacionados con las velocidades que se desarrollan en los manipuladores son de mucho interés.
Además una vez que se comienza con el control del robot manipulador es indispensable contar con la dinámica del mismo ya que en ellas intervienen no solo la posición sino también las velocidades, aceleraciones y torques. De aquí se deduce que se necesita un análisis en el que se estudien las velocidades del efector final y de cualquier punto del robot. En este punto es donde aparece el concepto de Jacobiano del manipulador [22]. En resumen el Jacobiano del manipulador es una herramienta que permite establecer una relación entre las velocidades de las articulaciones y las velocidades lineales y angulares del efector final [22]. Matemáticamente el Jacobiano es una matriz que puede verse como una transformación de espacios y es básicamente una versión vectorial de la derivada normal de una cierta función. Tiene una amplia variedad de aplicaciones entre las que se encuentran: planeación de trayectorias del manipulador, obtención de las ecuaciones dinámicas del robot manipulador empleando las ecuaciones de Euler-Lagrange, relación entre las fuerzas y pares del manipulador, etc.

2.3.1. Jacobiano del manipulador

Supóngase que se tiene una función vectorial de la forma $\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{x})$, donde $\boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{x} \in \mathbb{R}^m$. Usando la regla de la cadena se deriva y se obtiene lo siguiente:

$$\dot{\boldsymbol{y}} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} = J(\boldsymbol{x})\dot{\boldsymbol{x}} \tag{2.25}$$

donde la matriz $\frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}} \in \mathbb{R}^{n \times m}$ se denomina Jacobiano del manipulador.

2.3.2. Metodología para calcular el Jacobiano

Existen diferentes formas para calcular el Jacobiano de un robot manipulador, como métodos iterativos, e método de propagación de velocidades, entre otros, sin embargo estos métodos son largos y complejos, por otra parte existe un método expuesto en [] que nos permite calcular el jacobiano de un manipulador usando los datos obtenidos por la cinemática directa, dicho método se explica a continuación. En general se desea obtener una representación del Jacobiano que tenga la siguiente estructura:

$$\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} = J(\boldsymbol{q})\dot{\boldsymbol{q}} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{\boldsymbol{q}} \tag{2.26}$$

Ahora bien considerando al robot manipulador de n grados de libertad, donde se supone que la i-ésima articulación es rotacional y la j-ésima articulación es prismática. Por convención, los ejes de acción de las articulaciones rotacional y prismática se encuentran representados por los vectores unitarios $\boldsymbol{z}_i$ y $\boldsymbol{z}_j$ respectivamente. Por lo tanto se puede ver que:

$$\boldsymbol{\omega}_i = \dot{q}_i \boldsymbol{z}_i$$
$$\boldsymbol{v}_j = \dot{q}_j \boldsymbol{z}_j$$

Posteriormente lo que se intentara es buscar una expresión para calcular la velocidad lineal y angular del último eslabón del robot, por medio de la suma de las contribuciones en velocidad debidas a cada una de las articulaciones. Considere un vector $\boldsymbol{p}_{in}$ el cual representa la posición del sistema de referencia {n}, desde el origen del sistema de referencia {i}. Así pues la articulación rotacional i contribuye con $\boldsymbol{\omega}_i$ a la velocidad angular de {n} y por el teorema de Euler, contribuye con $\boldsymbol{\omega}_i \times \boldsymbol{p}_{in}$ a la velocidad lineal de {n}. Por otra parte la articulación prismática j contribuye con $\boldsymbol{v}_j$ a la velocidad lineal de {n}, pero no tiene contribución en cuanto a su velocidad angular puesto que este tipo de articulación no desarrolla velocidad angular.
Para un robot de n grados de libertad, la velocidad lineal total $\boldsymbol{v}$ del sistema de referencia $\{n\}$ se obtiene sumando la contribución de cada una de las n articulaciones, lo cual se puede resumir en la siguiente ecuación:

$$\boldsymbol{v} = \sum_{i=1}^{n} [\sigma_i \boldsymbol{v}_i + \bar{\sigma}_i [\omega_i \times \boldsymbol{p}_{in}]] \tag{2.27}$$

Por optra parte la velocidad angular total de $\{n\}$ se calcula de la siguiente manera:

$$\boldsymbol{\omega} = \sum_{i=1}^{n} \bar{\sigma}_i \boldsymbol{\omega}_i \tag{2.28}$$

donde $\sigma_i = 1$ si la articulación es prismática y $\sigma = 0$ si la articulación es rotacional, de aqui se puede ver que $\bar{\sigma}$ representa al complemento de σ_i, por lo que $\bar{\sigma}_i = 1$ si $\sigma_i = 0$. Como el eje de acción de la i-ésima articulación es $\boldsymbol{z}_i$ entonces $\boldsymbol{v}_i = \boldsymbol{z}_i \dot{q}_i$ y $\boldsymbol{omega}_i = \boldsymbol{z}_i \dot{q}_i$ donde $\dot{q}_i$ es la magnitud de la velocidad correspondiente en la i-ésima articulación. Teniendo en cuenta esta notación se vuelven a reescribir los vectores de velocidad lineal y angular del sistema de referencia $\{n\}$ como a continuación:

$$\boldsymbol{v} = \sum_{i=1}^{n} [\sigma_i \boldsymbol{z}_i \dot{q}_i + \bar{\sigma}_i [\boldsymbol{z}_i \dot{q}_i \times \boldsymbol{p}_{in}]] \tag{2.29}$$

$$\boldsymbol{\omega} = \sum_{i=1}^{n} \bar{\sigma}_i \boldsymbol{z}_i \dot{q}_i \tag{2.30}$$

Si nos damos cuenta en las expresiones, se puede factorizar el termino de velocidad articular $\dot{q}$ para obtener el siguiente par de expresiones:

$$\boldsymbol{v} = \sum_{i=1}^{n} [\sigma_i \boldsymbol{z}_i + \bar{\sigma}_i [\boldsymbol{z}_i \times \boldsymbol{p}_{in}]] \dot{q}_i \tag{2.31}$$

$$\boldsymbol{w} = \sum_{i=1}^{n} [\bar{\sigma}_i \boldsymbol{z}_i] \dot{q}_i \tag{2.32}$$

Desarrollando los términos de estas expresiones obtenemos lo siguiente:

$$\boldsymbol{v} = [\sigma_1 \boldsymbol{z}_1 + \bar{\sigma}[\boldsymbol{z}_1 \times \boldsymbol{p}_{1n}] \cdots \sigma_n \boldsymbol{z}_n + \bar{\sigma}_n [\boldsymbol{z}_n \times \boldsymbol{p}_{nn}]] \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J_v \dot{\boldsymbol{q}} \tag{2.33}$$

$$\boldsymbol{w} = [\bar{\sigma} \boldsymbol{z}_1 \cdots \bar{\sigma} \boldsymbol{z}_n] \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J_w \dot{\boldsymbol{q}} \tag{2.34}$$

Juntando las expresiones 2.33 y 2.34 se obtiene la expresión para calcular el Jacobiano:

$$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix} = \begin{bmatrix} [\sigma_1 \boldsymbol{z}_1 + \bar{\sigma}[\boldsymbol{z}_1 \times \boldsymbol{p}_{1n}] \cdots \sigma_n \boldsymbol{z}_n + \bar{\sigma}_n [\boldsymbol{z}_n \times \boldsymbol{p}_{nn}]] \\ [\bar{\sigma} \boldsymbol{z}_1 \cdots \bar{\sigma} \boldsymbol{z}_n] \end{bmatrix} \tag{2.35}$$

Ahora bien. la expresión anterior se puede trabajar mas si utilizamos la información del análisis de cinemática directa del manipulador. Para ello consideremos $\boldsymbol{p}_x = [x, y, z]^T$ el vector de posición del origen del sistema de referencia {n} con respecto a la base del robot. Teniendo esto en cuenta podemos calcular la velocidad lineal del sistema de referencia {n} derivando respecto al tiempo el vector $\boldsymbol{p}_x$ como a continuación:

$$\boldsymbol{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \dot{\boldsymbol{p}}_x = \frac{\partial \boldsymbol{p}_x}{\partial q_1} \dot{q}_1 + \cdots + \frac{\partial \boldsymbol{p}_x}{\partial q_n} \dot{q}_n = \begin{bmatrix} \frac{\partial \boldsymbol{p}_x}{\partial q_1} \cdots \frac{\partial \boldsymbol{p}_x}{\partial q_n} \end{bmatrix} \dot{\boldsymbol{q}} \tag{2.36}$$

Sustituyendo la expresión 2.36 en la expresión para el calculo del Jacobiano obtenida previamente se tiene que:

$$J(\boldsymbol{q}) = \begin{bmatrix} J_v(\boldsymbol{q}) \\ J_w(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{p}_x}{\partial q_1} \cdots \frac{\partial \boldsymbol{p}_x}{\partial q_n} \\ \bar{\sigma}_1 \boldsymbol{z}_1 \cdots \bar{\sigma}_n \boldsymbol{z}_n \end{bmatrix} \tag{2.37}$$

Como es de darse cuenta con esta ultima expresión podemos calcular el Jacobiano del manipulador usando únicamente los datos obtenidos por el análisis cinemático, es decir solo se requiere conocer el vector de posición $\boldsymbol{p}_x$ y la información de los vectores $\boldsymbol{z}_i$.

Por ultimo para expresar el Jacobiano en el sistema de referencia {0}, deben usarse las siguientes expresiones:

$$\boldsymbol{z}_i^0 = R_i^0 \boldsymbol{z}_i^i$$

$$\boldsymbol{z}_i^i = \boldsymbol{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Con esto ultimo, el Jacobiano expresado con respecto al sistema de referencia $\{0\}$ lo podemos calcular como a continuación:

$$J_0^n(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial \boldsymbol{p}_x^0}{\partial q_1} \cdots \frac{\partial \boldsymbol{p}_x^0}{\partial q_n} \\ \bar{\sigma}_1 R_1^0 \boldsymbol{z} \cdots \bar{\sigma}_n R_n^0 \boldsymbol{z} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{p}_x^0}{\partial q_1} \cdots \frac{\partial \boldsymbol{p}_x^0}{\partial q_n} \\ \bar{\sigma}_1 \boldsymbol{z}_1^0 \cdots \bar{\sigma} \boldsymbol{z}_n^0 \end{bmatrix} \tag{2.38}$$

2.3.3. Jacobiano del UR5

Una vez descrito el método para calcular el Jacobiano se procedió a realizar el calculo para el robot UR5, teniendo en cuenta los requerimientos se retoma la cinemática directa del manipulador dada por:

$$H_0^6 = \begin{bmatrix} \hat{x}_{x0}^6 & \hat{y}_{x0}^6 & \hat{z}_{x0}^6 & p_{0x}^6 \\ \hat{x}_{y0}^6 & \hat{y}_{y0}^6 & \hat{z}_{y0}^6 & p_{0y}^6 \\ \hat{x}_{z0}^6 & \hat{y}_{z0}^6 & \hat{z}_{z0}^6 & p_{0z}^6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.39}$$

Utilizando únicamente la posición del efector final que nos da la matriz de transformación homogénea H_0^6 podemos hallar la parte lineal del Jacobiano del robot UR5 de la siguiente manera:

$$J_v(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial p_{0x}^6}{\partial q_1} & \frac{\partial p_{0x}^6}{\partial q_2} & \frac{\partial p_{0x}^6}{\partial q_3} & \frac{\partial p_{0x}^6}{\partial q_4} & \frac{\partial p_{0x}^6}{\partial q_5} & \frac{\partial p_{0x}^6}{\partial q_6} \\ \frac{\partial p_{0y}^6}{\partial q_1} & \frac{\partial p_{0y}^6}{\partial q_2} & \frac{\partial p_{0y}^6}{\partial q_3} & \frac{\partial p_{0y}^6}{\partial q_4} & \frac{\partial p_{0y}^6}{\partial q_5} & \frac{\partial p_{0y}^6}{\partial q_6} \\ \frac{\partial p_{0z}^6}{\partial q_1} & \frac{\partial p_{0z}^6}{\partial q_2} & \frac{\partial p_{0z}^6}{\partial q_3} & \frac{\partial p_{0z}^6}{\partial q_4} & \frac{\partial p_{0z}^6}{\partial q_5} & \frac{\partial p_{0z}^6}{\partial q_6} \end{bmatrix} \tag{2.40}$$

Para realizar los cálculos que se requieren en la parte del Jacobiano lineal se realizo un programa en MATLAB que nos ayudara a realizar las derivadas parciales de la cinemática directa con respecto a cada una de las articulaciones, sin embargo el resultado del Jacobiano lineal no se expresa con totalidad aquí, debido a que sus expresiones son demasiado extensas.
Retomando la expresión 2.37 la parte angular del robot UR5 se puede calcular de la siguiente manera:

$$J_w = \begin{bmatrix} \bar{\sigma} \boldsymbol{z}_1^0 & \bar{\sigma} \boldsymbol{z}_2^0 & \bar{\sigma} \boldsymbol{z}_3^0 & \bar{\sigma} \boldsymbol{z}_4^0 & \bar{\sigma} \boldsymbol{z}_5^0 & \bar{\sigma} \boldsymbol{z}_6^0 \end{bmatrix} \tag{2.41}$$

Considerando que todas las articulaciones son rotacionales entonces $\sigma = 0$ por lo que $\bar{\sigma} = 1$. Por otra parte los vectores z_i^0 los podemos obtener de las matrices de transformación homogénea de las cuales se calculo la cinemática directa, utilizando

únicamente la parte que corresponde a las matrices de rotación R_i^0, por tanto el Jacobiano angular queda descrito de la siguiente manera:

$$J_w = \begin{bmatrix} R_1^0\boldsymbol{z} & R_2^0\boldsymbol{z} & R_3^0\boldsymbol{z} & R_4^0\boldsymbol{z} & R_5^0\boldsymbol{z} & R_6^0\boldsymbol{z} \end{bmatrix} \tag{2.42}$$

donde $\boldsymbol{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ como ya se analizó anteriormente. Así de esta forma se puede calcular cada elemento del Jacobiano angular del robot. A manera de ejemplo y puesto que para esta componente el calculo del Jacobiano es una expresión corta, se presenta el calculo para z_1^0:

$$\boldsymbol{z}_1^0 = \begin{bmatrix} cos(q_1) & 0 & sin(q_1) \\ sin(q_1) & 0 & -cos(q_1) \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} sin(q_1) \\ -cos(q_1) \\ 0 \end{bmatrix} \tag{2.43}$$

El Jacobiano angular resulta idéntico para los vectores z_2^0 y z_3^0 sin embargo para las componentes restantes las expresiones se hacen demasiado extensas, es también por ello que solo se muestran las expresiones para obtenerlas.

2.3.4. Jacobiano para un punto arbitrario en un eslabón

Para obtener la dinámica del manipulador es necesario no solo obtener el Jacobiano en el efector final, sino que se requiere el Jacobiano para puntos arbitrarios en un eslabón y más específicamente el Jacobiano en los centros de masa de cada eslabón. Entonces dado un vector $\boldsymbol{r}_{Pj}^0$ que apunta a un punto P en el eslabón l_j expresado en el marco de referencia inercial $\{0\}$, los elementos de la parte lineal de Jacobiano $J_{Pj}(\boldsymbol{q})$ son:

$$J_{Pj,v} = \frac{\partial \boldsymbol{r}_{Pj}^0}{\partial q_i} = z_{i-1}^0 \times (\boldsymbol{r}_{Pj}^0 - \boldsymbol{o}_{i-1}^0), \qquad \forall \quad i \leq j \tag{2.44}$$

$$J_{Pj,v} = 0 \qquad \forall \quad i > j \tag{2.45}$$

para $i \in \{1, 2, \cdots, n\}$. Las columnas de $J_{Pj,v}$ para $i > j$ son cero ya que la velocidad de el j^{th} eslabón no se ve afectada por el movimiento de los eslabones que vienen después. Esta característica también se da en el Jacobiano angular:

$$J_{Pj,w} = \boldsymbol{z}_{i-1}^0, \qquad \forall i \leq j \tag{2.46}$$

$$J_{Pj,w} = 0, \qquad \forall i > j \tag{2.47}$$

El Jacobiano entonces queda expresado como:

$$J_{Pj}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{z}_0^0 \times (\boldsymbol{r}_{Pj}^0 - \boldsymbol{o}_0^0) & \cdots & \boldsymbol{z}_{j-1}^0 \times (\boldsymbol{r}_{Pj}^0 - \boldsymbol{o}_{j-1}^0 & \mathbf{0} & \cdots & \mathbf{0}) \\ \boldsymbol{z}_0^0 & \cdots & \boldsymbol{z}_{j-1}^0 & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \tag{2.48}$$

2.4. Resolved Motion Rate Control

Existe una técnica para controlar al robot manipulador utilizando únicamente el análisis cinemático y el Jacobiano. Lo anterior se deduce de que dado un conjunto de desplazamientos infinitesimales de las articulaciones $\delta \boldsymbol{q}$ y de las coordenadas de operación $\delta \boldsymbol{x}$, el Jacobiano del manipulador permite establecer la siguiente relación:

$$\delta \boldsymbol{x} = J(\boldsymbol{q}) \delta \boldsymbol{q} \tag{2.49}$$

Teniendo en cuenta la relación 2.49, existe un control propuesto por Whitney[] denominado *Resolved Motion Rate Control* con siglas *RMRC*

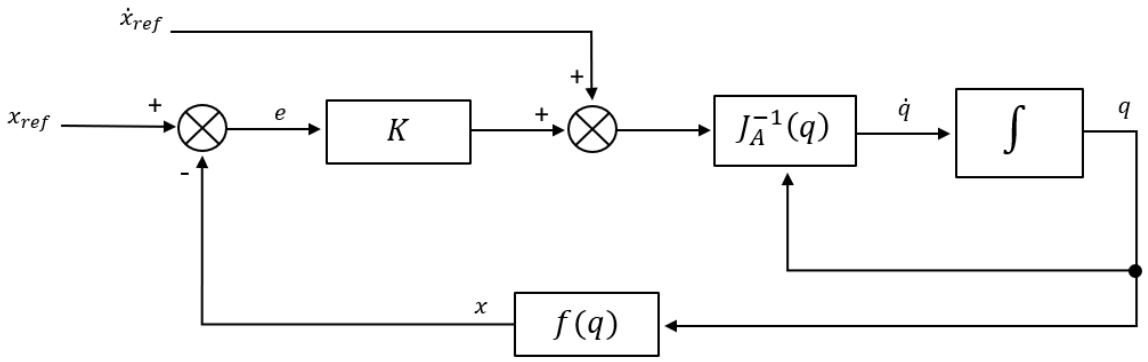

Figura 2.18: Diagrama a bloques de un control RMRC

El objetivo del control cinemático es proporcionar una ley de control con realimentación del estado $\dot{\boldsymbol{q}} = \pi(\boldsymbol{q})$, tal que el efector final siga una trayectoria deseada. Dado el diagrama de bloques expresado por la figura 2.18 se procede a definir la dinámica del error como:

$$\boldsymbol{e} = \boldsymbol{x}_{ref} - \boldsymbol{x} \tag{2.50}$$

donde:

$$\boldsymbol{x}_{ref} = \begin{bmatrix} \boldsymbol{\rho}_{ref} \\ \boldsymbol{\phi}_{ref} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix} \tag{2.51}$$

Las coordenadas operacionales del robot $\boldsymbol{x}$ se obtienen a través de la cinemática directa del manipulador:

$$\boldsymbol{x} = f(\boldsymbol{q}) \tag{2.52}$$

Por otra parte la cinemática diferencial se puede obtener haciendo uso de la matriz Jacobiana del manipulador de la siguiente manera:

$$\dot{\boldsymbol{x}} = J_A(\boldsymbol{q}) \cdot \dot{\boldsymbol{q}} \tag{2.53}$$

Derivando la expresión del error se tiene que:

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{x}} \tag{2.54}$$

.
Ahora bien, también sabemos que podemos expresar las coordenadas operacionales del manipulador con ayuda de la matriz Jacobiano de la siguiente forma:

$$\boldsymbol{x} = J_A(\boldsymbol{q}) \cdot \dot{\boldsymbol{q}} \tag{2.55}$$

Proponiendo una matriz de ganancias K se puede hacer que la derivada del error tienda a cero como a continuación:

$$\dot{\boldsymbol{e}} + K\boldsymbol{e} = 0 \quad \Rightarrow \quad \boldsymbol{e} \to 0 \tag{2.56}$$

De esta manera se puede tener la siguiente ley de control:

$$\dot{\boldsymbol{x}}_{ref} - J_A(\boldsymbol{q}) \cdot \dot{\boldsymbol{q}} + K_p(\boldsymbol{x}_{ref} - \boldsymbol{x}) = 0 \tag{2.57}$$

Despejando las velocidades de las articulaciones podemos encontrar la ecuación en bucle cerrado:

$$\dot{\boldsymbol{q}} = \pi(\boldsymbol{q}) = J_A^{-1}(\boldsymbol{q})(\dot{\boldsymbol{x}}_{ref} + K(\boldsymbol{x}_{ref} - f(\boldsymbol{q}))) \tag{2.58}$$

Para poder utilizar esta expresión se debe garantizar que exista la inversa de la matriz Jacobiana y por tanto el robot no se encuentre en una singularidad.

2.4.1. Implementación en el UR5

Para la implementación del control RRMC en el robot UR5, como se explica en la ecuación en lazo cerrado se necesito únicamente de la cinemática directa del robot y del Jacobiano Analítico como elementos matemáticos el robot. Además se requirió establecer las referencias para posición y orientación en el espacio cartesiano y sus derivadas. Posterior a esto los resultados que se obtuvieron se presentan a continuación. De la misma manera que en el control dinámico la trayectoria a realizar es una rosa polar de 5 pétalos, la cual se encuentra definida en el espacio cartesiano, puesto que la velocidad articular es la salida de la ecuación en lazo cerrado para este control, son las posiciones articulares lo que se obtiene al momento de solucionar dicha ecuación, los valores de posición articular se muestran en la gráfica 2.19. Posteriormente luego de realizar un registro con la función de la ecuación en lazo cerrado, es decir proporcionándole el vector de tiempo y los estados que corresponden a las posiciones articulares se puede obtener la gráfica de las velocidades articulares, esta se muestra en la figura 2.20. Posteriormente otra de las gráficas interesantes para mostrar son las de el error de posición, para lo cual tenemos dos gráficas: la primera en la figura 2.21 se muestran los errores de posición en el espacio cartesiano, como se puede apreciar estos tienden a cero conforme el tiempo tiende a infinito lo cual es adecuado, la segunda gráfica en la figura 2.22 corresponde a los errores de orientación en el efector final

los cuales como se aprecia también tienden a cero conforme el tiempo tiende a infinito. Estas dos gráficas demuestran que en el control RRMC esta realizando la trayectoria propuesta adecuadamente, sin embargo ya que este control como se menciono trabaja en el espacio cartesiano en la figura 2.23 se puede visualizar la trayectoria propuesta y la trayectoria realizada por este control en el espacio cartesiano, en esta gráfica se puede comprobar que la trayectoria propuesta se realiza de una manera adecuada. Además con el fin de tener otra perspectiva de la trayectoria realizada en el espacio cartesiano se puede visualizar en la figura otro ángulo de la imagen para ver la trayectoria realizada, en ella se puede apreciar que se parte de un punto el cual esta dado por las condiciones iniciales que se han planteado para todos los controladores expuestos y se contrasta que el controlador sigue adecuadamente la trayectoria que se propone. Finalmente en la figura 2.25 se puede visualizar el modelo 3D del robot en el espacio de tareas y también es posible distinguir la trayectoria que esta realizando, esto con la finalidad de mostrar una idea gráfica de que el robot efectivamente esta realizando la tarea que se le pide. Otra perspectiva del modelo 3D del robot físico se puede ver en la figura 2.26 donde se tiene una vista normal al plano xy y se distingue mejor la trayectoria que se realiza.

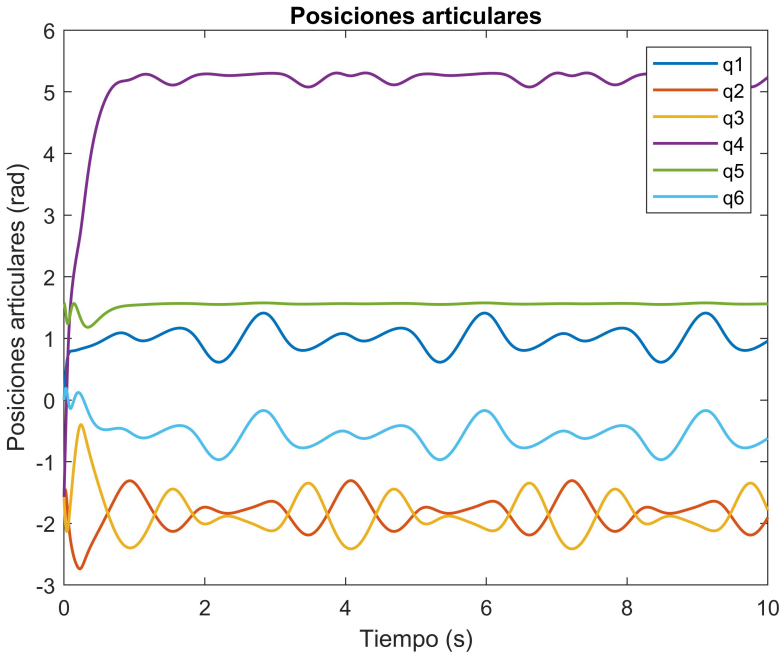

Figura 2.19: Posiciones articulares en el control RRMC

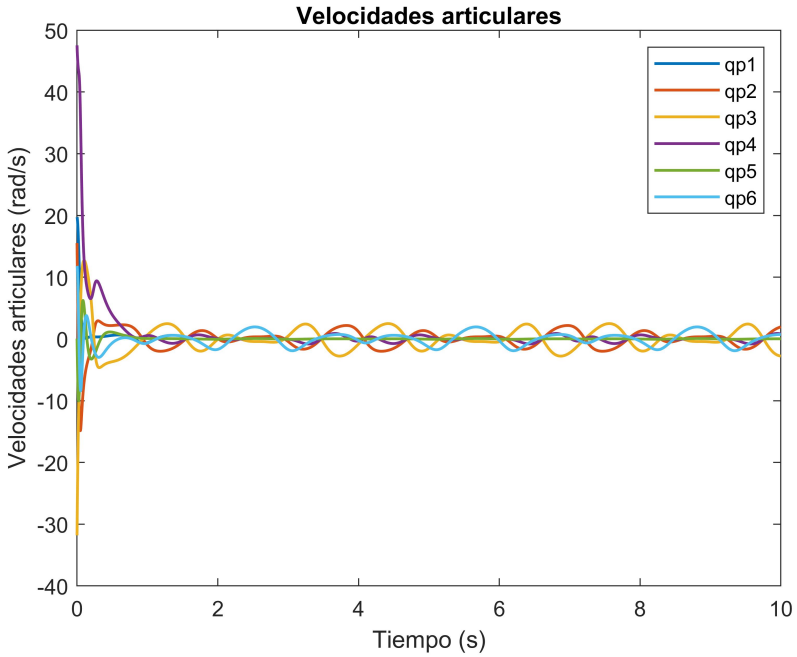

Figura 2.20: Velocidades articulares en el control RRMC

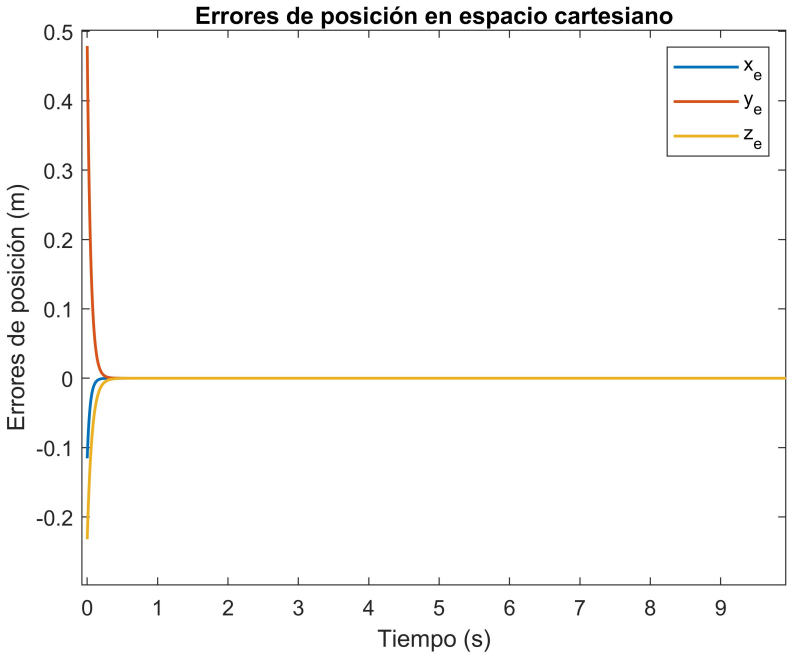

Figura 2.21: Errores de posición cartesiana en el control RRMC

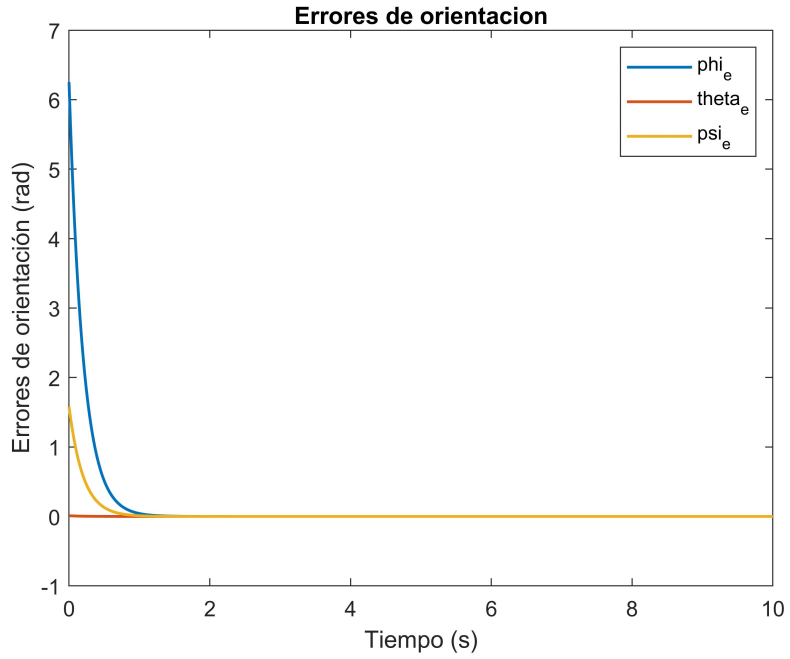

Figura 2.22: Errores de orientación en el control RRMC

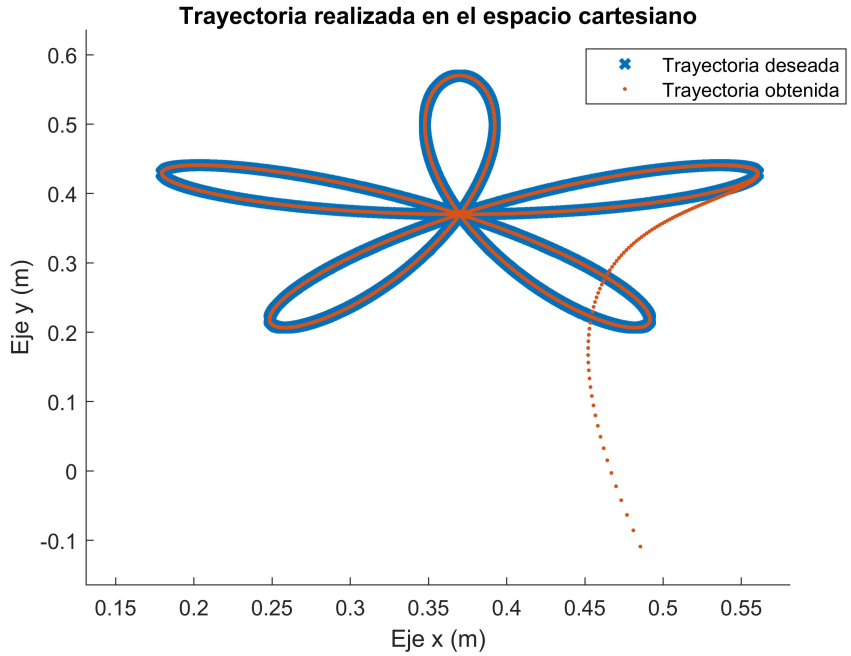

Figura 2.23: Trayectoria realizada en el espacio cartesiano con el control RRMC

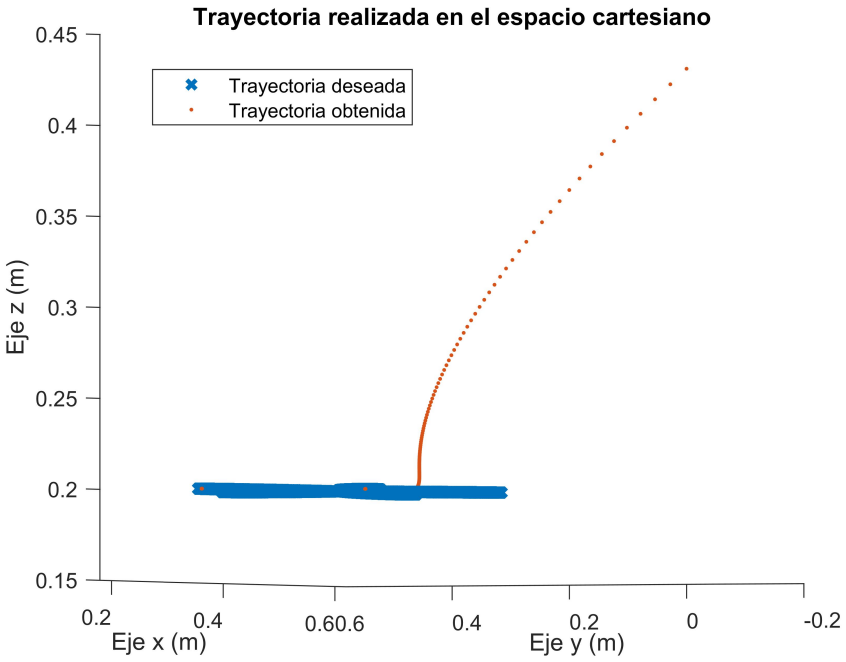

Figura 2.24: Trayectoria realizada en el espacio cartesiano con el control RRMC

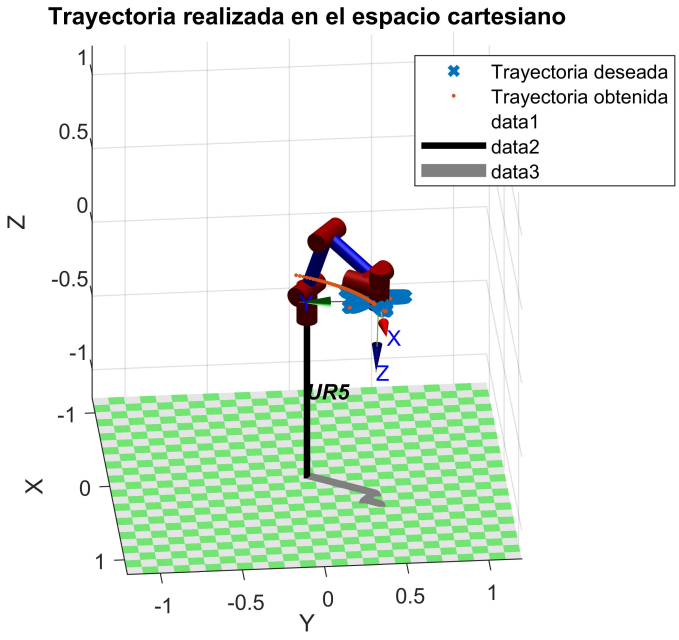

Figura 2.25: Visualización 3D del robot en el seguimiento de la trayectoria con el control RRMC

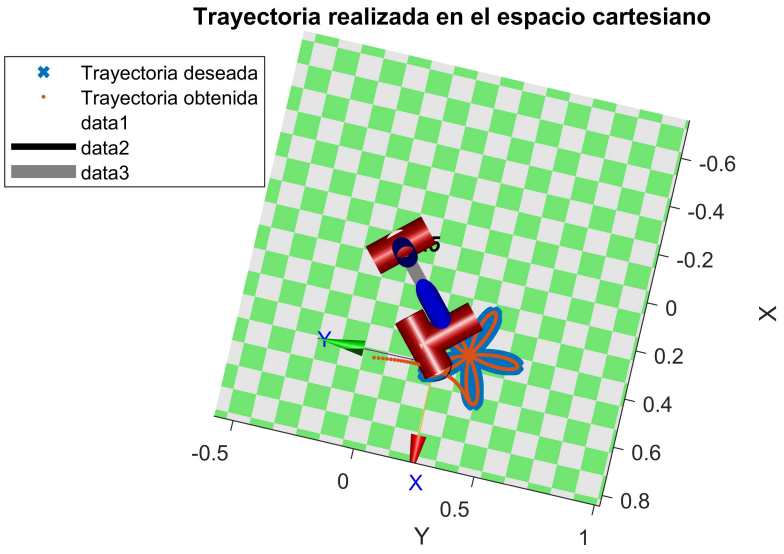

Figura 2.26: Visualización 3D del robot en el seguimiento de la trayectoria con el control RRMC (Vista Normal)

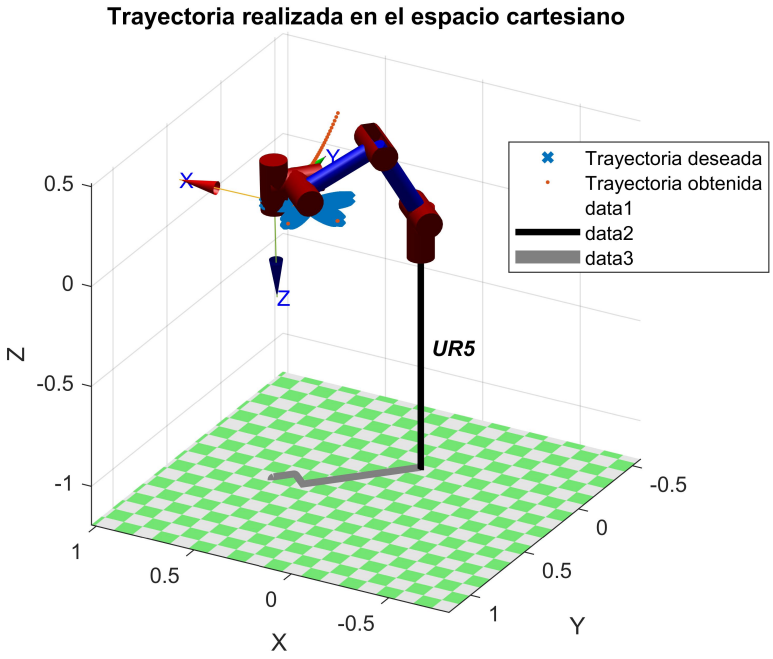

Figura 2.27: Visualización 3D del robot en el seguimiento de la trayectoria con el control RRMC (Vista Isométrica)

2.5. Dinámica

La dinámica es una disciplina que se encarga de estudiar el movimiento de un cuerpo en relación con las causas (fuerzas) que lo producen [23]. El estudio de la dinámica nos permite conocer y entender los fenómenos físicos que rigen a una estructura rígida compleja como la de un robot manipulador. El modelo dinámico es una estructura matemática descrita en ecuaciones diferenciales que utiliza variables de estado físicas como lo son: posición, velocidad y aceleración. Se puede decir que con el conocimiento de los valores numéricos de los parámetros del robot (masas, momentos de inercia, centros de masa y coeficientes de fricción) se puede reproducir de una manera "**fiel**" todos los fenómenos físicos del sistema. El modelo dinámico nos permite analizar en forma muy completa y detallada el comportamiento no lineal del manipulador [2]. Existen básicamente 3 enfoques que nos permiten estudiar la mecánica de los cuerpos en el espacio-tiempo:

- El enfoque Newtoniano, también conocido como ***Mecánica Clásica o Mecánica Vectorial*** debido a que la fuerza, velocidad y aceleración representan cantidades vectoriales en el análisis [2].
- El modelo dinámico hamiltoniano es una opción no muy popular pero que resulta interesante ya que es expresado como la suma de la energía cinética y energía potencial, estructura denominada ´´hamiltoniano". Este enfoque toma en cuenta el momentum (cantidad de movimiento), velocidad y posición articular [22].
- La ***Mecánica Lagrangiana*** también conocida como ***Dinámica Analítica*** se expresa directamente en términos del "lagrangiano" que se define como la diferencia entre la energía cinética y potencial. Esta metodología es la que proporciona las mejores ventajas en cuanto a propiedades matemática del modelo dinámico. Es gracias a esta metodología que la robótica ha tenido un gran desarrollo en los últimos años [2] y [22].

Por otra parte el modelo dinámico de robots manipuladores es fundamental para propósitos de :

- Simulación
- Análisis de sistemas de control
- Diseño de algoritmos de control
- Diseño de la estructura del robot

Como requerimiento esencial para diseñar algoritmos de control de robots manipuladores es fundamental conocer el modelo dinámico, sobre todo cuando la técnica utilizada para diseñar tal algoritmo se encuentra basada en la teoría de estabilidad de Lyapunov [2]. Esto se debe a que en al análisis de estabilidad, las propiedades matemáticas del modelo dinámico son explotadas para facilitar y deducir el análisis.

2.5.1. Ecuaciones de movimiento de Euler Lagrange

La suma de la energía cinética $\mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ más energía potencial $\mathcal{U}(\boldsymbol{q})$ se representa por $\mathcal{H}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ y se denomina el ***hamiltoniano*** del manipulador [2]:

$$\mathcal{H}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \mathcal{U}(\boldsymbol{q})$$

donde:

- $\boldsymbol{q}$, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ representan los vectores de posición y velocidad articular, respectivamente.
- Observe que la energía cinética $\mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ guarda una dependencia de la posición y velocidad articular.
- La energía potencial $\mathcal{U}(\boldsymbol{q})$ esta relacionada con el campo conservativo de la gravedad y por lo tanto únicamente depende de la posición.

Por otra parte el ***lagrangiano*** $\mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ de un robot manipulador de n grados de libertad se define como la diferencia entre la energía cinética y la energía potencial [2]:

$$\mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q}) - \mathcal{U}(\boldsymbol{q}) \tag{2.59}$$

Las ecuaciones de movimiento de Euler - Lagrange representan la mejor alternativa para el modelado de manipuladores debido a sus propiedades matemáticas principalmente. Para un robot manipulador de n grados de libertad se calculan de la siguiente manera [2]:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right] - \frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = \boldsymbol{\tau} - \boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \tag{2.60}$$

Donde:

- $\boldsymbol{q} = [q_1, q_2, ..., q_n]^T \in \mathbb{R}^n$ representa el vector de posiciones articulares o coordenadas generalizadas.
- $\dot{\boldsymbol{q}} = [\dot{q}_1, \dot{q}_2, ..., \dot{q}_n]^T \in \mathbb{R}^n$ es el vector de velocidades articulares.
- $\boldsymbol{\tau} = [\tau_1, \tau_2, ..., \tau_n]^T \in \mathbb{R}^n$ es el vector de pares aplicados.
- $\boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \in \mathbb{R}^n$ es el vector de fuerzas o pares de fricción que depende de la velocidad articular $\dot{\boldsymbol{q}}$ y de la fricción estática $\boldsymbol{f}_e$ presente en las articulaciones del mismo.
- $t \in \mathbb{R}_+$ representa la evolución del tiempo.
- $n \in N$ es el número de grados de libertad.

La energía cinética del manipulador muestra una estructura matemática cuadrática bien definida que se encuentra en función de la velocidad articular [23]:

$$\mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.61}$$

Donde $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ es la matriz de inercia del manipulador, y es una matriz definida positiva y simétrica. Por otro parte la energía potencial $\mathcal{U}(\boldsymbol{q})$ no tiene una forma especifica, pero guarda una dependencia únicamente del vector de posición $\boldsymbol{q}$, esto debido a que este tipo de energía se da como consecuencia de la interacción de campos conservativos como la fuerza de gravedad. Las ecuaciones de movimiento de Euler - Lagrange pueden escribirse en forma compacta como [2]:

$$\frac{\partial \mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})}{\partial \dot{\boldsymbol{q}}} = \frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right] - \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \dot{\boldsymbol{q}}} = M(\boldsymbol{q})\dot{\boldsymbol{q}}$$

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})}{\partial \dot{\boldsymbol{q}}}\right] = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

$$\frac{\partial \mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})}{\partial \boldsymbol{q}} = \frac{\partial}{\partial \boldsymbol{q}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right] - \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}}$$

Finalmente:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{\partial}{\partial \boldsymbol{q}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right] + \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}} + \boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \tag{2.62}$$

2.5.2. Modelo dinámico de un manipulador

El modelo dinámico de un robot manipulador de n grados de libertad, formado con eslabones rígidos conectados por articulaciones libres de elasticidad en cadena cinemática abierta [2]:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \tag{2.63}$$

- $\boldsymbol{q} \in \mathbb{R}^n$ es el vector de coordenadas generalizadas o posiciones articulares.
- $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ es el vector de velocidades articulares.
- $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ es el vector de aceleraciones articulares.
- $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ es la matriz de inercia, la cual es simétrica y definida positiva.
- $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ es la matriz de fuerza centrípetas y de Coriolis.

$$C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{\partial}{\partial \boldsymbol{q}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right] \tag{2.64}$$

- $g(\boldsymbol{q}) \in \mathbb{R}^n$ es el vector de fuerzas o pares gravitacionales obtenido como el gradiente de la energía potencial, es decir:

$$\boldsymbol{g}(\boldsymbol{q}) = \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{2.65}$$

  debida a la acción de la gravedad.

- $\boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \in \mathbb{R}^n$ es el vector de fricción que incluye la frcción viscosa, de Coulomb y estática ($\boldsymbol{f}_e$) de cada articulación del robot.

Metodología para calcular el modelo dinámico

Se retoma una metodología basada en las ecuaciones de Euler - Lagrange para obtener el modelo dinámico de robots manipuladores, consiste en un procedimiento de 4 pasos que se presenta en [2], y que se describe a continuación:

- Paso1
  Obtener la cinemática directa del centro de masa de cada uno de los eslabones. Para el i-ésimo eslabón tomar en cuenta las longitudes anteriores l_{i-1} y ángulos q_i y q_{i-1}:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \boldsymbol{f}_R(l_i, l_{i-1}, l_{ci}, q_{i-1}, q_i) \tag{2.66}$$

- Paso 2
  Calcular la cinemática diferencial del i-ésimo eslabón y deducir la rápidez lineal:

$$\boldsymbol{v}_i = \frac{d}{dt} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{2.67}$$

  La rapidez lineal del centro de masa de cada eslabón se calcula de la siguiente forma:

$$\boldsymbol{v}_i^T \boldsymbol{v}_i = \dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2 \tag{2.68}$$

- Paso 3
  Obtener el lagrangiano $\mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})$:

$$\mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q}) - \mathcal{U}(\boldsymbol{q}) \tag{2.69}$$

  - La energía cinética $\mathcal{K}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ incluye el movimiento de traslación y rotación, por ejemplo:

$$\mathcal{K}_i(\dot{q}_i, \boldsymbol{q}_i) = \frac{1}{2} m_i \boldsymbol{v}_i^T(\dot{q}_i) \boldsymbol{v}_i(\dot{q}_i) + \frac{1}{2} I_i \left[\sum_i^n \dot{q}_i \right]^2 \tag{2.70}$$

    donde I_i es el momento de inercia del i-ésimo eslabón.

- La energía potencial $\mathcal{U}(\boldsymbol{q})$ no tiene una forma especifica como en el caso de la energía cinética, depende de la geometría del robot en general $\mathcal{U}_i(\boldsymbol{q}) = m_i g l_{ci} h_i(\boldsymbol{q})$, siendo $h_i(\boldsymbol{q})$ una función que indica la altura del eslabón con respecto al origen del sistema de referencia del robot.

- Paso 4
  Aplicar las ecuaciones de movimiento de Euler-Lagrange e incluir el modelo de fricción.

$$\tau_i = \frac{d}{dt}\left[\frac{\partial \mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})}{\partial \dot{q}_i}\right] - \frac{\partial \mathcal{L}(\dot{\boldsymbol{q}}, \boldsymbol{q})}{\partial q_i} + b_i \dot{q}_i + f_{ci} signo(\dot{q}_i) + f_{ei}\left[1 - |signo(\dot{q}_i)|\right] \quad (2.71)$$

2.5.3. Modelo dinámico del UR5

Para obtener el modelo de este manipulador se utilizo la metodología de 4 pasos retomada en este trabajo de [2], además de basarse en el trabajo presentado en [21] donde se presenta una manera muy resumida de obtener el modelo dinámico de este manipulador sin embargo el cálculo numérico no se encuentra incluido.

Matriz de inercias

Dado el modelo dinámico de un robot manipulador de n grados de libertad, descrito por la expresión 2.63 la matriz de inercia se puede calcular como:

$$M(\boldsymbol{q}) = \left[\sum_{i=1}^{n} \left(m_i J_{vi}^T J_{vi} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i}\right)\right] \quad (2.72)$$

Donde J_{vi} y $J_{\omega i}$ son la parte lineal y angular de la matriz Jacobiano J_i, del centro de masa de cada eslabón.

Tensores de inercia

Como se puede ver en la ecuación 2.72 es necesario conocer el tensor de inercia de cada enlace para obtener la matriz de inercias. Este en el caso de muchos robots como el UR5 es proporcionado por el fabricante, sin embargo cuando no se proporciona se debe calcular, a continuación a manera de resumen se presenta la forma de como calcularlo.
El tensor de inercia general tiene la forma:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \quad (2.73)$$

donde I_{xx}, I_{yy} y I_{zz} son los principales momentos de inercia y donde $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$, y $I_{yz} = I_{zy}$ son los productos de inercia. Estos se calculan de la siguiente forma:

$$I_{xx} = \int\int\int (y^2 + z^2)\rho(x, y, z)dxdydz \tag{2.74}$$

$$I_{yy} = \int\int\int (x^2 + z^2)\rho(x, y, z)dxdydz \tag{2.75}$$

$$I_{zz} = \int\int\int (x^2 + y^2)\rho(x, y, z)dxdydz \tag{2.76}$$

los productos de inercia se calculan con:

$$I_{xy} = I_{yx} = -\int\int\int xy\rho(x, y, z)dxdydz \tag{2.77}$$

$$I_{xz} = I_{zx} = -\int\int\int xz\rho(x, y, z)dxdydz \tag{2.78}$$

$$I_{yz} = I_{zy} = -\int\int\int yz\rho(x, y, z)dxdydz \tag{2.79}$$

donde $\rho(x, y, z)$ es la densidad de masa del objeto, representada como una función de posición.

Los tensores de inercia para cada eslabón, son proporcionados por el fabricante, estos serán necesarios para obtener la energía cinética del robot manipulador.

$$I_1 = \begin{bmatrix} 0{,}0084 & 0 & 0 \\ 0 & 0{,}0064 & 0 \\ 0 & 0 & 0{,}0084 \end{bmatrix} \tag{2.80}$$

$$I_2 = \begin{bmatrix} 0{,}0078 & 0 & 0 \\ 0 & 0{,}2100 & 0 \\ 0 & 0 & 0{,}2100 \end{bmatrix} \tag{2.81}$$

$$I_3 = \begin{bmatrix} 0{,}0016 & 0 & 0 \\ 0 & 0{,}0462 & 0 \\ 0 & 0 & 0{,}0462 \end{bmatrix} \tag{2.82}$$

$$I_4 = \begin{bmatrix} 0{,}0016 & 0 & 0 \\ 0 & 0{,}0016 & 0 \\ 0 & 0 & 0{,}0009 \end{bmatrix} \tag{2.83}$$

$$I_5 = \begin{bmatrix} 0{,}0016 & 0 & 0 \\ 0 & 0{,}0016 & 0 \\ 0 & 0 & 0{,}0009 \end{bmatrix} \tag{2.84}$$

$$I_6 = \begin{bmatrix} 0{,}0001 & 0 & 0 \\ 0 & 0{,}0001 & 0 \\ 0 & 0 & 0{,}0001 \end{bmatrix} \tag{2.85}$$

Como ya se menciono en caso del robot UR5 los puntos centrales de masa para cada enlace son proporcionados por el fabricante. Sus posiciones están dadas por vectores en los marcos coordenados de su respectivo eslabón. Los vectores y masas se muestran en el cuadro 2.3. Los centros de masas se pueden visualizar en la figura 2.28.

Eslabón	Masa (Kg)	$\boldsymbol{r}^i_{Mi}$ (mm)
1	$M_1 = 3.7$	$\boldsymbol{r}^1_{M1} = [0, -25{,}61, 1{,}93]^T$
2	$M_2 = 8.393$	$\boldsymbol{r}^2_{M2} = [212{,}5, 0, 113{,}36]^T$
3	$M_3 = 2.275$	$\boldsymbol{r}^3_{M3} = [119{,}93, 0, 26{,}5]^T$
4	$M_4 = 1.219$	$\boldsymbol{r}^4_{M4} = [0, -1{,}8, 16{,}34]^T$
5	$M_5 = 1.219$	$\boldsymbol{r}^5_{M5} = [0, 1{,}8, 16{,}34]^T$
6	$M_6 = 0.1879$	$\boldsymbol{r}^6_{M6} = [0, 0, -1{,}159]^T$

Cuadro 2.3: Masas y vectores que apuntan a los centros de masa de cada eslabón

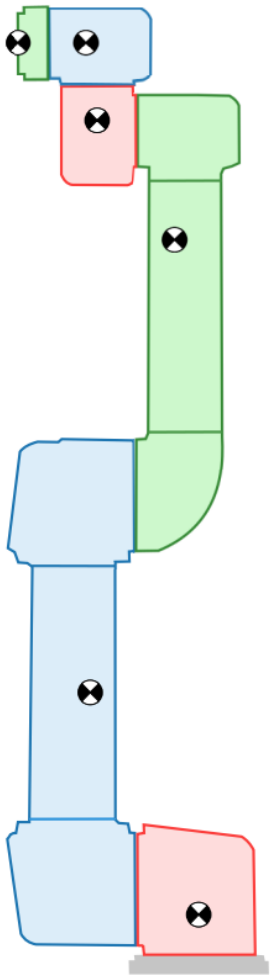

Figura 2.28: Centros de masa de cada eslabón [12]

Jacobiano para los puntos centrales de la masa de los eslabones

Como se puede ver en el cuadro 2.3 el fabricante proporciona la posición de los puntos centrales de masa de cada eslabón. Estos vectores están expresados en un marco de referencia que es adjunto al enlace respectivo y que se denota por $\boldsymbol{r}^j_{Mj}$.

Para calcular la matriz de inercias expresada por la ecuación 2.72 es necesario transformar los vectores de los centros de masa al marco inercial. Esto se puede hacer de la siguiente manera:

$$\boldsymbol{r}^0_{M1} = \boldsymbol{o}^0_1 + R^0_1 \boldsymbol{r}^1_{M1} \tag{2.86}$$
$$\boldsymbol{r}^0_{M2} = \boldsymbol{o}^0_2 + R^0_2 \boldsymbol{r}^2_{M2} \tag{2.87}$$
$$\boldsymbol{r}^0_{M3} = \boldsymbol{o}^0_3 + R^0_3 \boldsymbol{r}^3_{M3} \tag{2.88}$$
$$\boldsymbol{r}^0_{M4} = \boldsymbol{o}^0_4 + R^0_4 \boldsymbol{r}^4_{M4} \tag{2.89}$$
$$\boldsymbol{r}^0_{M5} = \boldsymbol{o}^0_5 + R^0_5 \boldsymbol{r}^5_{M5} \tag{2.90}$$
$$\boldsymbol{r}^0_{M6} = \boldsymbol{o}^0_6 + R^0_6 \boldsymbol{r}^6_{M6} \tag{2.91}$$

donde $\boldsymbol{o}^0_j$ y R^0_j se obtienen, extrayéndolos de las matrices de transformación homogénea expresadas respecto al marco de referencia inercial. Una vez realizado esto se debe calcular el Jacobiano para cada punto central de masa, de hecho aquí solo se debe calcular el Jacobiano lineal ya que la parte angular resulta ser igual a la calculada en el Jacobiano para el efector final.

Matriz de Coriolis

Por otra parte una forma de calcular la matriz de Coriolis $C(\dot{\boldsymbol{q}}, \boldsymbol{q})$ es a través de los símbolos de Christoffel los cuales se definen de la siguiente manera:

$$c_{ij} = \sum_{k=1}^{n} \frac{1}{2} \left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i} \right) \dot{q}_k \tag{2.92}$$

La expresión anterior es consecuencia de la propiedad de pasividad de los robots manipuladores que es resultado de la propiedad de antisimetría de la matriz $\dot{M}(q) - 2C(\dot{\boldsymbol{q}}, \boldsymbol{q})$.

Par gravitacional

Por ultimo solo queda hallar el vector de par gravitacional, cuyos elementos se pueden calcular como:

$$g_i(q) = \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial q_i} \tag{2.93}$$

Obteniendo $M(\boldsymbol{q})$, $C(\dot{\boldsymbol{q}}, \boldsymbol{q})$, $g_i(q)$ y añadiendo algún modelo de fricción $\boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e)$ se completa el modelo dinámico del manipulador.

Implementación en MATLAB

Debido a que los cálculos para hallar la matriz de inercias y de Coriolis son complejos y demasiado largos se requiere de una implementación en software para poder realizarlos con precisión. En el caso del robot UR5 ya se tiene un programa de variables simbólicas que calcula el modelo dinámico y que retorna las matrices de inercia, Coriolis y par gravitacional en unos archivos de texto. De nuestra parte se hicieron algunas modificaciones a los vectores que apuntan a los centros de masa ya que en el programa no eran los mismos que proporciona el fabricante [12], así como los tensores de inercia que también diferían de los propuestos por el fabricante. Una vez realizado esto se extrajeron las matrices correspondientes al modelo dinámico y se realizo una implementación del modelo dinámico en variables de estado para resolverlo mediante la funcione ODE45 de MATLAB que utiliza el algoritmo de integración numérica Runge-Kutta 4/5. Para ello se utiliza una función que devuelve el modelo dinámico del robot en variables de estado y un programa principal que soluciona el sistema. A continuación se muestran las gráficas de los resultados obtenidos, cabe decir que estos resultados corresponden a una entrada cero es decir cuando $\tau = \mathbf{0}$. Las condiciones iniciales que se utilizaron son las siguientes:

$$\boldsymbol{q}_0 = \begin{bmatrix} 0 & \frac{\pi}{2} & \frac{\pi}{4} & -\frac{\pi}{2} & 0 & 0 \end{bmatrix}$$

$$\dot{\boldsymbol{q}}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{q}_0 & \dot{\boldsymbol{q}}_0 \end{bmatrix}^T \tag{2.94}$$

A continuación en las figuras 2.29, 2.30, 2.31 se muestran las gráficas de posición velocidad y aceleración respectivamente. como se puede observar conforme el tiempo pasa ningunas de las variables converge a un valor en especifico, esto se debe a que como ya se dijo no se esta añadiendo ningún torque como entrada y debido a la naturaleza no lineal del modelo dinámico del manipulador, puede llegar a tener un comportamiento caótico y que las variables articulares no se establezcan en algún valor conforme el tiempo tienda a infinito. Finalmente en la figura 2.32 se muestra las posiciones iniciales en las que el robot inicia al momento de sus simulación, las cuales corresponden a las condiciones iniciales para la posición.

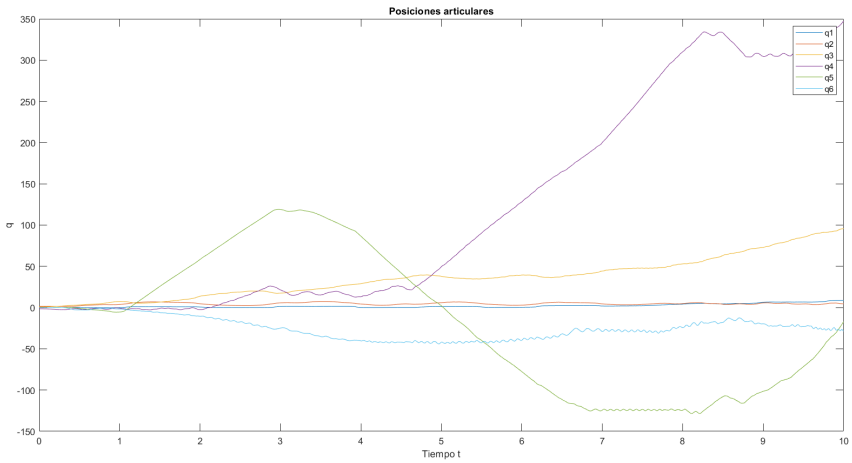

Figura 2.29: Posiciones Articulares $\tau = \mathbf{0}$

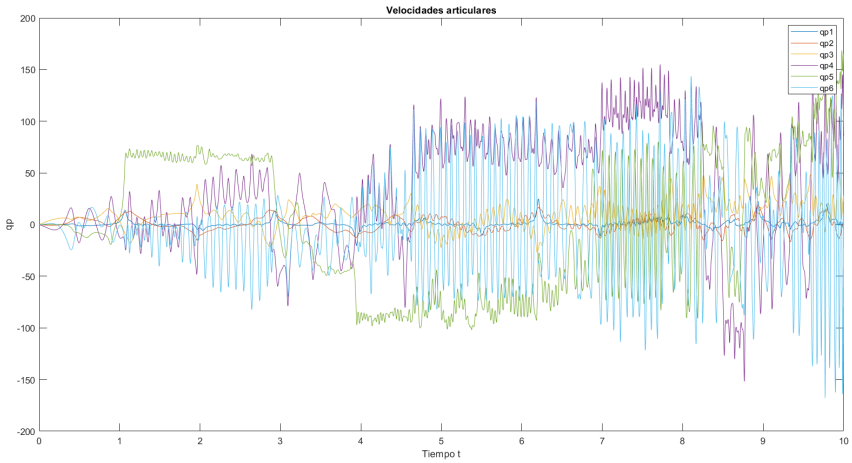

Figura 2.30: Velocidades Articulares $\tau = \mathbf{0}$

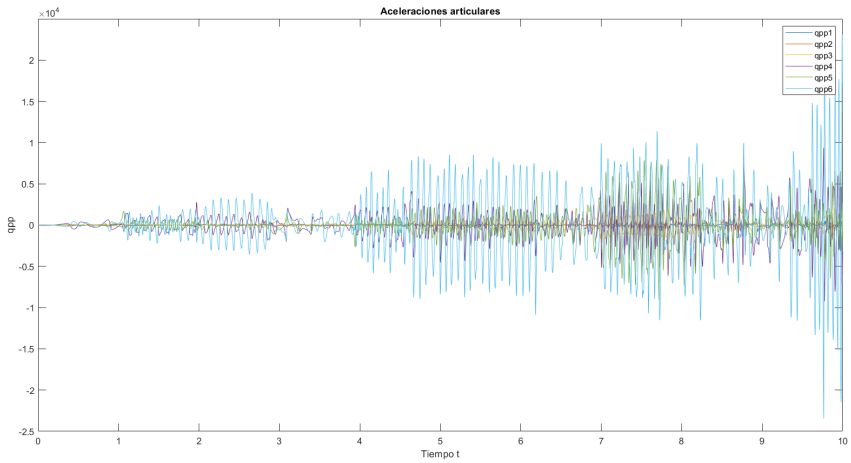

Figura 2.31: Aceleraciones Articulares $\tau = \mathbf{0}$

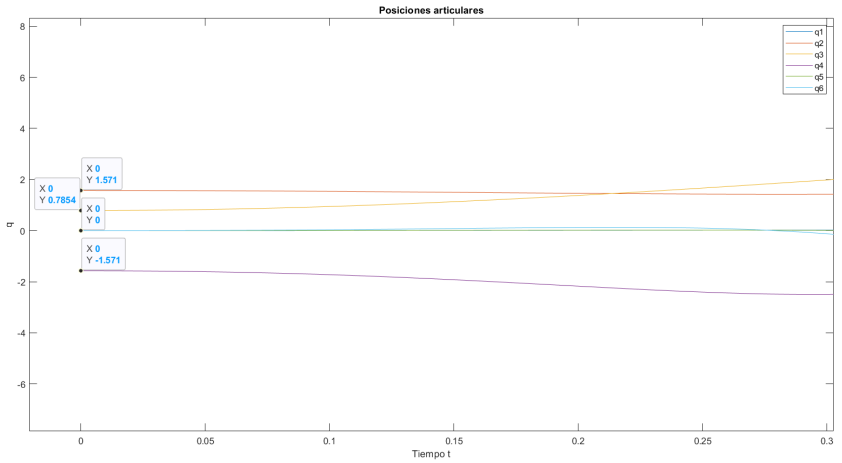

Figura 2.32: Condiciones iniciales para las posiciones articulares

Capítulo 3

Planificación de ruta para manipuladores

En la literatura de planificación de ruta para robots se encontraron una gran diversidad de métodos y algoritmos para plantear el problema de encontrar una ruta libre de colisiones dada una configuración de inicio q_{init} y una configuración de objetivo q_{goal} en un determinado espacio de trabajo del robot. Sin embargo algunos de estos métodos no son factibles para aplicarse a robots manipuladores debido a que varios métodos como los "roadmap" clásicos y el gráfico de Voronoi son aplicables únicamente a robots de dos grados de libertad como por ejemplo un robot planar con articulaciones (q_1, q_2) o un robot móvil con coordenadas en el espacio solamente en (x, y). Puesto que nuestro interés es trabajar con un robot manipulador se han filtrado aquellos métodos que ya han sido aplicados a robots manipuladores y que han conseguido buenos resultados. Entre ellos encontramos 3 principales grupos o familias que son los algoritmos basados en muestreo, los algoritmos basados en campos potenciales y los métodos de descomposición celular. En la primera sección de este capitulo se resaltan a través de una tabla las ventajas y desventajas de estos algoritmos con la finalidad de seleccionar los que presenten las mejores ventajas. De este análisis se concluye que los algoritmos basados en muestreo y los métodos de campos potenciales artificiales son los más factibles para implementarse en un robot manipulador. Una vez realizado esto se seleccionaron 3 algoritmos de las dos familias restantes, dos algoritmos basados en muestreo: RRT y PRM y un algoritmo basado en campos potenciales como métodos factibles de planificación de ruta, por lo que en las secciones siguientes de este capitulo se describen en detalle como funcionan y se describe su algoritmo a manera de pseudocódigo.

3.1. Justificación

Como se ha visto en la literatura sobre algoritmos de planificación de ruta, existen bastantes tipos, cada uno con ventajas y desventajas que son propias de la finalidad para la cual se plantea el algoritmo. Sin embargo, no todos los algoritmos de planificación de ruta son aplicables a un robot manipulador, de hecho muchos son específicos para trabajar con robots que se mueven en un espacio de configuración de dos dimensiones como son los métodos de planificación tipo "roadmap" entre los que encontramos el grafo de visibilidad o el gráfico de Voronoi. Estos algoritmos trabajan muy bien con robots móviles, ya que únicamente se necesitan especificar dos coordenadas (x,y) para conocer la posición del robot móvil , incluso estos métodos se han aplicado satisfactoriamente a robots planares de dos grados de libertad debido a que similar al robot móvil, el robot planar solo necesita conocer los valores de los ángulos de sus dos articulaciones para conocer la posición de su extremo final. Por otra parte encontramos algoritmos de planificación de ruta que se pueden aplicar a un espacio de configuración de mayor dimensión como lo son los algoritmos de descomposición celular, sin embargo cuando el espacio de configuración crece en dimensión, estos algoritmos se vuelven computacionalmente costosos y consumen mucha memoria. Teniendo estas consideraciones en cuenta las cuales derivan del cuadro 3.1, los tipos de algoritmos que se encontraron factibles para ser implementados en un robot manipulador son los algoritmos que se encuentran basados en muestreo y los que utilizan campos potenciales, a continuación se describe la justificación para cada uno de estos tipos de algoritmos.

3.1.1. Algoritmos basados en muestreo

Como ya se revisó, estos algoritmos buscan realizar algún tipo de muestreo en $\mathcal{Q}_{free}$ interconectando de alguna manera aquellas configuraciones que se encuentran libres de colisión y que nos dirigen a la configuración de objetivo. Ahora bien dentro de este grupo encontramos principalmente dos tipos de algoritmos: aquellos que son de consulta única y los de consulta múltiple. Un algoritmo de consulta única, toma en consideración solo a un par de configuración de inicio y objetivo (q_{init}, q_{goal}) y solo queda esperar a que el algoritmo retorne un camino factible o retorne NULL en caso de no hallar tal camino. Por otra parte el algoritmo de consulta múltiple es capaz de recibir varios pares de (q_{init}, q_{goal}) y hallar si es posible el camino entre cada par de configuración de inicio y objetivo. Entendiendo esto y puesto que una de las principales acciones que se pretenden en este trabajo es realizar una comparativa entre algoritmos de planificación, se propone tener en cuenta los siguientes algoritmos basados en muestreo para implementarse en el manipulador.

Algoritmo RRT

Se propone la implementación de este algoritmo debido a que resulta interesante tener un algoritmo de consulta única, ya que este al ocuparse únicamente de un par de

configuraciones (q_{init}, q_{goal}) se centra únicamente en explorar el Q_{free} que es relevante para la consulta, acción por la cual este método tiene buena velocidad de ejecución. Otra de las cualidades de este algoritmo es que la planificación de movimiento, se puede realizar mediante cinemática directa. Este algoritmo de planificación de consulta única, ha sido implementado con éxito en un robot manipulador serial 6R en [34], donde se realiza la planificación de movimiento teniendo en cuenta las restricciones cinemáticas y la posición de los obstáculos que se quieren evitar, después mediante la interpolación de Hermite se realiza el suavizado del camino. Por otra parte en [29] se diseña una variante dinámica del algoritmo RRT que combina la planificación global con la re-planificación local para realizar la planificación de ruta en un entorno dinámico, este algoritmo se implemento en un manipulador de 7 grados de libertad demostrando que puede planificar y ejecutar el camino al tiempo que conserva las limitaciones de la tarea y reacciona a los cambios del entorno en tiempo real.

Algoritmo PRM

Como se busca realizar una comparativa de algoritmos y puesto que ya se propuso implementar el RRT que es de consulta única, la propuesta seria implementar también un planificador que sea de consulta múltiple con el propósito de contrastar esta diferencia. Teniendo en cuenta la literatura revisada, el PRM es el que resulta mas factible para robots manipuladores, puesto que es virtualmente aplicable a cualquier tipo de robot holonómico y trabaja bien con robots de 5 grados de libertad o más [27]. Por otra parte en [28] se propone un Lazy PRM, un planificador que se diferencia del original PRM por minimizar el número de comprobaciones de colisión realizadas durante la planificación y por lo tanto minimiza el tiempo de ejecución del planificador, este algoritmo se implemento en un robot ABB 4400 de 6 grados de libertad, en donde se verifica que este algoritmo tiene tiempos de comprobación de colisiones menores que el PRM y por lo tanto un menor tiempo de ejecución.

3.1.2. Algoritmo basado en campos potenciales

Al igual que el algoritmo RRT, la planificación de ruta mediante los algoritmos que se basan en funciones potenciales se puede realizar mediante cinemática directa. Además dentro de las características encontradas, resulto que estos algoritmos tienen ventajas que pueden ser aprovechadas por un robot manipulador, como que pueden ser aplicables a espacios de configuración multidimensionales y no euclidianos, que el movimiento planificado mediante campos potenciales es muy suave y que muchas de las funciones potenciales se pueden calcular on-line. Desde luego la única desventaja de estos métodos son el problema de los mínimos locales en las funciones potenciales. En [30] se utiliza una nueva función potencial para planificar la ruta de un manipulador de 6 grados de libertad, los resultados muestran que este método puede planificar eficientemente la ruta del manipulador y evitar los obstáculos. Otro trabajo basado en este método se propone en [35] donde se utiliza el método de campo potencial

artificial para la evitación de obstáculos de un robot manipulador redundante de 9 grados de libertad, los resultados corroboran la eficacia que tiene este algoritmo.

Cuadro 3.1: Ventajas y desventajas de los algoritmos

Familia o tipo	Algoritmo	Ventajas	Desventajas
Algoritmos basados en muestreo	PRM (Probabilistic Road Map)	Trabaja bien con robots de varios grados de libertad, 5 o más.	Son necesarias dos fases: una de aprendizaje y otra de consulta.
		El tiempo de construcción de la hoja de ruta va de unos a segundos a algunos minutos.	Se paga un costo inicial en el tiempo de pre-procesamiento del espacio C.
		Los tiempos de consulta se resuelven en una fracción de segundo.	El muestreo aleatorio uniforme de C puede tener dificultados al presentarse el problema del pasaje estrecho.
		Planificador de consultas múltiples.	
	RRT (Rapidly-exploring Random Tree)	Planificador de consulta única.	Fundamentalmente condicional(depende de la configuración de inicio y objetivo.)
		Probabilísticamente completo.	Cuidar el tamaño del paso de exploración en problemas de alta dimensión.
		Permite realizar la planificación mediante cinemática directa.	
		Velocidad de ejecución para cualquier número de dimensiones.	
		Eficiente en la planificación de movimiento cinodinámico.	
Algoritmos basados en campos potenciales	Funciones Potenciales	Evita el uso de la cinemática inversa en el proceso de planificación de ruta.	Presentan mínimos locales en los campos potenciales.

		El movimiento del robot puede llegar a ser muy suave.	Las funciones libres de mínimos locales consumen muchos recursos computacionales.
		Las funciones potenciales se pueden calcular en línea.	
		Existen funciones de potencial libres de mínimos locales.	
		Se aplica a una clase general de espacios de configuración	
	RPP	Planificador de consulta única.	Tiene el defecto de caer en mínimos locales.
		Mejora el método de campos potenciales, generando rutas aleatorias.	Se comporta mal al enfrentarse a problemas del pasaje estrecho.
		Se centra en cubrir las partes de Cfree relevantes para la consulta.	
		Experimentalmente eficiente en robots de 3 a 31 gdl.	
	Wave-Front Planner	Su función potencial tiene un mínimo local.	Solo se puede implementar en espacios representados por cuadriculas.
		Se puede generalizar a problemas de mayor dimensión.	El planificador tiene que buscar un camino en todo el espacio hasta halla Qgoal.
			Su implementación en dimensiones grandes se vuelve computacionalmente intratable.
Algoritmos basados en descomposición celular	Descomposición trapezoidal	Planificador de rutas de cobertura (área).	Solo se aplica a espacios de configuración plana.

		Cada celda se puede cubrir con movimientos simples	Se realiza en dos pasos: localización y búsqueda.
			Fundamentalmente requiere un espacio de trabajo poligonal.
			Computacionalmente caro(depende de la discretización de las celdas)
			Puede producir demasiadas celdas vecinas(cobertura no uniforme)
	Morse Cell Decomposition	Planificador de rutas de cobertura eficiente.	Se puede trabajar en espacios no poligonales.
		Usa funciones morse para mejorar la cobertura de Qfree.	Proporciona una representación más uniforme del Qfree.
		Cada celda se puede cubrir con movimientos simples.	Computacionalmente caro(depende de la discretización de las celdas)
		Se puede aplicar en espacios no planos.	La distribución de sus celdas en Qfree es uniforme.

3.2. Algoritmo PRM básico

Este algoritmo se realiza en dos fases [24] y [27], la primera llamada fase de aprendizaje en la que se construye una hoja de ruta de manera probabilística para un determinado espacio de trabajo. Esta hoja de ruta se encuentra representada por un grafo no dirigido $G = (V, E)$. Los nodos en V son un conjunto de configuraciones de robot elegidas por algún método sobre $\mathcal{Q}_{free}$, generalmente la generación de configuraciones se realiza a partir de una distribución uniforme. Por otro lado las aristas E corresponden a caminos entre los nodos generados, es decir, una arista $(q1, q2)$ corresponde a una ruta libre de colisiones que conecta las configuraciones $q1$ y $q2$. A estos caminos se les denomina caminos locales y se calculan mediante un planificador local. Intuitivamente la forma más simple de conectar dos configuraciones es mediante una linea recta si es que esta se puede establecer en el par de configuraciones analizado.
La segunda fase de este algoritmo es la de consulta [27], en la cual se utiliza la hoja de ruta para resolver problemas de planificación de rutas individuales. Entonces dada una configuración inicial q_{init} y una configuración de objetivo q_{goal}, este método intentara conectar q_{init} y q_{goal} a dos nodos q' y q'' respectivamente, en V. En caso de tener exito el planificador busca en el gráfico G una secuencia de aristas en E que conecten q' con q''. Finalmente esta secuencia es transformada en una ruta factible para el robot al volver a calcular los caminos locales correspondientes y concatenándolos. Aunque la fase de aprendizaje generalmente se realiza antes de cualquier consulta las dos fases también se pueden entrelazar. Sin embargo vale la pena gastar un tiempo considerable en la fase de aprendizaje si la hoja de ruta se utilizara para resolver muchas consultas. A continuación se describen las dos fases de este algoritmo las cuales se retoman de [24], [27] y [28].

3.2.1. Fase de aprendizaje o construcción del roadmap

El algoritmo 1 describe los pasos para la construcción de la hoja de ruta y la figura 3.1 describe gráficamente este proceso, tener en cuenta que:

- sea Δ el planificador local con entrada $(q, q') \in \mathcal{Q}_{free} \times \mathcal{Q}_{free}$ que retorna un camino libre de colisiones de q a q' o NULL si no puede encontrar dicha ruta. En principio asumir que Δ es simétrico y determinista.

- sea *dist* una función $\mathcal{Q} \times \mathcal{Q} \to \mathbb{R}^+ \cup \{0\}$ llamada *función distancia* que usualmente es una métrica en $\mathcal{Q}$.

Inicialmente el gráfico $G(V, E)$ está vacío. Posteriormente una configuración es muestreada de $\mathcal{Q}$ repetidamente. Suponer que el muestreo se realiza con una distribución aleatoria uniforme. si la configuración está libre de colisiones se agrega a la hoja de ruta. Este proceso se repite hasta que se hayan muestreado las n configuraciones libres de colisiones. Luego para cada nodo $q \in V$ se toma un conjunto N_q de k vecinos mas cercanos a la configuración q de acuerdo con alguna metrica *dist* son elegidos de V, entonces se llama al planificador local para conectar q a cada nodo $q' \in N_q$. Si el

planificador local tiene exito en calcular una ruta factible entre q y q' la arista $(q,\ q')$ se agrega a la hoja de ruta.

Algoritmo 1 Construcción de la hoja de ruta

Entrada:

n: numero de nodos para colocar en la hoja de ruta

k: numero de vecinos más cercanos a examinar para cada configuración

Salida:

Una hoja de ruta $G = (V, E)$

1: $V \leftarrow \emptyset$
2: $E \leftarrow \emptyset$
3: **while** $|V| < n$ **do**
4: **repeat**
5: $q \leftarrow$ una configuración aleatoria en $\mathcal{Q}$
6: **until** q este libre de colisión
7: $V \leftarrow V \cup \{q\}$
8: **end while**
9: **for all** $q \in V$ **do**
10: $N_q \leftarrow$ los k vecinos mas cercanos de q escogidos de V conforme a $dist$
11: **for all** $q' \in N_q$ **do**
12: **if** $(q, q') \notin E$ **and** $\Delta(q, q') \neq$ NULL **then**
13: $E \leftarrow E \cup \{(q, q')\}$
14: **end if**
15: **end for**
16: **end for**

3.2.2. Fase de consulta

Durante esta fase, se encuentran rutas entre configuraciones de entrada arbitrarias q_{init} y q_{goal} utilizando la hoja de ruta construida en la fase de aprendizaje, esto se ilustra en la figura 3.2. El algoritmo 2 muestra este proceso. Para ello se debe suponer que $\mathcal{Q}_{free}$ esta conectado y que la hoja de ruta consiste de un único componente conectado. La pregunta principal es cómo conectar q_{init} y q_{goal} a la hoja de ruta. Puesto que se requiere que las consultas sean lo mas rápido posibles entonces se desea un algoritmo ligero aquí. La estrategia que se usa en el algoritmo 2 para conectar q_{init} a la hoja de ruta es considerar los k nodos mas cercanos en la hoja de ruta en orden de distancia creciente de q_{init} de acuerdo a la metrica $dist$ e intentar conectar cada uno de ellos con el planificador local hasta que alguna conexión se realiza correctamente. De la misma forma se utiliza este procedimiento para conectar q_{goal} con la hoja de ruta. Si se logra conectar exitosamente q_{init} y q_{goal} a la hoja de ruta el camino más corto es encontrado de acuerdo a $dist$ donde se puede usar el algoritmo de Dijkstra´s o el algoritmo A*. Si se desea en esta parte el camino se puede mejorar ejecutando algún algoritmo de suavizado de pos-procesamiento.

Algoritmo 2 Resolver consulta

Entrada:

q_{init}: configuración inicial
q_{goal}: configuración objetivo
k: numero de vecinos más cercanos a examinar para cada configuración
$G = (V,\ E)$: la hoja de ruta obtenida por el algoritmo 1

Salida:

Un camino desde q_{init} a q_{goal} o FAILURE

1: $N_{q_{init}} \leftarrow$ los k vecinos mas cercanos de q_{init} provenientes de V de acuerdo a $dist$
2: $N_{q_{goal}} \leftarrow$ los k vecinos mas cercanos de q_{goal} provenientes de V de acuerdo a $dist$
3: $V \leftarrow \{q_{init}\} \cup \{q_{goal}\} \cup V$
4: establecer q' ser el vecino mas cercano de q_{init} en $N_{q_{init}}$
5: **repeat**
6: **if** $\Delta(q_{init}, q') \neq$ NIL **then**
7: $E \leftarrow (q_{init}, q') \cup E$
8: **else**
9: establecer q' ser el siguiente vecino mas cercano de q_{init} en $N_{q_{init}}$
10: **end if**
11: **until** una conexión sea exitosa o el conjunto $N_{q_{init}}$ este vació.
12: establecer q' ser el vecino más cercano de q_{goal} en $N_{q_{goal}}$
13: **repeat**
14: **if** $\Delta(q_{goal}, q') \neq$ NIL **then**
15: $E \leftarrow (q_{goal}, q') \cup E$
16: **else**
17: establecer q' ser el siguiente vecino mas cercano de q_{goal} en $N_{q_{goal}}$
18: **end if**
19: **until** una conexión sea exitosa o el conjunto $N_{q_{goal}}$ este vacío
20: $P \leftarrow$ camino mas corto $(q_{init},\ q_{goal},\ G)$
21: **if** P no esta vacío **then**
22: **return** P
23: **else**
24: **return** FAILURE
25: **end if**

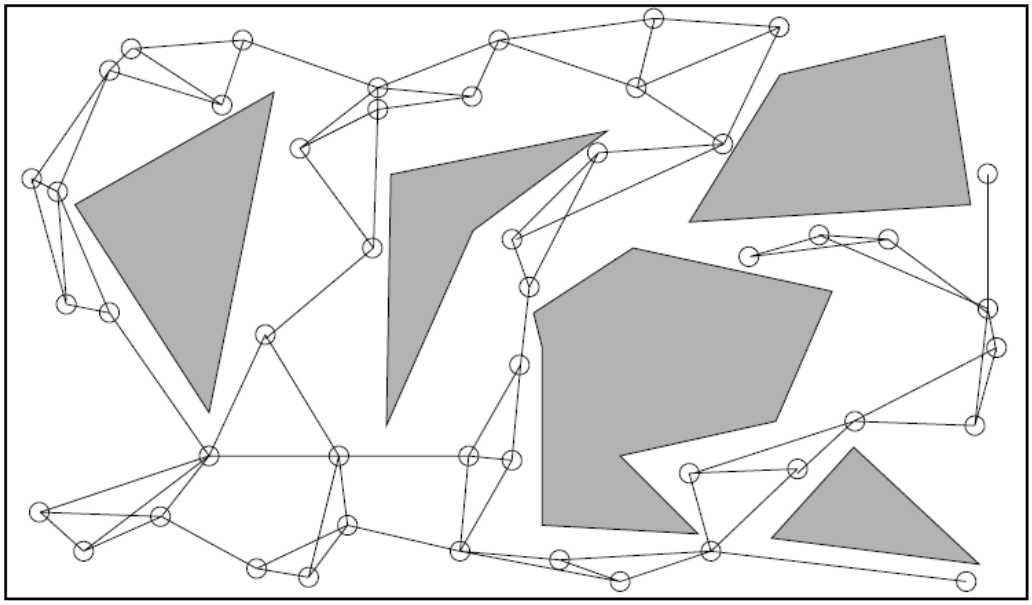

Figura 3.1: Construcción de la hoja de ruta [24]

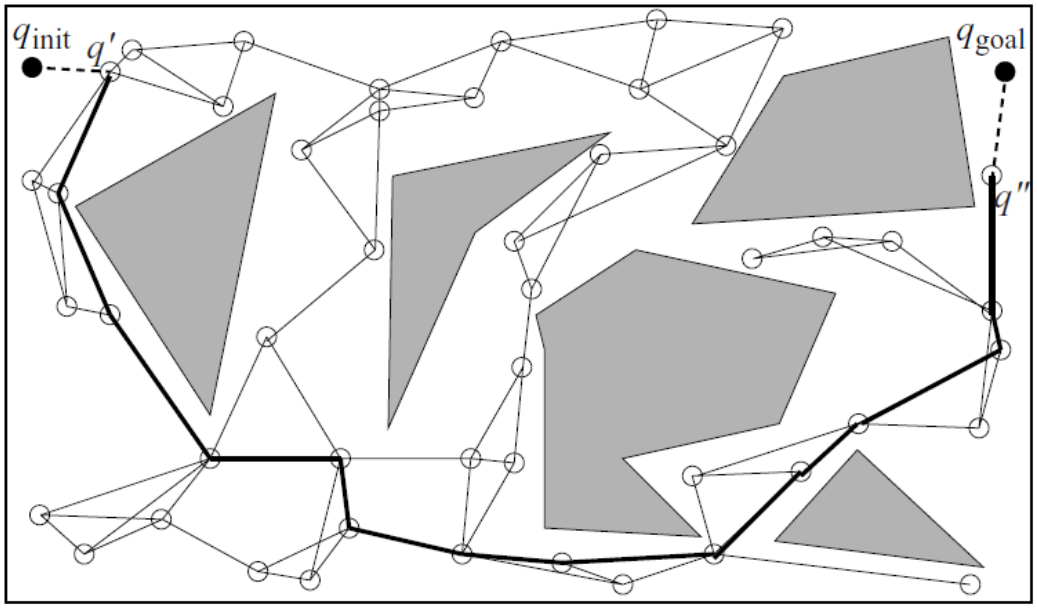

Figura 3.2: Fase de consulta en el PRM [24]

3.3. Algoritmo RRT

El RRT es un algoritmo de consulta única que destaca entre otros factores por su velocidad de ejecución para cualquier número de dimensiones [29]. Esto se debe en gran medida a que al enfocarse en una única consulta RRT desarrolla una estrategia de muestreo que se encarga de cubrir las partes de Q_{free} que son relevantes para la consulta [24]. El objetivo original de este método consiste en construir un árbol de exploración que proporcionara una cobertura uniforme de todo el espacio libre de colisión, esta idea se muestra en el algoritmo 3 retomado de [32]. Como se observa el objetivo del algoritmo es seleccionar un punto q_{rand} de forma aleatoria y extender hacia él el árbol de configuraciones. Esto se hace mediante el uso de la función *Extiende* que se encarga de ampliar el árbol en el sentido que marca q_{rand} determinando si existe un camino libre de colisión. La estructura de esta función se muestra en el algoritmo 4 descrito en [32].

Algoritmo 3 Algoritmo RRT

Entrada:

q_{init}: configuración inicial
K_{max}: número de iteraciones

Salida:

Un árbol aleatorio de exploración rápida

```
1: Arbol[0] = q_init
2: for k = 1 hasta K_max do
3:    q_rand = ConfiguraciónAleatoria()
4:    Extiende(Arbol, q_rand)
5: end for
6: return Arbol
```

La estructura de la función *Extiende* comienza con el calculo de la configuración mas cercana a q_{rand} y se le denomina q_{near}. Esto calculo se realiza mediante la función *VecinoMásPróximo* que se encarga de aplicar una métrica R a todos los vertices del árbol, obteniendo así el punto mas cercano a q_{rand}. Posteriormente la función *NuevaConfiguración* calcula q_{new}, el nuevo punto a agregar, mediante un salto de tamaño ϵ partiendo de q_{near} en dirección hacía q_{rand}, la figura 3.3 ilustra este planteamiento. Al momento que se obtiene q_{new} se considera si existe o no colisión en este movimiento, devolviendo verdadero si no hay colisión y falso si la hay. Si no se encuentra colisión, se agrega q_{new} al árbol y se distinguen dos casos. Primero, si el punto aleatorio q_{rand} se encuentra dentro de un circulo con centro q_{near} y radio ϵ, entonces se considera que se ha alcanzado q_{rand} y se retorna *alcanzado*. En el segundo caso, si q_{rand} no se encuentra dentro de este circulo entonces se retorna *avanzado*. Finalmente si existe alguna colisión de q_{near} hacía q_{new}, se informa de que no ha habido ramas y el punto no se agrega al árbol.

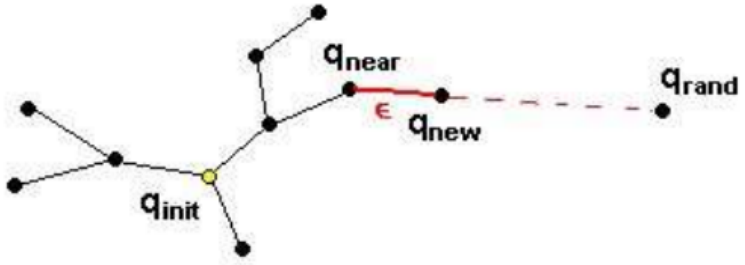

Figura 3.3: Funcionamiento Algoritmo RRT [32]

Algoritmo 4 Función Extiende (Arbol, q_{rand})

Entrada:

Arbol: RRT dentro del algoritmo 3

q_{rand}: una configuración aleatoria

Salida:

Añadir vértices al árbol o devolver alcanzado, avanzado, rechazado

1: q_{near} = VecinoMásPróximo (q_{rand}, Arbol)
2: **if** NuevaConfiguración($q_{rand}, q_{near}, q_{new}$) **then**
3: AñadeVértice(Arbol, q_{new})
4: **if** $q_{new} = q_{rand}$ **then**
5: **return** alcanzado
6: **else**
7: **return** avanzado
8: **end if**
9: **else**
10: **return** rechazado
11: **end if**

3.3.1. RRT para el manipulador UR5

Habiendo revisado la manera en como trabaja el algoritmo RRT se hace la propuesta de un algoritmo RRT que sea aplicable a brazos manipuladores, para ello se tomo como punto de partida a los algoritmos 3 y 4 de esta sección. El resultado se muestra en el algoritmo 5. Como se puede ver en la linea 1 y 2 del algoritmo se obtienen los parámetros geométricos del robot y de su entorno. Posteriormente se inicializa el árbol en la configuración inicial q_{init}, luego se entra a un bucle regido por el número de iteraciones máximo K_{max}, inmediatamente se manda a llamar a la función *NewPoint* que recibe la configuración inicial q_{init} y entra en un bucle *while* del cual solo se puede salir si la configuración aleatoria q_{rand} generada a partir de q_{init} no invade los límites físicos del robot y se encuentra libre de colisión con algún elemento del entorno. En caso de presentar colisiones o entrar en los límites de las articulaciones del UR5 se permanecerá en el ciclo *while* hasta que la configuración aleatoria se encuentre libre

de choque. Finalmente el árbol se extiende con cada punto nuevo calculado.

Algoritmo 5 Algoritmo RRT para implementar en el UR5

Entrada:

q_{init}: configuración inicial
K_{max}: número de iteraciones

Salida:

Un árbol aleatorio de exploración rápida

```
1: robot = modeloUR5()
2: entorno = ModeloDeColisión()
3: Arbol[0] = q_init
4: for k=1 hasta K_max do
5:    NewPoint(q_init)
6:    while (ComprobarLimites=1 o ComprobarColisiones=1) do
7:       q_rand = ConfiguraciónAleatoria(q_init)
8:       np = cinematicaUR5(q_rand, robot)
9:       ComprobarLimites = LimitesUR5(q_rand, robot)
10:      ComprobarColisiones = Colisiones(q_rand, robot, entorno)
11:   end while
12:   q_init = q_rand
13:   newPoint = np
14:   Extiende (Arbol, newPoint)
15: end for
16: return Arbol
```

3.4. Campos Potenciales

Limitada por la estructura mecánica del manipulador y la relación de acoplamiento entre las articulaciones, la planificación de ruta de evitación de obstáculos para un brazo robótico es más complicada que el robot móvil [31]. Los métodos tradicionales de planificación de ruta para manipuladores planifican la trayectoria final del manipulador y luego descomponen las restricciones entre las articulaciones para buscar a través de la cinemática directa resolver el movimiento del manipulador. Por otra parte se debe tener en cuenta que los métodos tradicionales tienen un límite en la estructura de los robots, es decir se requiere que la solución cerrada de cada articulación del brazo mecánico se obtenga mediante un método analítico a través de la pose final del robot [**?**]. Sin embargo debido al acoplamiento dinámico "fuerte" de las juntas, incluso si se tiene una solución de cinemática directa resuelta por un método analítico se cuenta con problemas de "soluciones múltiples " o soluciones singulares. Esto quiere decir que estos métodos resuelven el problemas de evitación de planificación de movimiento a través de variables conjuntas [31]. La principal ventaja de estos métodos es que cuentan con una solución óptima global, mientras que la desventaja es la complejidad de la estructura del algoritmo la cual aumenta exponencialmente con el número de articulaciones [24]. Alternativamente el método de potencial artificial crea un campo atractivo alrededor del objetivo y un campo repulsivo alrededor de los obstáculos de tal forma que el manipulador "navega" hacia el objetivo mientras evita los obstáculos. Sin embargo puesto que este método solo considera el espacio local, el manipulador puede caer en un mínimo local provocando que el manipulador deje de moverse u oscile cerca del obstáculo [35]. Teniendo en cuenta las deficiencias del método de campos potenciales, muchos investigadores han tratado de resolver el problema de los mínimos locales empleando diversas técnicas. En [35] se salta del mínimo local a través de una fuerza de control adicional. Sin embargo al aplicarla, la dirección de convergencia de la energía potencial resultaba incierta. Por otra parte en [37] se escapa del mínimo local utilizando al RRT para generar caminos aleatorios alrededor del mínimo local y logra escapar de este. Sin embargo la estructura de estos algoritmos es complicada y no garantiza el rendimiento en tiempo real. Por otra parte en [31] se propone un método interesante basado en campos potenciales que escapa del mínimo local agregando obstáculos virtuales para emular que en estos puntos el campo debe ser repulsivo para evitar que el robot caiga en ellos. Este método tuvo buenos resultados al ser aplicado en un robot de 6 gdl. A continuación se muestra una idea de algoritmo basado en esta ultima alternativa:

Algoritmo 6 Idea del algoritmo de campos potenciales

Entrada:

K_{max}: número de iteraciones

Salida:

Planificación exitosa o planificación fallo

```
1: Inicialización de parámetros
2: for k=1 hasta K_max do
3:    Generar una combinación de ángulos para cada una de las articulaciones del
      robot q_i.
4:    Cada ángulo de articulación se combina entre si para generar diferentes situa-
      ciones.
5:    Se obtiene la combinación de ángulos donde la energía potencial total se mini-
      mice.
6:    if ¿Se cae en un mínimo local? then
7:       Agregar obstáculos virtuales.
8:    else
9:       if ¿Se llegó al punto destino? then
10:         return Planificación exitosa
11:      else
12:         Planificación fallo
13:      end if
14:   end if
15: end for
```

Capítulo 4

Controladores de trayectoria

En términos prácticos, la "regulación" o control de posición de robots manipuladores consiste en hallar una función vectorial τ tal que el error de posición $\tilde{\boldsymbol{q}}(t)$ y la velocidad de movimiento $\dot{\boldsymbol{q}}$ convergen asintóticamente a cero, sin importar las condiciones iniciales $\tilde{\boldsymbol{q}}(0)$ y $\dot{\boldsymbol{q}}(0)$. Sin embargo dados los términos en que se plantea un regulador no nos permite controlar la velocidad de un robot, si no que mas bien esta se usa como inyección de amortiguamiento. La cuestión anterior no es ningún problema cuando se tiene como referencia una posición constante $\boldsymbol{q}_d$, sin embargo cuando se requiere que la referencia sea variable $\boldsymbol{q}_d(t)$ el controlador de posición puede presentar un comportamiento inadecuado, pese a esto si este controlador es de alto desempeño , por ejemplo uno basado en una estructura tangente hiperbólica se puede seguir la referencia variable sin problemas. Dado la cuestión anterior es que surge el control de trayectoria. Resaltando, la estructura de un controlador de trayectoria es un poco más compleja que la de un control o regulador de posición, debido a distintos aspectos, como lo son: controlar el error de posición articular $\tilde{\boldsymbol{q}}(t)$ y el error de velocidad articular $\dot{\tilde{\boldsymbol{q}}}(t)$, incluir la dinámica del manipulador en su estructura matemática y el uso del Jacobiano analítico para realizar la transformación de planear trayectorias en el espacio cartesiano y realizar el mapeo al espacio articular. En este trabajo se decidieron usar dos controladores de trayectoria: el controlador Par Calculado y el controlador PD+. La elección de utilizar el controlador Par Calculado es debido a que garantiza estabilidad asintótica global y calcula la cantidad de energía exacta que requieren los servomotores para llevar a cabo una trayectoria planeada, por otro lado el controlador PD+ es una variación del controlador Par Calculado, la ventaja de este controlador radica en que la matriz de inercia se encuentra fuera del lazo de realimentación de la posición y velocidad articular. Esto permite una descomposición del esquema de control, en un desacoplo del esquema de control articular PD *más* un bucle de compensación dinámica, atributo que le dio el nombre de PD+. Resumiendo, este tipo de controladores nos permite controlar posición y velocidad pero su desempeño y buena implementación están relacionados con la precisión con la cual se conozcan los parámetros dinámicos del manipulador.

4.1. Norma $\mathcal{L}_2$

La norma $\mathcal{L}_2$ es ampliamente utilizada para evaluar el desempeño de algoritmos de control de robots manipuladores, el resultado de procesar esta norma da como resultado un número que se calcula con la ecuación 4.1:

$$\mathcal{L}_2[\boldsymbol{f}] = \sqrt{\frac{1}{t}\int_0^t \boldsymbol{f}^T(\psi)\boldsymbol{f}(\psi)d\psi} = \sqrt{\frac{1}{t}\int_0^t \|\boldsymbol{f}(\psi)\|^2 d\psi} < \infty \tag{4.1}$$

Donde $\boldsymbol{f}$ serán los errores de posición $\tilde{\boldsymbol{q}}$, velocidad $\dot{\tilde{\boldsymbol{q}}}$ y aceleración $\ddot{\tilde{\boldsymbol{q}}}$, dependiendo del desempeño que se quiera medir, en este trabajo solo se presenta la norma $\mathcal{L}_2$ para el error de posición y error de velocidad en los controladores, en el avance se cuenta con los cálculos restantes.

4.2. Par Calculado

La estructura de este controlador es la siguiente:

$$\boldsymbol{\tau} = M(\boldsymbol{q})[\ddot{\boldsymbol{q}}_d + K_p\tilde{\boldsymbol{q}} + K_v\dot{\tilde{\boldsymbol{q}}}] + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_d + B\dot{\boldsymbol{q}}_d + \boldsymbol{g}(\boldsymbol{q}) \tag{4.2}$$

Donde K_p y $K_v \in \mathbb{R}^{n\times n}$ son matrices definidas positivas correspondientes a la ganancia proporcional y derivativa respectivamente. La combinación del modelo dinámico y la estructura de control Par Calculado deriva en la siguiente ecuación en lazo cerrado:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} = \begin{bmatrix}\dot{\tilde{\boldsymbol{q}}}\\ -K_p\tilde{\boldsymbol{q}} - K_v\dot{\tilde{\boldsymbol{q}}}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & I\\ -K_p & K_v\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} \tag{4.3}$$

4.3. PD+

La estructura de este controlador se describe en la siguiente ecuación:

$$\boldsymbol{\tau} = K_p\tilde{\boldsymbol{q}} + K_v\dot{\tilde{\boldsymbol{q}}} + M(\boldsymbol{q})\ddot{\boldsymbol{q}}_d(t) + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_d + B\dot{\boldsymbol{q}}_d + \boldsymbol{g}(\boldsymbol{q}) \tag{4.4}$$

Donde K_p y $K_v \in \mathbb{R}^{n\times n}$ son matrices definidas positivas correspondientes a la ganancia proporcional y derivativa respectivamente. La ecuación en lazo cerrado de este controlador con el modelo dinámico del manipulador se describe en la siguiente expresión:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} = \begin{bmatrix}\dot{\tilde{\boldsymbol{q}}}\\ M(\boldsymbol{q})^{-1}[-K_p\tilde{\boldsymbol{q}} - K_v\dot{\tilde{\boldsymbol{q}}}]\end{bmatrix} = \begin{bmatrix}\mathbf{0} & I\\ -M(\boldsymbol{q})^{-1}K_p - M(\boldsymbol{q})^{-1}K_v & \end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\tilde{\boldsymbol{q}}}\end{bmatrix} \tag{4.5}$$

4.4. Implementación en MATLAB

Una vez desarrollado el modelo dinámico del manipulador UR5 y estudiado las estructuras de los controladores que se van a utilizar, se procedió a realizar una simulación en el software de MATLAB de las ecuaciones en lazo cerrado de los controladores con el modelo dinámico. De la manera mas general la implementación cuenta con una función que contiene al modelo dinámico descrito en variables de estado, otra función para describir la estructura de cada controlador y por ultimo un programa principal que nos permite solucionar el sistema. Para realizar dicha solución se utiliza ODE45 de MATLAB que utiliza el algoritmo de integración Runge-Kutta.

4.4.1. Resultados del controlador Par Calculado

Como trayectoria deseada en la implementación de este controlador se planifica en el espacio cartesiano una rosa polar de 5 pétalos que se encuentra dentro de los parámetros geométricos del robot, es decir que sea físicamente alcanzable, posteriormente se hace uso de la cinemática inversa para obtener el vector de posiciones articulares deseadas $\boldsymbol{q}_d(t)$, luego para obtener el vector de velocidades articulares deseadas $\dot{\boldsymbol{q}}_d(t)$ y el vector de aceleraciones articulares deseadas $\ddot{\boldsymbol{q}}_d(t)$ se hace uso del Jacobiano analítico que se explico anteriormente. Para los valores de ganancias se propusieron valores de tal forma que los valores de torque de los servomotores que utiliza el UR5 no se saturen. En la tabla 4.1 se describen estos valores:

Cuadro 4.1: Tabla de ganancias del controlador Par Calculado

Articulación	τi_{max}(Nm)	kp_i	kvi
1	150	75	60
2	150	12	9,6
3	150	15	12
4	28	22,4	13,44
5	28	22,4	13,44
6	28	22,4	13,44

En la figura 4.1 en azul se puede observar a la trayectoria propuesta, una rosa polar de 5 pétalos k=5 descrita con un radio r=0.2 y que se encuentra trazada en el primer cuadrante. En rojo se puede observar a la trayectoria realizada por el robot, como se puede apreciar se parte de las condiciones iniciales $\boldsymbol{q} = [\frac{\pi}{2}\ \frac{\pi}{2}\ -\frac{\pi}{2}\ -\frac{\pi}{2}\ \frac{\pi}{2}\ 0]^T$ y $\dot{\boldsymbol{q}} = [0\ 0\ 0\ 0\ 0\ 0]^T$ mapeadas al espacio cartesiano mediante la cinemática directa, luego se aprecia la respuesta transitoria del sistema y finalmente el seguimiento adecuado de la trayectoria propuesta como parte de la respuesta estable del sistema. Esto se puede verificar en las gráficas de error de posición y error de velocidad que se muestran en la figura 4.2. El tiempo que tarda en estabilizarse este controlador es de t=4.788s. Por otra parte en la figura 4.3 podemos observar que los torques solicitados por el

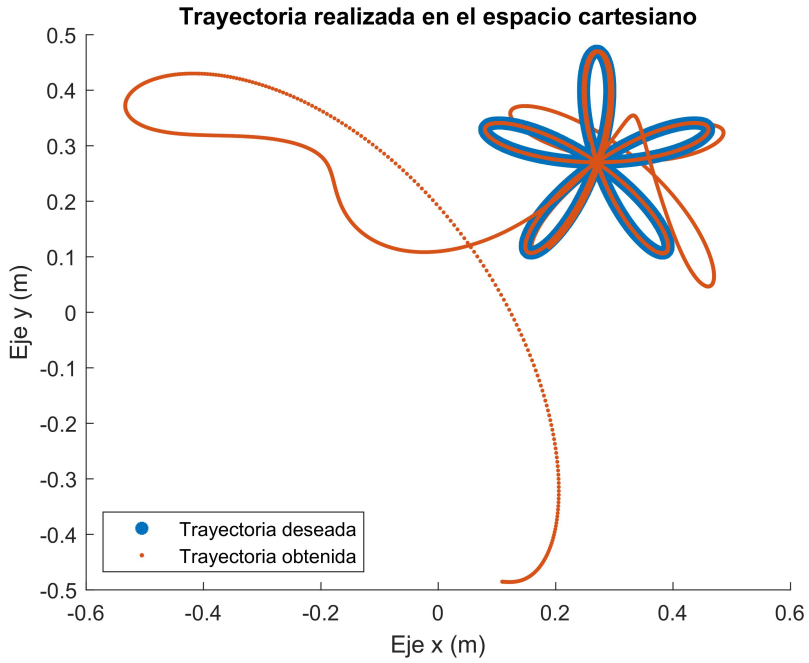

Figura 4.1: Trayectorias deseada y obtenida en el Par Calculado

controlador se mantienen muy por debajo de su valor máximo lo cual es adecuado. Sin embargo al inicio como es normal debido al cálculo de energía que realiza el algoritmo de control por la distancia en la que se encuentra la posición deseada y la actual se puede observar un sobre-impulso en los torques, pero aun así estos valores se mantienen dentro del rango de trabajo.

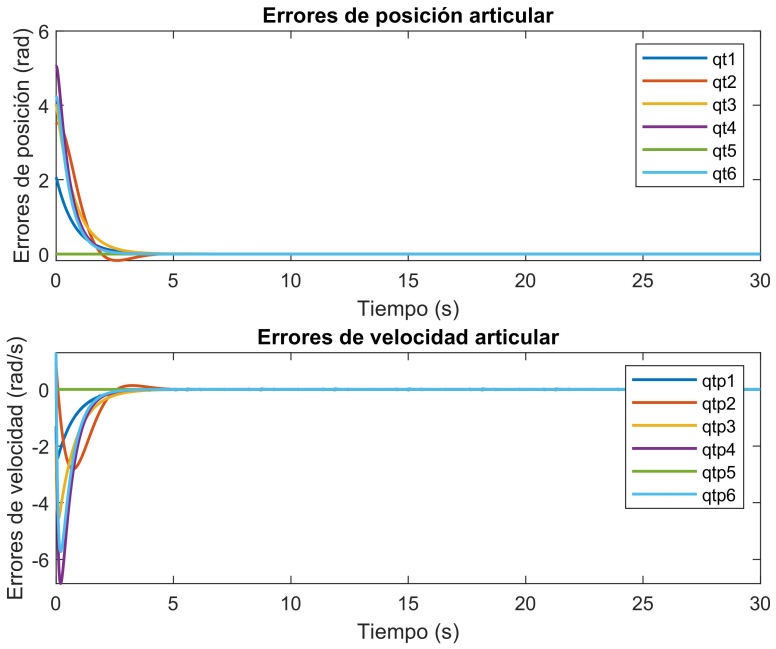

Figura 4.2: Errores de posición y velocidad en el Par Calculado

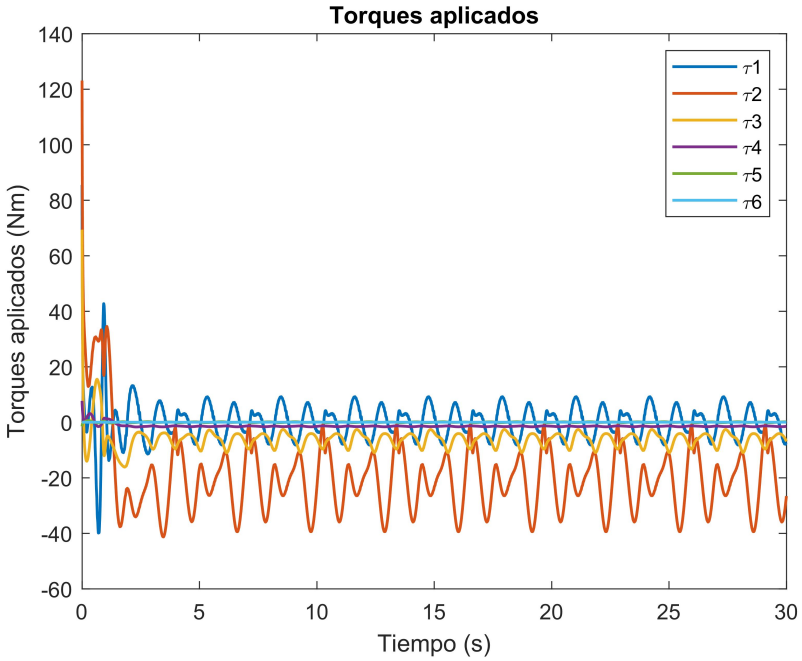

Figura 4.3: Torques aplicados en el Par Calculado

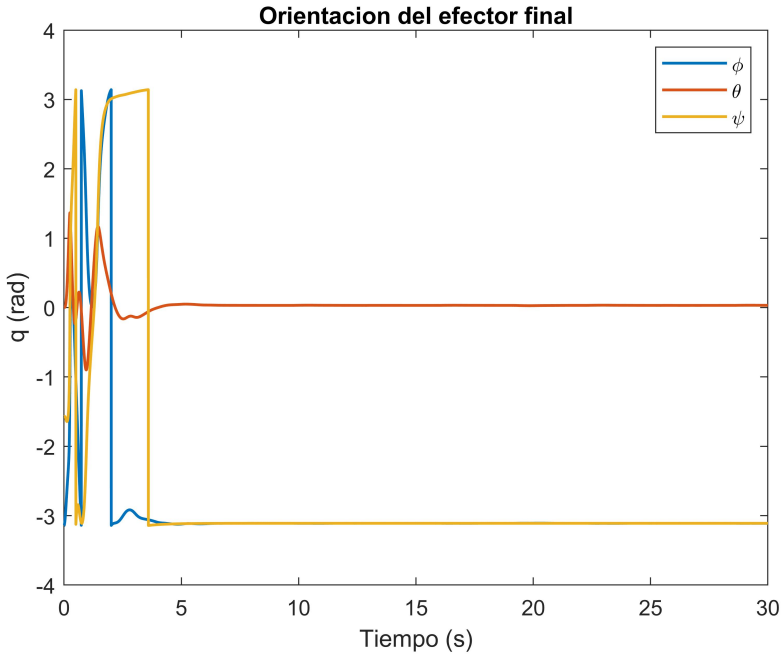

Figura 4.4: Orientación del efector final en el Par Calculado

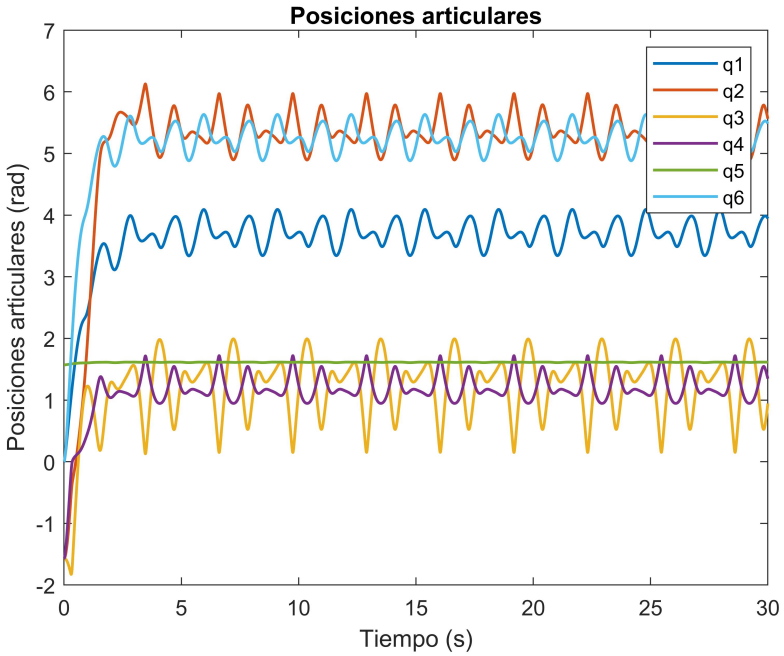

Figura 4.5: Posiciones articulares en el Par Calculado

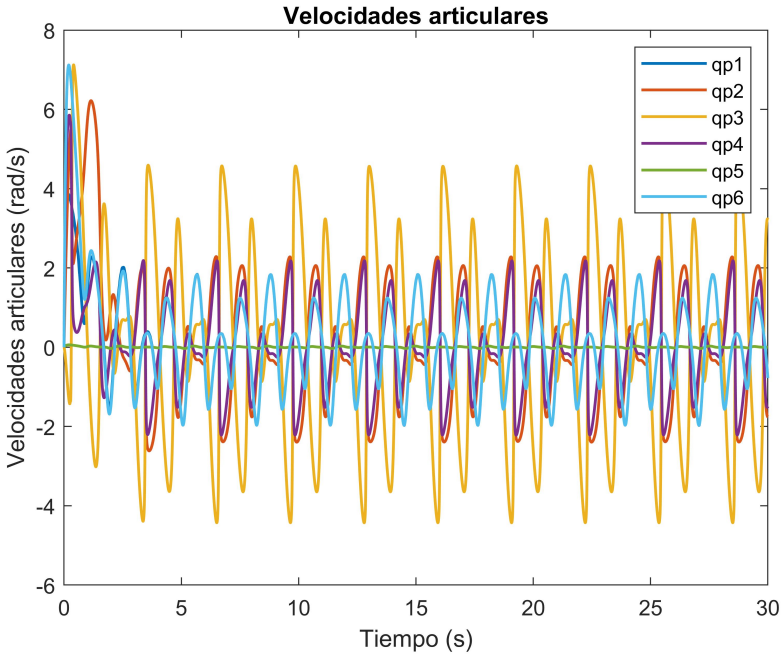

Figura 4.6: Velocidades articulares en el Par Calculado

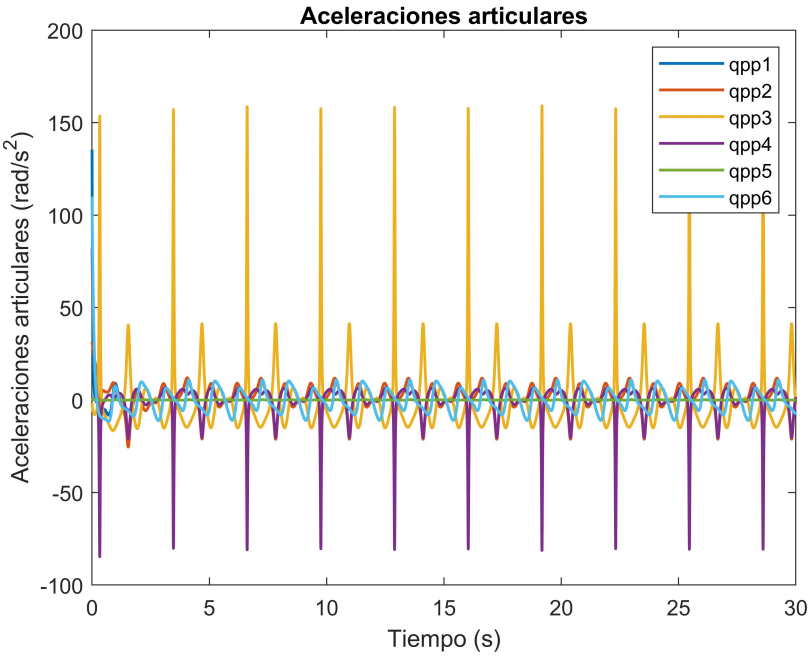

Figura 4.7: Aceleraciones articulares en el Par Calculado

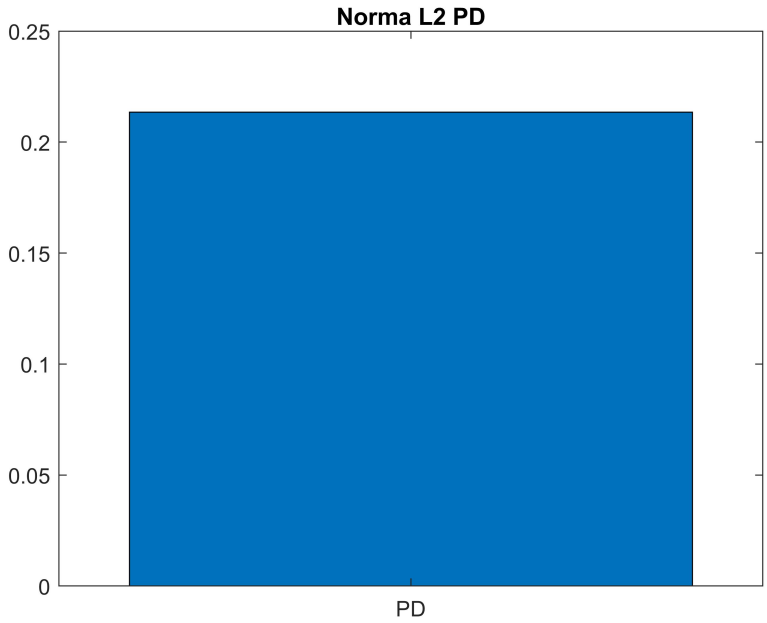

Figura 4.8: Norma $\mathcal{L}_2$ en el Par Calculado

4.4.2. Resultados del controlador PD+

Posteriormente se procedió a realizar la implementación de la ecuación en lazo cerrado descrita por la expresión 4.5 que combina el modelo dinámico del manipulador con el controlador PD+, los valores de ganancias para las matrices K_p y K_v se muestran en la tabla 4.2. En la implementación de este controlador se utilizan las mismas condiciones iniciales que en el controlador Par Calculado.

Cuadro 4.2: Tabla de ganancias del controlador PD+

Articulación	τi_{max}(Nm)	kp_i	kvi
1	150	75	60
2	150	7,5	6
3	150	37,5	30
4	28	2,8	1,68
5	28	22,4	13,44
6	28	2,8	1,68

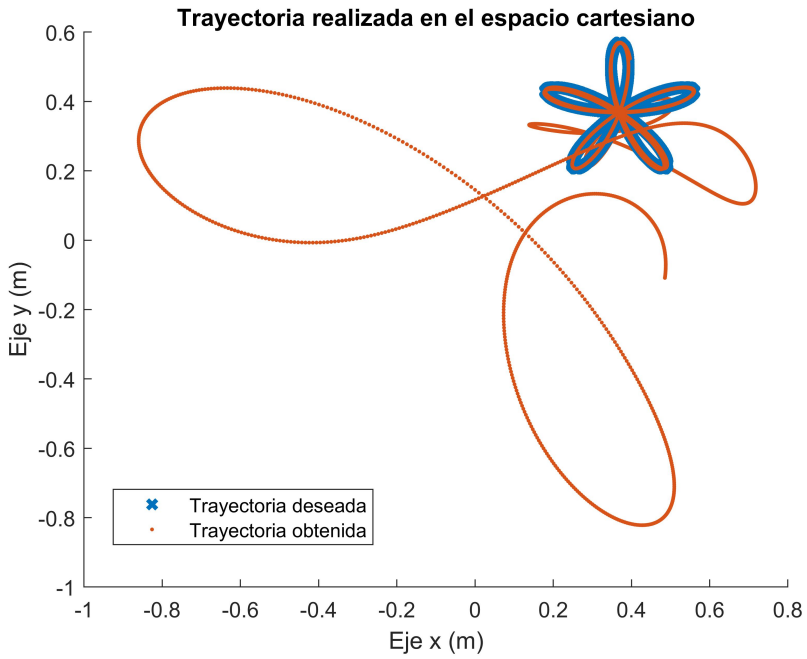

Figura 4.9: Trayectorias deseada y obtenida en el PD+

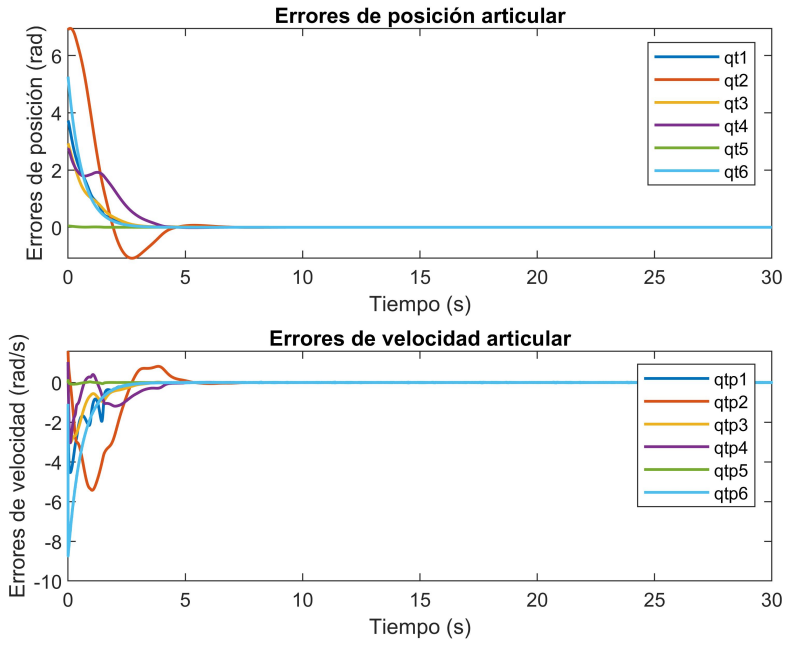

Figura 4.10: Errores de posición y velocidad en el PD+

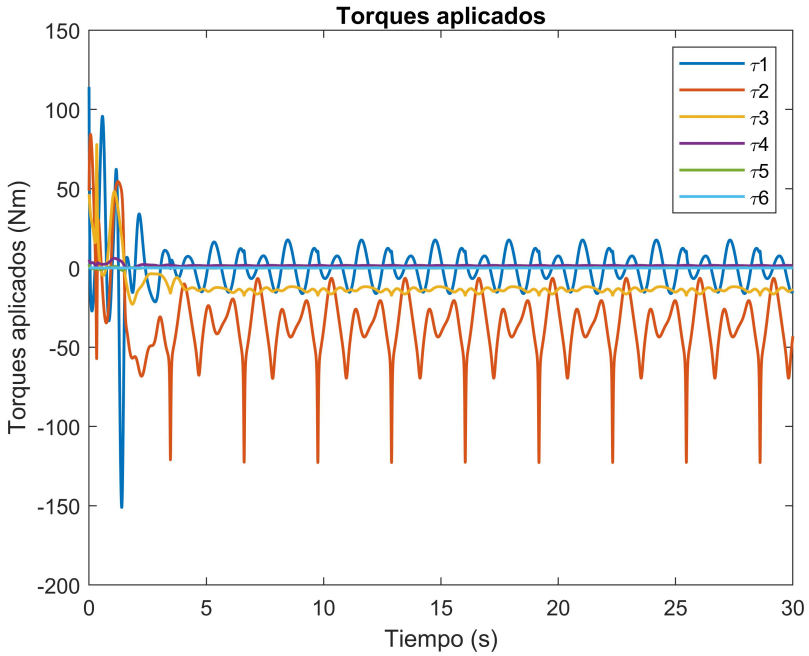

Figura 4.11: Torques aplicados en el PD+

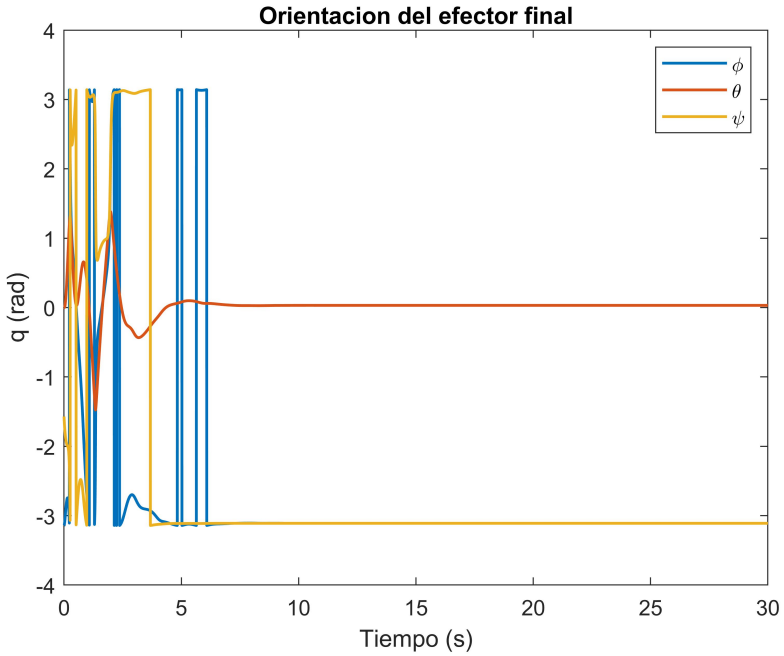

Figura 4.12: Orientación del efector final en el PD+

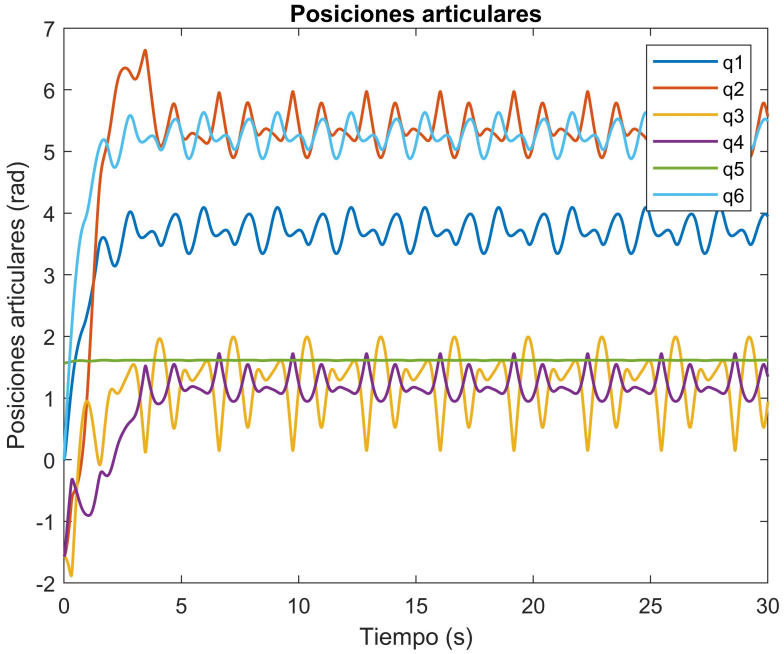

Figura 4.13: Posiciones articulares en el PD+

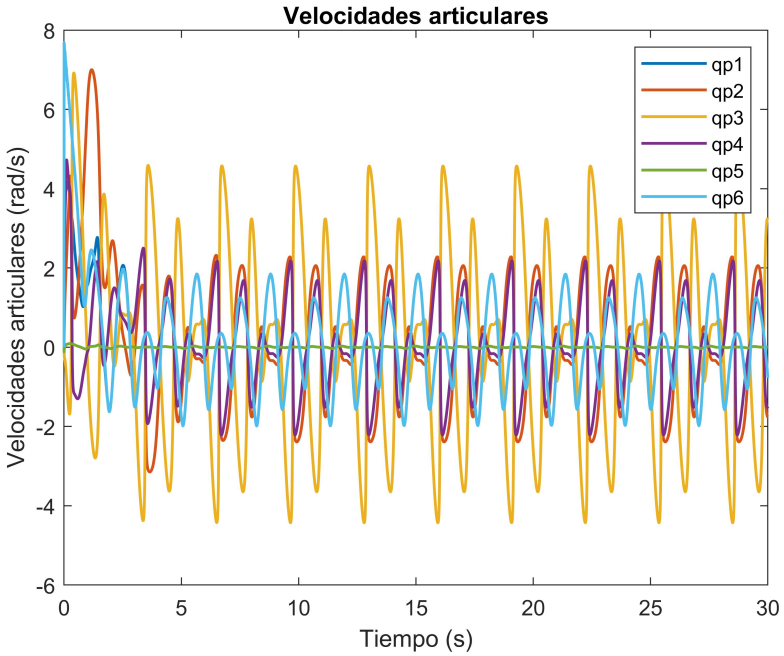

Figura 4.14: Velocidades articulares en el PD+

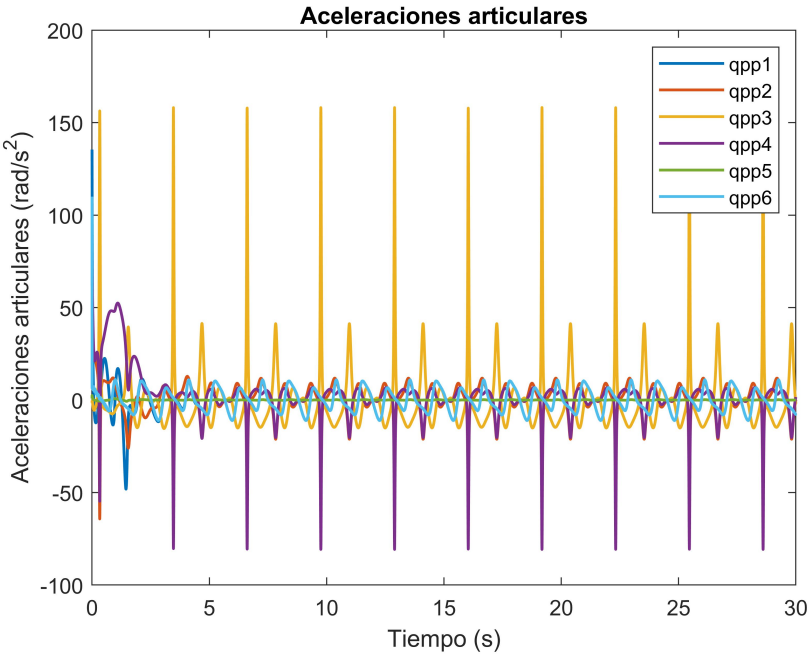

Figura 4.15: Aceleraciones articulares en el PD+

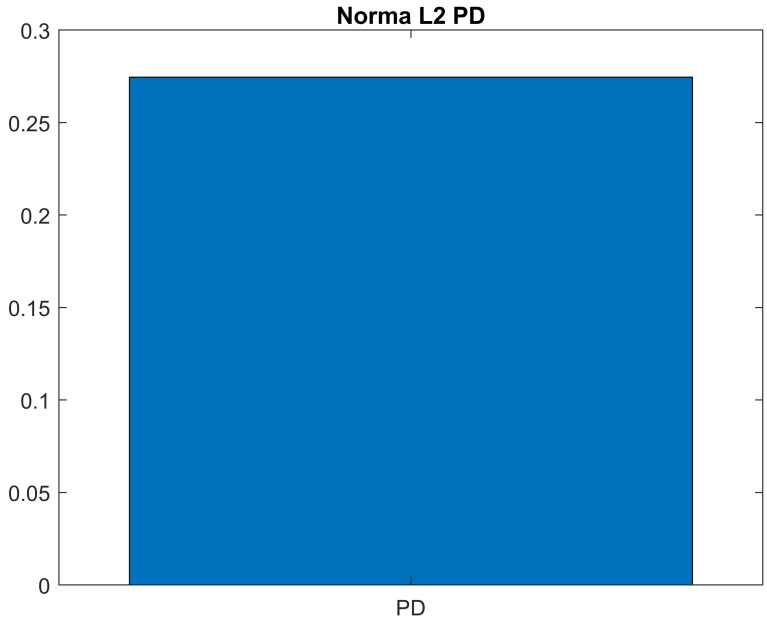

Figura 4.16: Norma $\mathcal{L}_2$ en el PD+

Capítulo 5

Nuevo esquema de regulación desarrollado

En la actualidad el control de robots manipuladores representa un tema de gran importancia debido al gran impacto que tiene la robótica en el desarrollo de la ciencia e industria. Los robots manipuladores han pasado a operar prácticamente todas las actividades cotidianas que el ser humano pueda imaginar con un alto grado de precisión, destreza y rapidez. Esto ha hecho que los robots manipuladores se posicionen como máquinas muy eficientes en los trabajos que realizan.
Para que un manipulador realice una tarea adecuadamente se requiere de un algoritmo de control. Existen infinidad de algoritmos de control, de hecho su diseño se ha convertido en una actividad científica permanente [1] con el objetivo de proponer nuevos esquemas cuyas características mejoren el desempeño en el control de robots manipuladores conforme a las características requeridas.
Una técnica ampliamente usada en la literatura para el diseño de algoritmos de control es el moldeo de energía [2], que utiliza el teorema de invarianza de La Salle [3] para demostrar estabilidad asintótica. Este método se ha refinado a través de la propuesta de una función estricta de Lyapunov para demostrar la estabilidad asintótica del esquema de control propuesto. En [4] se propone una función estricta de Lyapunov para estudiar la estabilidad asintótica del control PD con compensación de gravedad. Mas tarde en [5], Kelly y Santibañez generalizan la función estricta que se propone en [6] para el estudio de los problemas de regulación y control de trayectoria.
La técnica de moldeo de energía mas inyección de amortiguamiento ha sido la base para crear muchas familias de esquemas de control de robots manipuladores, ver [1], [7], [8], [9], [10]. Su principio se estudia en [8], este se basa en obtener esquemas de control a partir del gradiente de la energía potencial del robot manipulador, posteriormente se agrega una inyección de amortiguamiento y finalmente se agrega una compensación de los pares gravitacionales del robot en caso de ser necesario .
El problema de control de posición o regulación es un caso particular del control de trayectoria, la regulación corresponde a seguir una referencia deseada constante en el tiempo. Desde el punto de vista de estabilidad el regulador debe ser diseñado tal que

las posiciones del robot manipulador convergen asintóticamente hacia las posiciones deseadas conforme el tiempo evoluciona sin importar las condiciones iniciales [11], [12], [13]. Desde el punto de vista de moldeo de energía esto es posible debido a que la función que se propone como gradiente de la energía potencial es una función que depende del error, la cual es continua en su dominio, tiene un valor de cero únicamente cuando el error es cero y se vuelve una función definida positiva al ser multiplicada por su argumento [3].
Una vez comprendida la técnica de moldeo de energía se pueden proponer diferentes funciones para "moldear"la energía potencial artificial tal que la función candidata de Lyapunov sea definida positiva. En este punto encontramos diferentes funciones utilizadas que cumplen con esta característica como lo son algunas funciones hiperbólicas, exponenciales, cuadráticas y racionales, revisar [1], [7], [8], [9], [11], [14], [15]. Se puede deducir que estas funciones se pueden combinar para crear otras funciones con diferentes atributos, parte clave de la infinita gama de esquemas de control que se pueden diseñar. La estructura de control que se propone en este trabajo esta basada en una combinación de la función tangente hiperbólica y la función coseno hiperbólico a través de una razón u cociente, este esquema tiene la ventaja de tener una respuesta rápida y presentar menor sobre-tiro en los valores de torque que el controlador tangente hiperbólica original.

5.1. Diseño del esquema de control

La técnica propuesta por Takegaki y Arimoto para el diseño de esquemas de control que es conocida como moldeo de energía mas inyección de amortiguamiento se describe matemáticamente como:

$$\boldsymbol{\tau} = \nabla \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}}) - \boldsymbol{f}_v(K_v, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \tag{5.1}$$

donde $\mathcal{U}_a(K_p > 0$ es la energía potencial artificial, el término $\nabla \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}}) = \frac{\partial}{\partial \tilde{\boldsymbol{q}}} \mathcal{U}_a(K_p \tilde{\boldsymbol{q}})$ describe el esquema de control propuesto y físicamente es el moldeo de energía de $\mathcal{U}_a(K_p, \tilde{\boldsymbol{q}})$ a través del gradiente $\nabla \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}})$. $K_p \in \mathbb{R}^{n \times n}$ es una matriz diagonal definida positiva que representa la ganancia proporcional. $\tilde{\boldsymbol{q}}$ es el error de posición articular. $\boldsymbol{f}_v(K_v, \dot{\boldsymbol{q}})$ es una función disipativa tal que: $\dot{\boldsymbol{q}}^T \boldsymbol{f}_v(K_v, \dot{\boldsymbol{q}}) > 0$. $K_v \in \mathbb{R}^{n \times n}$ es una matriz diagonal definida positiva que representa la ganancia derivativa. $\dot{\boldsymbol{q}}$ es la velocidad articular. $\boldsymbol{g}(\boldsymbol{q})$ es la compensación de gravedad.
El gradiente $\nabla \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}})$ como propuesta para el esquema de control debe existir como función continua en $\tilde{\boldsymbol{q}}$, unicamente es cero cuando el error es cero: $\nabla \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}}) = \boldsymbol{0} \iff \tilde{\boldsymbol{q}} = \boldsymbol{0}$ y debe satisfacer que $\tilde{\boldsymbol{q}}^T \nabla \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}}) > 0$. Aqui podemos proponer diferentes estructuras en cuanto a la experiencia personal. Sin embargo como regla de diseño también podemos proponer una función candidata de Lyapunov:

$$V(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) = \frac{1}{2} \dot{\boldsymbol{q}}^T M(\boldsymbol{q}) \dot{\boldsymbol{q}} + \mathcal{U}_a(K_p, \tilde{\boldsymbol{q}}) > 0 \Longrightarrow \dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) \leq 0 \tag{5.2}$$

el termino $\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} > 0$ corresponde a la energía cinética del manipuladores y es definido positivo por tener una estructura cuadrática. Por ende la función que corresponde a la energía potencial artificial $\mathcal{U}_a(K_p, \tilde{\boldsymbol{q}})$ debe ser propuesta tal que $V(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) > 0$, si esto se cumple se garantiza que su derivada temporal sera semi-definida negativa $\dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) \leq 0$, luego, por el principio de invarianza de LaSalle podemos demostrar estabilidad asintótica del punto de equilibrio. La función candidata de Lyapunov que se propone es:

$$V(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} + \begin{bmatrix} \sqrt{log(cosh(\tilde{q}_1)) - sech(\tilde{q}_1) + 1} \\ \sqrt{log(cosh(\tilde{q}_2)) - sech(\tilde{q}_2) + 1} \\ \vdots \\ \sqrt{log(cosh(\tilde{q}_n)) - sech(\tilde{q}_n) + 1} \end{bmatrix}^T K_p \begin{bmatrix} \sqrt{log(cosh(\tilde{q}_1)) - sech(\tilde{q}_1) + 1} \\ \sqrt{log(cosh(\tilde{q}_2)) - sech(\tilde{q}_2) + 1} \\ \vdots \\ \sqrt{log(cosh(\tilde{q}_n)) - sech(\tilde{q}_n) + 1} \end{bmatrix} \tag{5.3}$$

en la ecuación 5.3 solo debemos demostrar que el termino correspondiente a la energía potencial artificial es definido positivo. Para ello tenemos que la función: $log(cosh(\tilde{\boldsymbol{q}}))-sech(\tilde{\boldsymbol{q}})$ es semi-definida positiva ya que $log(cosh(\boldsymbol{0}) - sech(\boldsymbol{0}) = -\boldsymbol{1}$ es por ello que en la expresión 5.3, se suma 1 para lograr $log(cosh(\boldsymbol{0})) - sech(\boldsymbol{0}) + \boldsymbol{1} = \boldsymbol{0}$. Finalmente el radical $\sqrt{log(cosh(\tilde{\boldsymbol{q}})) - sech(\tilde{\boldsymbol{q}}) + \boldsymbol{1}} > \boldsymbol{0}$ es definido positivo. Sin embargo con el propósito de comprobar y visualizar que en efecto la energía potencial artificial propuesta es definida positiva se presenta la gráfica 5.1. En esta gráfica se puede notar que usamos la variable $\boldsymbol{x}$ como argumento de la función, esto nos sirve para visualizar el perfil que tendrá la energía potencial artificial propuesta, además con esta gráfica podemos verificar que la estructura de la función propuesta es definida positiva y que por ende se hace cero únicamente en $f(0) = 0$. Así la función candidata de Lyapunov tiene una estructura adecuada. Posteriormente se procede a calcular la derivada temporal de la función candidata de Lyapunov con la finalidad de obtener el gradiente de la función de energía potencial artificial que se propuso:

$$\dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}^T M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - [tanh(\tilde{\boldsymbol{q}}) - sech(\tilde{\boldsymbol{q}})tanh(\tilde{\boldsymbol{q}})]^T K_p \dot{\boldsymbol{q}} \tag{5.4}$$

usando identidades trigonométricas básicas y simplificando:

$$\dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}^T M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \left[\frac{tanh(\tilde{\boldsymbol{q}})(1 + cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})}\right]^T K_p \dot{\boldsymbol{q}} \tag{5.5}$$

de la expresión 5.5 extraemos el termino que contiene a la ganancia proporcional $\left[\frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})}\right]$ y definimos el esquema del regulador propuesto:

$$\boldsymbol{\tau} = K_p \frac{tanh(\tilde{\boldsymbol{q}})(1 + cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})} - K_v \frac{tanh(\dot{\boldsymbol{q}})(1 + cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} + \boldsymbol{g}(\boldsymbol{q}). \tag{5.6}$$

como se puede observar se usa la misma estructura para representar al gradiente de la energía potencial $\nabla\mathcal{U}_a(K_p, \tilde{\boldsymbol{q}})$ y para la función disipativa o de amortiguamiento $\boldsymbol{f}_v(K_v, \dot{\boldsymbol{q}})$, por ultimo se añade la compensación de los pares gravitacionales $\boldsymbol{g}(\boldsymbol{q})$.

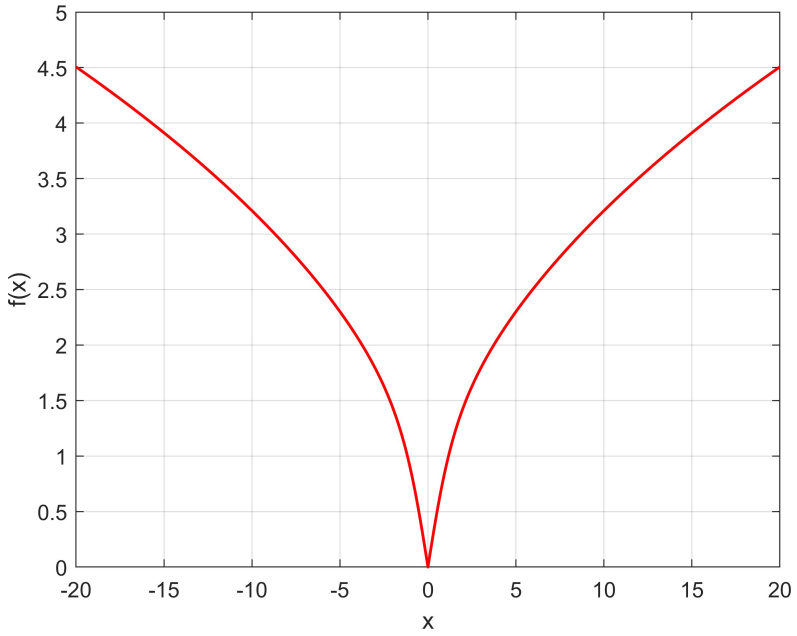

Figura 5.1: Comportamiento de la función $\sqrt{log(cosh(\boldsymbol{x}) - sech(\boldsymbol{x}) + \mathbf{1}}$

5.2. Demostración de estabilidad asintótica

La ecuación en lazo cerrado del esquema de control propuesto con el modelo dinámico de un robot manipulador de n grados de libertad es:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\boldsymbol{q}}\\ \dot{\boldsymbol{q}}\end{bmatrix} = \begin{bmatrix} -\dot{\boldsymbol{q}} \\ M(\boldsymbol{q})^{-1}[K_p\frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})} - K_v\frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} - \\ -C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - B\dot{\boldsymbol{q}}] \end{bmatrix} \tag{5.7}$$

5.2.1. Existencia y unicidad del punto de equilibrio

Se tiene que:

$$-\dot{\boldsymbol{q}} = -I\dot{\boldsymbol{q}} \Rightarrow -I\dot{\boldsymbol{q}} = \mathbf{0} \Leftrightarrow \dot{\boldsymbol{q}} = \mathbf{0}$$

Para el otro término tenemos que:

$$M(\boldsymbol{q})^{-1}[K_p\frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})} - K_v\frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} - \\ -C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - B\dot{\boldsymbol{q}}] = 0 \tag{5.8}$$

Para lo cual sabemos que:

- $M(\boldsymbol{q})^{-1} > 0$ por lo que no puede ser 0.

- Por otra parte sabemos que la matriz de Coriolis evaluada $\dot{\boldsymbol{q}} = \mathbf{0}$ se vuelve una matriz de ceros $\in \mathbb{R}^{n\times n}$.
- $B\dot{\boldsymbol{q}} = 0 \Leftrightarrow \dot{\boldsymbol{q}} = 0$ ya que $B > 0$.
- Los términos $K_p \frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})}$ y $K_v \frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})}$ se hacen cero, si y solo si $\tilde{\boldsymbol{q}} = \mathbf{0}$ y $\dot{\boldsymbol{q}} = \mathbf{0}$ ya que $K_p > 0$ y $K_v > 0$ son matrices definidas positivas.

Con los argumentos anteriores queda demostrada la existencia y unicidad del punto de equilibrio.
Sustituyendo la aceleración articular, ecuación 5.7 en la derivada de la función candidata de Lyapunov, ecuación 5.5 se tiene:

$$\begin{aligned}\dot{V}(\tilde{\boldsymbol{q}},\dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}^T K_p \frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})} - \dot{\boldsymbol{q}}^T K_v \frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} \\ - \dot{\boldsymbol{q}}^T C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^T B\dot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \left[\frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})}\right]^T K_p\dot{\boldsymbol{q}}\end{aligned} \tag{5.9}$$

simplificando se tiene que:

$$\dot{\boldsymbol{q}}^T K_p \frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})} - \left[\frac{tanh(\tilde{\boldsymbol{q}})(1+cosh(\tilde{\boldsymbol{q}}))}{cosh(\tilde{\boldsymbol{q}})}\right]^T K_p\dot{\boldsymbol{q}} = \mathbf{0}$$

luego por la propiedad de antisimetría: $-\dot{\boldsymbol{q}}^T C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} = \mathbf{0}$, entonces la expresión se reduce a:

$$\dot{V}(\tilde{\boldsymbol{q}},\dot{\boldsymbol{q}}) = -\dot{\boldsymbol{q}}^T K_v \frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} - \dot{\boldsymbol{q}}^T B\dot{\boldsymbol{q}} \tag{5.10}$$

Para esta última expresión el término $\dot{\boldsymbol{q}}^T B\dot{\boldsymbol{q}} > 0$, es definido positivo entonces, $-\dot{\boldsymbol{q}}^T B\dot{\boldsymbol{q}} < 0$ es definido negativo. Luego $\dot{\boldsymbol{q}}^T K_v \frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} > 0$ es definido positivo puesto que $K_v > 0$ y además el cociente $\frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})}$ es una función impar que tiene la estructura del controlador. Como estamos multiplicando por $\dot{\boldsymbol{q}}^T$ a la izquierda, este término se vuelve una función par y estrictamente definida positiva, ahora puesto que tiene multiplicando un signo menos, este termino se vuelve definido negativo, es decir: $-\dot{\boldsymbol{q}}^T K_v \frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} < 0$. En la figura 5.2 se muestra la gráfica que representa la estructura del controlador y se puede ver que es una función saturada e impar. Esta estructura también se utiliza para la inyección de amortiguamiento. Luego en la figura 5.3 podemos visualizar, a la estructura de control una vez que se multiplica por su argumento, lo que hace que sea una función definida negativa y en términos de energía se convierte en una función disipativa que propiciara la estabilidad del sistema. Con los argumentos expuestos y las imágenes presentadas podemos concluir que:

$$\dot{V}(\tilde{\boldsymbol{q}},\dot{\boldsymbol{q}}) = -\dot{\boldsymbol{q}}^T K_v \frac{tanh(\dot{\boldsymbol{q}})(1+cosh(\dot{\boldsymbol{q}}))}{cosh(\dot{\boldsymbol{q}})} - \dot{\boldsymbol{q}}^T B\dot{\boldsymbol{q}} \leq 0 \tag{5.11}$$

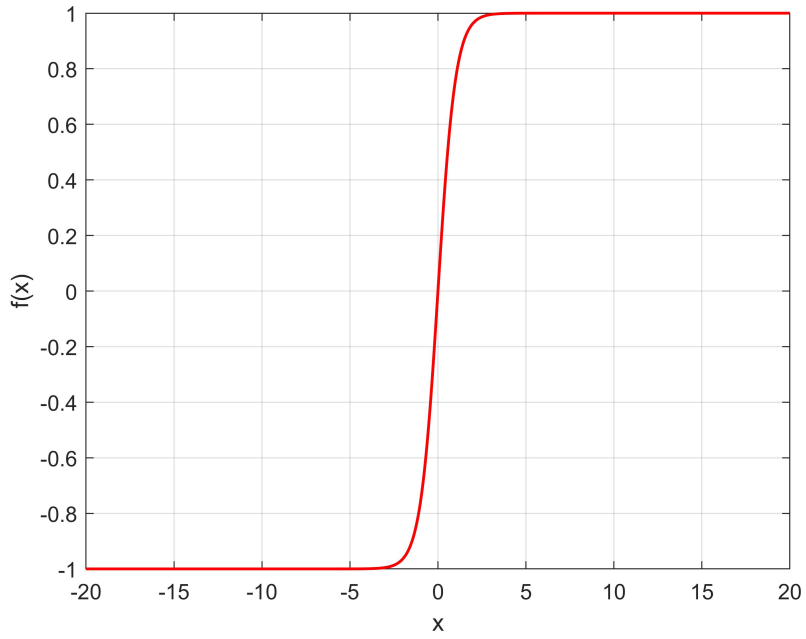

Figura 5.2: Comportamiento de la función $\frac{tanh(\boldsymbol{x})(\mathbf{1}+cosh(\boldsymbol{x}))}{cosh(\boldsymbol{x})}$

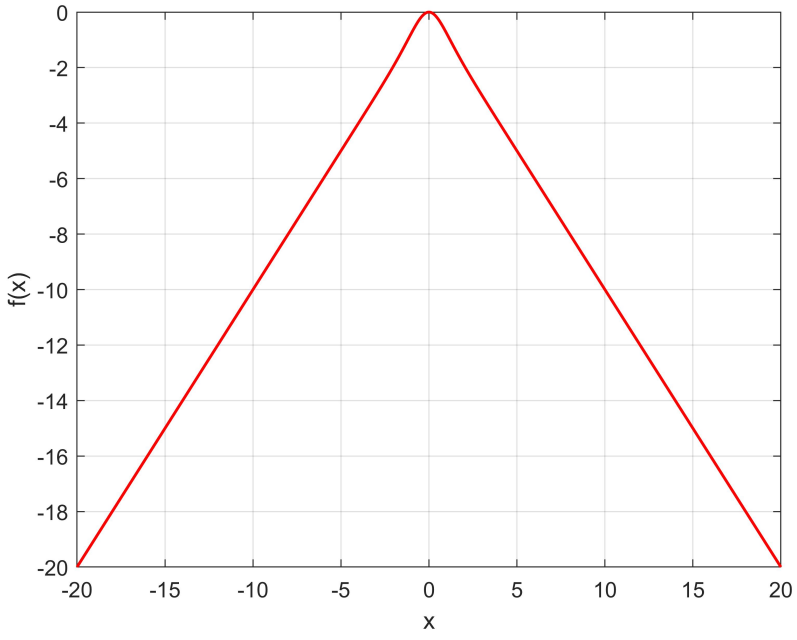

Figura 5.3: Comportamiento de la función $-\boldsymbol{x}^T \frac{tanh(\boldsymbol{x})(\mathbf{1}+cosh(\boldsymbol{x}))}{cosh(\boldsymbol{x})}$

La expresión 5.11 solamente demuestra que el punto de equilibrio es estable. Sin embargo podemos explotar la naturaleza autónoma de la ecuación en lazo cerrado aplicando el principio de invarianza de LaSalle. Para esto sabemos que la función candidata de Lyapunov es definida positiva en forma global, es radialmente no acotada y continuamente diferenciable. Ahora considere una región Ω tal que:

$$\begin{aligned}\boldsymbol{\Omega} = & \left\{ \begin{bmatrix} \tilde{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^{2n} | \dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) = 0 \right\} \\ & = \{ \tilde{\boldsymbol{q}} \in \mathbb{R}^n, \dot{\boldsymbol{q}} = 0 \in \mathbb{R}^n | \dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) = 0 \}\end{aligned}$$

ya que $\dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}}) \leq 0 \in \Omega$, entonces $\dot{V}(\tilde{\boldsymbol{q}}(t), \dot{\boldsymbol{q}}(t))$ es una función decreciente de t. $\dot{V}(\tilde{\boldsymbol{q}}, \dot{\boldsymbol{q}})$ es continua dentro del conjunto compacto Ω y es acotada inferiormente en Ω. Ya que Ω es un conjunto invariante, y el único conjunto que no varia es $\tilde{\boldsymbol{q}} = 0$ y $\dot{\boldsymbol{q}} = 0$ y ya que la solución trivial (unicidad y existencia del punto de equilibrio) es la única solución del sistema en lazo cerrado, y esta se encuentra restringida a Ω, entonces se concluye que el punto de equilibrio de la ecuación en lazo cerrado que se presentó es asintóticamente estable.

5.3. Resultados de la simulación

Se implementó en MATLAB el sistema en variables de estado dado por la ecuación en lazo cerrado 5.7. A continuación se muestra la matriz de transformación que tiene a la posición y orientación deseada en la regulación:

$$H_d = \begin{bmatrix} 1 & 0 & 0 & 0{,}2 \\ 0 & 1 & 0 & 0{,}2 \\ 0 & 0 & 1 & 0{,}2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

posteriormente se hace uso de la cinemática inversa para obtener los valores en el espacio articular. El vector de posiciones articulares deseadas es el siguiente:

$$\boldsymbol{q}_d = [3{,}6136 \ 4{,}8364 \ 2{,}5099 \ 3{,}6493 \ 1{,}5708 \ 4{,}2404]^T \tag{5.13}$$

Para el controlador propuesto se tuvo dificultad al momento de establecer las ganancias proporcional y derivativa con valores constantes, ya que se producía una singularidad en la ecuación en lazo cerrado. Como opción factible se optó por usar una estructura de ganancias variables estudiada en [12] para la parte proporcional y

derivativa. Las ganancias proporcionales quedan establecidas de la siguiente manera:

$$
\begin{aligned}
kp_1 &= 120(1 - 0{,}5e^{-0{,}5\tilde{q}_1^2}) \\
kp_2 &= 50(1 - 0{,}1e^{-0{,}1\tilde{q}_2^2}) \\
kp_3 &= 100(1 - 0{,}5e^{-0{,}5\tilde{q}_3^2}) \\
kp_4 &= 22(1 - 0{,}3e^{-0{,}3\tilde{q}_4^2}) \\
kp_5 &= 22(1 - 0{,}3e^{-0{,}3\tilde{q}_5^2}) \\
kp_6 &= 22(1 - 0{,}3e^{-0{,}3\tilde{q}_6^2})
\end{aligned}
$$

y las ganancias derivativas tomaron los siguientes valores:

$$
\begin{aligned}
kv_1 &= 100(1 - 0{,}4e^{-0{,}4\dot{q}_1^2}) \\
kv_2 &= 35(1 - 0{,}1e^{-0{,}1\dot{q}_2^2}) \\
kv_3 &= 80(1 - 0{,}4e^{-0{,}4\dot{q}_3^2}) \\
kv_4 &= 17(1 - 0{,}2e^{-0{,}2\dot{q}_4^2}) \\
kv_5 &= 17(1 - 0{,}2e^{-0{,}2\dot{q}_5^2}) \\
kv_6 &= 17(1 - 0{,}2e^{-0{,}2\dot{q}_6^2})
\end{aligned}
$$

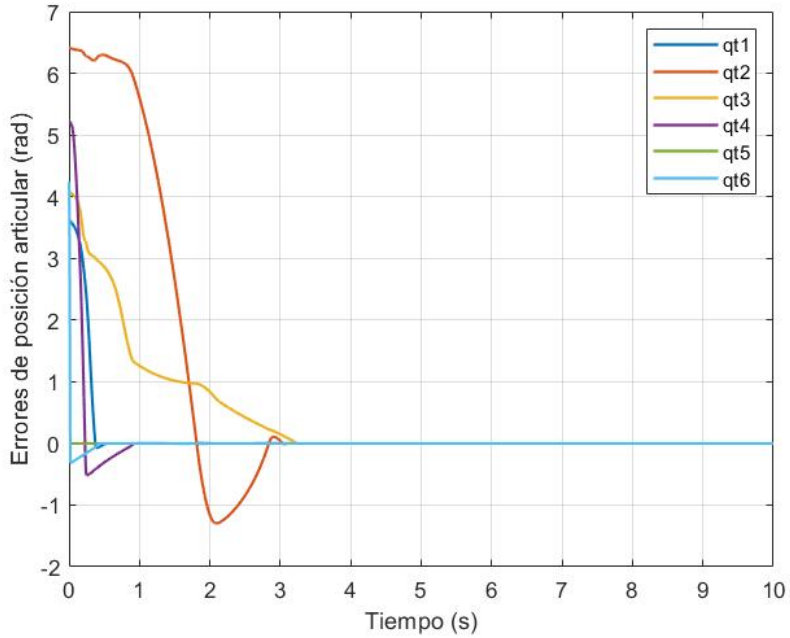

Figura 5.4: Errores de posición del esquema de control propuesto

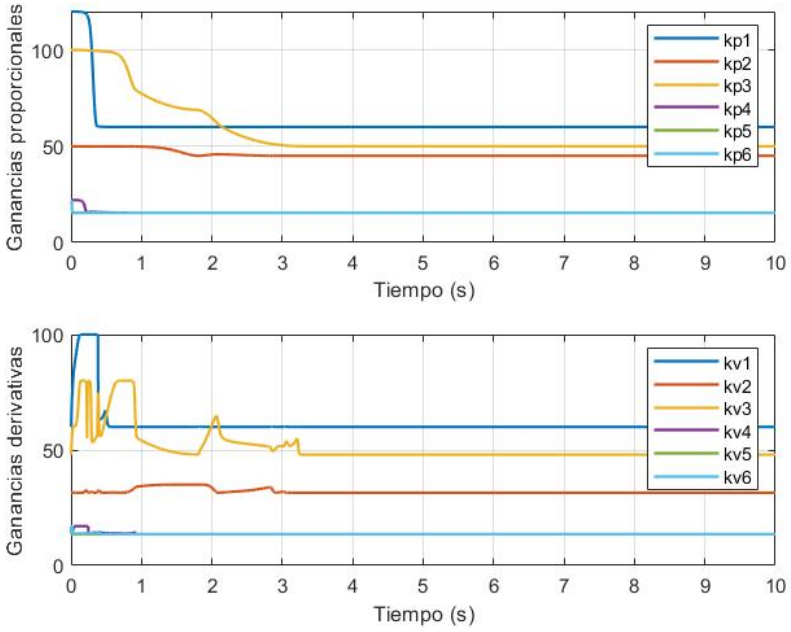

Figura 5.5: Ganancias utilizadas en el esquema de control propuesto

En la figura 5.4 podemos visualizar los errores de posición articulares, como se observa estos tienden a cero conforme el tiempo evoluciona, el tiempo de estabilización fue de 3.24s, lo que se traduce en que el esquema de control propuesto puede llevar a cabo de manera adecuada el control de posición. Por otra parte en la figura 5.5 se pueden visualizar los valores que las ganancias proporcionales y derivativas toman a lo largo del tiempo hasta que el error de posición y la velocidad toman valores de $\tilde{\boldsymbol{q}} = \mathbf{0}$ y $\dot{\boldsymbol{q}} = \mathbf{0}$ respectivamente y los valores de las ganancias se estabilicen. Finalmente en la figura 5.6 se pueden visualizar los valores de torque que solicita el regulador para llevar a cabo la tarea planeada.

5.4. Comparación de desempeño

Para comparar el desempeño del esquema de control propuesto se utilizo la norma $\mathcal{L}_2$, la cual se aplicó al error de posición, ecuación 5.14:

$$\mathcal{L}_2[\tilde{\boldsymbol{q}}] = \sqrt{\frac{1}{t_f}\int_0^{t_f} \|\tilde{\boldsymbol{q}}(t)\|^2 dt} \tag{5.14}$$

Como se comento anteriormente también se aplicó la norma $\mathcal{L}_2$ para evaluar el desempeño de los algoritmos PD (proporcional derivativo) y tangente hiperbólica aplicados a realizar la misma tarea que el esquema de control propuesto, los valores de norma $\mathcal{L}_2$ se muestran en la tabla 5.1, donde $\mathcal{L}_2[\tilde{\boldsymbol{q}}]$ representa la norma $\mathcal{L}_2$ aplicada al error de posición articular y t_s es el tiempo de estabilización del regulador.

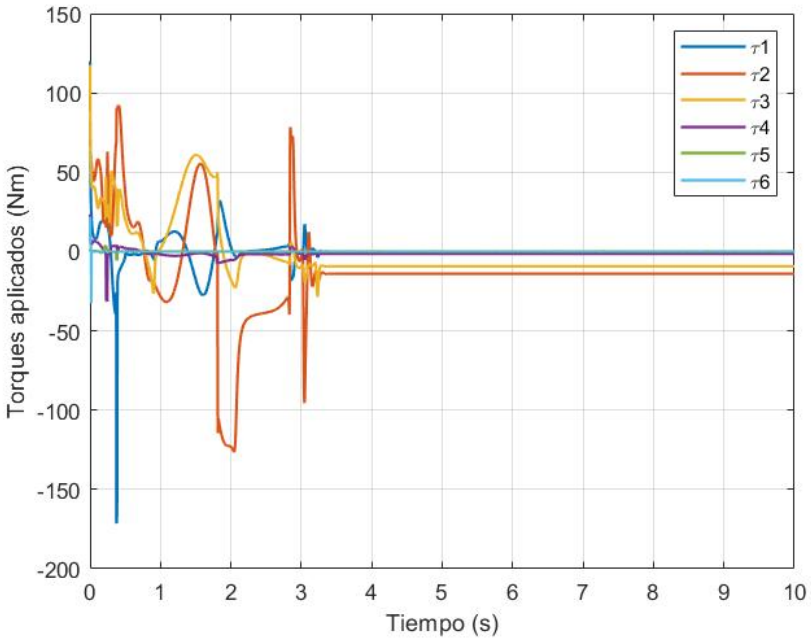

Figura 5.6: Torques aplicados en el esquema de control propuesto

Cuadro 5.1: Tabla de desempeño de los reguladores

Controlador	$\mathcal{L}_2[\tilde{\boldsymbol{q}}] t_s(s)$
Controlador propuesto	0.3773.337
PD	0.1850.612
tanh	0.38 3.09

El esquema de control propuesto es capaz de realizar adecuadamente la regulación o control de posición y poder llevar al efector final del robot manipulador a una posición y orientación deseada. Como se puede visualizar en la tabla 5.1 este esquema presenta un valor de $\mathcal{L}_2[\tilde{\boldsymbol{q}}]$ menor que el del regulador tangente hiperbólica sin embargo el regulador PD es el que presenta mejor desempeño en cuanto a valor de $\mathcal{L}_2[\tilde{\boldsymbol{q}}]$ y tiempo de estabilización t_s. Esto puede cambiar, proponiendo una sintonía diferente para cada controlador sin embargo la propia estructura del controlador propuesto requiere de una sintonización muy peculiar, ya que como se comento fue muy difícil lograr una sintonía con ganancias constantes. En estos casos existen varias alternativas para que un esquema de control pueda adquirir mejores características de desempeño como: ganancias variables, redes neuronales y algoritmos de optimización (genéticos, PSO, Ant Colony, Chicken Swarm). Por ultimo es importante conocer y destacar las diferentes técnicas que existen para proponer nuestros propios algoritmos de control y así poder crear nuevas alternativas en el campo del control de robots manipuladores.

Conclusiones

5.5. Conclusiones generales

- Del estudio de los algoritmos de planificación de ruta se derivó que no todos los algoritmos de planificación de ruta son aplicables a manipuladores, lo que nos orilló a realizar una investigación mas a detalle e identificar aquellos algoritmos que son potencialmente aplicables a un robot manipulador.

- Se concluyó que los algoritmos basados en muestreo y los métodos basados en campos potenciales artificiales son la mejor opción para implementarse en un manipulador debido a sus ventajas como son, consultas rápidas y exploración del espacio libre de configuraciones de manera homogénea.

- El modelo dinámico de un robot manipulador nos permite reproducir fielmente su comportamiento físico al momento de interactuar con el entorno. Este modelo es de vital importancia para desarrollar reguladores de posición o algoritmos de control de trayectoria.

- De los 2 controladores simulados con el CoBot UR5 el controlador Par Calculado presento un mejor desempeño en valor de norma L2 comparado con el controlador PD+.

- La principal ventaja del controlador Par Calculad radica en que su estructura matemática permite una cancelación completa del modelo dinámico del manipulador lo que se traduce en una disminución de los tiempos de calculo que se necesitan para simular el sistema en lazo cerrado.

- Los resultados obtenidos en cuanto a la simulación de los controladores con el modelo dinámico del manipulador nos permiten tener una perspectiva de que al momento de realizar una implementación práctica y en tiempo real el esquema Par Calculado seria el utilizado debido al desempeño obtenido en control de trayectoria.

- El nuevo esquema de regulación que se propone es capaz de llevar a cabo el control de posición que se propone sin embargo el control PD presenta un mejor desempeño en cuanto a valor de norma L2, sin embargo el control propuesto

presenta menor sobre-amortiguamiento en los torques al momento de realizar la regulación.

5.6. Trabajo futuro

- Como primera acción a realizar en trabajo futuro se pretende realizar la implementación física de los experimentos simulados, al tener ya los modelos del robot, los planificadores y las simulaciones realizadas con resultados satisfactorios es posible realizar la implementación en el robot físico.
- Posteriormente se piensa en mejorar el desempeño de los controladores de trayectoria utilizados es decir, en los experimentos realizados se utilizaron valores de ganancias constantes por lo que en trabajos posteriores se podría pensar en desarrollar y utilizar un esquema de ganancias variables para mejorar su desempeño.
- De igual forma se piensa mejorar el desempeño del regulador propuesto utilizando alguna técnica de PSO (particle Swarm Optimization) para que su desempeño sea mejor que el obtenido con el control PD.

Bibliografía

[1] D. A. López, "Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos", Tesis Doctoral, Departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática, Universidad de Huelva, Huelva, España, 2012.

[2] F. Reyes *Robótica Control de Robots Manipuladores.* Primera Edición. México: AlfaOmega, 2011.

[3] L. Kene and Q. Yang "Motion Planning of Robot Manipulator for Cucumber Picking" in 3rd International Conference on Robotics and Automation Engineering, Guangzhou, China, 2018, pp. 50-54.

[4] C. Abdymanapov and A. Popov "Motion Planning Algorithms Using DISC", in 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering(EIConsRus), Saint Petersburg and Moscow, Russia, Russia, 2019, pp. 1844-1847.

[5] B. Gillespie and M.Peshkin "A General Framework for Cobot Control", *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 391-401, Agosto 2001.

[6] G. Carbone and F. Gomez, "Motion and Operation Planing of Robotic Systems", Italia: Springer, 2015.

[7] L. Sukhan and L. Yeonho "Real-Time Industrial Bin-Picking with a Hybrid Deep Learning-Engineering Approach" in 2020 IEEE International Conference on Big Data and Smart Computing(BigComp), Busan, Korea (South), 2020, pp. 584-588.

[8] L. Guohong and W. Yuanliang "Industrial Robot Optimal Time Trajectory Planning Based on Genetic Algorithm" in International Conference on Mechatronics and Automation, Tianjin, China, 2019, pp. 136-140.

[9] I. El Makrini "How a Cobot Was Developed and Inserted on an Auto Assembly Line", *IEEE Robotics and Automation Magazine*, vol. 25, pp. 2-9, Mayo 2018.

[10] E. Kyrkjebø"Feasibility of the UR5 Industrial Robot for Robotic Rehabilitation of the Upper Limbs After Stroke" in International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 6124-6129.

[11] S. Moe, J. T. Gravdahl and K. Y. Pettersen. "Set-Based Control for Autonomous Spray Painting", *IEEE Transactions on Automation Science and Engineering*, vol. 15, pp. 1-12, Marzo 2018.

[12] Manual de UR5 de Universal Robots

[13] Z. Cheng "Joint Trajectory Time Optimization of Cobot Based on Particle Swarm Optimization" in IOP Conf. Series: Materials Science and Engineering, Shanghai, China, 2019, pp.1-7.

[14] Y. Mu *et al.,* "Optimal Trajectory Planning for Robotic Manipulators Using Chicken Swarm Optimization" in 8th International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 2016, pp. 369-373.

[15] W. Chaoqun *et al.,* "Efficient Autonomous Robotic Exploration With Semantic Road Map in Indoor Environments", *IEEE ROBOTICS AND AUTOMATION LETTERS*, vol. 4, pp. 2989-2996, Julio 2019.

[16] L. Yi, G. Chen and W. Yongpeng "Online Time-Optimal Trajectory Planning for Robotic Manipulators Using Adaptive Elite Genetic Algorithm With Singularity Avoidance", *IEEE Access*, vol. 7, pp. 146301-146308, Octubre 2019.

[17] Sitio oficial de UR en www.universal-robots.com/mx/

[18] R. Skovgaard "Kinematics of a UR5", Aalborg University, Mayo 2018.

[19] K. P. Hawkins "Analytic inverse kinematics for the universal robots ur-5/ ur-10 arms", Technical report, Georgia Institute of Technology.

[20] R. Keating "Analytic inverse kinematics for the universal robots ur-5/ur-10 arms", Technical report, Jhon Hopkins, 2017.

[21] P. M. Kebria *et al.,* " Kinematic and Dynamic Modelling of ur5 Manipulator" in Systems Man and Cybernetics (SMC), IEEE International Conference, 2016, pp. 004229-004234.

[22] M. W. Spong and M. Vidyasagar "Robot Dynamics and Control", Jhon Wiley & Sons, 2008.

[23] J.J. Craig *Introduction to Robotics: Mechanics and Control* Pearson/Prentice Hall Upper Saddle River, NJ, USA, 2005.

[24] H. Choset *et al., Principles of motion: theory, algorithms and implementation* The MIT Press, London, England, 2005.

[25] L. Kravaki and J. C. Latombe "Randomized Preprocessing of Configuration Space for Path Planning: Articulated Robots" in International Conference on Intelligent Robots and Systems (IROS), 1994, pp. 1764-1762.

[26] L. Kravaki and J. C. Latombe "Randomized Preprocessing of configuration Space for Fast Path Planning" in International Conference on Robotics and Automation, San Diego, 1994, pp. 2138-2146.

[27] L. Kravaki *et al.,* "Probabilistic Road Map for Path Planning in High - Dimensional Configuration Spaces", *IEEE Trans. on Robotics and Automation*, vol.12, pp. 566-580, 1996.

[28] R. Bohlin and L. Kravaki "Path PLanning Using Lazy PRM" in International Conference on Robotics and Automation, San Francisco, CA, Abril, 2002, pp. 521-528.

[29] D. Hao *et al.,* "Robotic Manipulation Planning Using Dynamic RRT" in International Conference on Real-Time Computing and Robotics, Angkov Wat, Cambodia, Junio 2016, pp. 500-504.

[30] Z. Ning *et al.,*"Path Planning of Six-DOF Serial Robots Based on Improved Artificial Potential Field Method" in IEEE International Conference on Robotics and Biomimetics (ROBIO), Deciembre 2017.

[31] C. Zhongyi, Ma Lin and Shao Zhijiang "Path Planning for Obstacle Avoidance of Manipulators Based on Improved Artificial Potential Field" in Chinese Automation Congress (CAC), Hangzhou, China, Noviembre 2019, pp. 2991-2996.

[32] D. López, F. Gómez-Bravo and F. Cuesta "El algoritmo RRT. Aplicación a robots no holónomos", *Revista Iberoamericana de Automática e Informática Industrial*, vol.3, pp.56-67, Julio 2006.

[33] M. Clifton *et al.,* "Evaluating Performance of Multiple RRTs" in IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications, Beijing, China, Octubre 2008, pp. 564 - 569.

[34] M. Chao *et al.,* "6R Serial Manipulator Space Path Planning Based on RRT" in INternational Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, Diciembre 2016, pp. 99-102.

[35] W. Wenrui, Z. Mingchao and W. Xiaoming "An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators", *International Journal of Advanced Robotic Systems*, pp. 1-13, Agosto 2018.

[36] Z. Jianying, Z. Zhiping, L. Dun "A path planning method for mobile robot based on artificial potential field" , *Journalof Harbin Institute of Technology*, vol.38, pp. 1306-1309, 2006.

[37] H. Zhaochu, H. Yuanlie and Z. Bi "Obstacle avoidance pathplanning for robot arm based on mixed algortihm of artificial potential field method and RRT", *Industrial Engineering Journal*, vol.20(2):56-63,2017.